PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

APRENDIZAGEM DE ONTOLOGIAS PARA O APOIO AO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE ORIENTADO A CONHECIMENTO

FABIANA WINOVSKI DORNELES

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Marcelo Blois Ribeiro

Porto Alegre
2011

Dados Internacionais de Catalogação na Publicação (CIP)

D713a Dorneles, Fabiana Winovski
Aprendizagem de ontologias para o apoio ao processo de desenvolvimento de software orientado a conhecimento / Fabiana Winovski Dorneles. – Porto Alegre, 2011.
139 p.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Marcelo Blois Ribeiro.

1. Informática. 2. Engenharia de Software. 3. Ontologia. I. Ribeiro, Marcelo Blois. II. Título.

CDD 005.1

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "**Aprendizagem de Ontologias para Apoio ao Processo de Desenvolvimento de Software Orientado a Conhecimento**", apresentada por Fabiana Winovski Dorneles, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas de Informação, aprovada em 24/03/2011 pela Comissão Examinadora:

Prof. Dr. Marcelo Blois Ribeiro– Orientador — PPPGCC/PUCRS

Prof. Dr. Ricardo Melo Bastos - — PPGCC/PUCRS

Prof. Dr. José Palazzo Moreira de Oliveira – — UFRGS

Homologada em 28../.06../.11....., conforme Ata No. .11/11... pela Comissão Coordenadora.

Prof. Dr. Fernando Luís Dotti
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320–3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

"E não somente isso, mas também gloriemo-nos nas tribulações; sabendo que a tribulação produz a perseverança, a perseverança a experiência, e a experiência a esperança; e a esperança não desaponta, porquanto o amor de Deus está derramado em nossos corações pelo Espírito Santo que nos foi dado."

Rm 5:3-5

AGRADECIMENTOS

Tenho muito a agradecer, e assim faço primeiramente a Deus que me guiou durante esta caminhada. Mesmo nos momentos mais difíceis, a presença de Deus sempre esteve em minha vida, trazendo alegrias onde havia tristeza, esperança onde existiam desilusões, fazendo permanecer sempre a fé e a coragem em todos os momentos.

Aos pais agradeço o apoio recebido e os anos de amor e dedicação. O amor é muito mais do que uma simples palavra definida no dicionário. É através do amor que percebemos a união de todas as coisas com perfeição, e o verdadeiro amor é expresso diariamente pelos meus pais, nas inúmeras vezes que se colocaram de lado para me oferecer o apoio necessário.

Aos amigos, que estiveram ao meu lado no incentivo ao estudo e troca de conhecimentos e experiências profissionais, meu eterno reconhecimento.

Ao meu esposo pelo apoio e carinho, mas principalmente por entender o quanto priorizo o estudo e o amor que dedico à minha profissão.

Aos professores da instituição, agradeço por me acompanhar, incentivar e agregar conhecimento no cumprimento desta etapa da minha vida. Em especial, agradeço ao meu orientador, pelo tempo dedicado, pela transmissão de conhecimento e coragem e por oportunizar a realização do meu sonho de cursar o mestrado.

Ao convênio *Dell* / PUCRS por viabilizar a bolsa de estudos.

Aos meus colegas e amigos, por disponibilizar um tempo precioso para o preenchimento da *survey* aplicada neste trabalho, me auxiliando no cumprimento das atividades do curso.

Por fim, tomando por verdade a frase dita por Albert Einstein "*Se quer viver uma vida feliz, amarre-se a uma meta, não às pessoas nem às coisas*", concluo o curso tendo cumprido minha meta, com o apoio de todos os anjos que cruzaram meu caminho.

APRENDIZAGEM DE ONTOLOGIAS PARA O APOIO AO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE ORIENTADO A CONHECIMENTO

RESUMO

As ontologias possibilitam o formalismo semântico e permitem o desenvolvimento de aplicações utilizando-as como base de conhecimento, fornecendo assim melhor representação dos dados. Esta dissertação propõe uma abordagem para solucionar os três principais desafios da utilização de ontologias em Engenharia de Software: possibilitar a modelagem do conhecimento dispersa nos artefatos de software; viabilizar a criação da ontologia de modo semi-automático, em contrapartida ao oneroso processo de criação manual; e trazer maior formalismo ao processo de desenvolvimento de software, uma vez que a ontologia é processável por máquina. Para a definição da abordagem utilizou-se das áreas de conhecimento existentes, como a aplicação de um conjunto de técnicas de Processamento da Linguagem Natural para identificar e extrair as informações relevantes contidas nos detalhamentos de casos de uso. Sendo assim, tornou-se possível a geração da ontologia inicial, a partir do mapeamento do diagrama de classes que descreve o modelo de domínio do software, e seu posterior refinamento, através do detalhamento dos casos de uso. O resultado final é um conjunto de tuplas extraídas a partir da descrição detalhada dos casos de uso. Através da inspeção manual é gerada uma ontologia refinada de modo semi-automático, minimizando, com isto, a intervenção humana na construção da ontologia. Por fim, a ontologia final descreve a solução de software proposta, possibilitando o melhor entendimento dos conceitos relacionados à solução que está sendo construída.

Palavras-chave: ontologia; Engenharia de Software (ES); Processo Racional Unificado (RUP).

ONTOLOGY LEARNING FOR SUPPORTING SOFTWARE DEVELOPMENT PROCESS KNOWLEDGE-ORIENTED

ABSTRACT

Ontologies enable the semantic formalism and enable the development of applications, using them as a knowledge base, and providing a better data representation. This dissertation proposes an approach that addresses the three major challenges to the use of ontologies in Software Engineering: enabling the modeling of knowledge scattered among software artifacts; meaking feasible a semi-automatic generation of a ontology, in contrast to the costly process of manual creation; and bring more formality to the process of software development, since the ontology is machine processable. For this, we used the knowledge areas to the approach definition, such as applying a set of techniques of Natural Language Processing to identify and extract relevant information contained in some deployments of use cases. Thus, it was possible to generate the initial ontology, from the mapping of the class diagram that describes the domain model of the software, and its subsequent refinement using the details of use cases. The result is a set of tuples extracted from the detailed use cases. By manual inspection is generated semi-automatically a refined ontology, there by minimizing the need for human intervention in the ontology construction. Finally, the ontology describes the final proposal software solution, enabling a better understanding of the concepts related to the solution that is being built.

Keywords: ontology; Software Engineering (SE); Rational Process Unified (RUP).

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AO – Aprendizagem de Ontologia

BD – Banco de Dados

ES – Engenharia de Software

EO – Evolução de Ontologia

IA – Inteligência Artificial

KDD – Descoberta de Conhecimento em Banco de Dados

LN – Linguagem Natural

PLN – Processamento da Linguagem Natural

POS – *Part of Speech*

RUP – *Rational Process Unified*

TI – Tecnologia da Informação

UML – *Unified Modeling Language*

UP – *Unified Process*

SUMÁRIO

1. INTRODUÇÃO

A ontologia exerce o papel de apoiar a atividade de compartilhar o conhecimento de um determinado domínio [CHE07]. Um conceito de uma ontologia pode ser definido como o significado semântico de um termo em um determinado domínio. Um domínio é caracterizado por um conjunto de propriedades que o descrevem (uma área do conhecimento). O conhecimento do domínio é representado através de um formalismo declarativo, refletido em um vocabulário. Gruber define a ontologia como uma especificação explícita de uma conceituação [GRU93].

As ontologias possibilitam o formalismo semântico e permitem o desenvolvimento de aplicações utilizando-as como base de representação do conhecimento. São utilizadas com a finalidade de permitir a interoperabilidade entre aplicações, disponibilizando conceitos comuns e formais do domínio [CHE07]. Sua finalidade é caracterizar uma conceituação, limitando as interpretações possíveis e estabelecendo um consenso sobre o conhecimento por ela representado [GUA97]. As ontologias também podem revelar-se muito úteis quando utilizadas com a finalidade de explicitar formalmente a semântica dos artefatos durante o processo de desenvolvimento de software. Abaixo encontram-se listadas algumas das vantagens de utilizar ontologias no processo de desenvolvimento de software [ASS10], [CHE07], [MAE03]:

- Permitir o compartilhamento do entendimento comum das informações.
- Possibilitar a reutilização do conhecimento do domínio.
- Explicitar as informações do domínio, através do entendimento e atualização dos dados.
- Fornecer a formalização dos termos, descrevendo o domínio.

Embora a utilização de ontologias para o apoio ao processo de desenvolvimento de software traga muitos benefícios por proporcionar a formalização do conhecimento, o processo de criação manual da ontologia é definido por Maedche e Staab [MAE03] como “uma tarefa tediosa que pode facilmente resultar em um gargalo na aquisição do conhecimento”.

A Aprendizagem de Ontologia (do termo inglês O*ntology Learning*, inicialmente definido por Maedche e Staab [MAE01]) é descrita como uma atividade que envolve diferentes áreas de pesquisa, sob a perspectiva de um processo de aquisição e definição

de um modelo de domínio a partir de diferentes tipos de dados [MAE03]. A AO pode ser definida como “um suporte semi-automático ou automático para construção de ontologias” [CIM06]. Na literatura, dentre as áreas de pesquisa, é encontrada a utilização de algoritmos de aprendizagem de máquina, Processamento da Linguagem Natural (PLN), Mineração de Dados e Recuperação de Informações a partir de bases textuais, com a finalidade de auxiliar na aquisição do conhecimento relacionado à aprendizagem de ontologias.

Este trabalho apresenta uma abordagem que utiliza técnicas de PLN existentes para viabilizar a criação e o refinamento de ontologias que expressem o conhecimento contido nas descrições detalhadas de casos de uso, geradas no processo de desenvolvimento de software. A ontologia inicial é gerada a partir do mapeamento do modelo de domínio descrito através do diagrama de classes. O refinamento e enriquecimento da ontologia inicial utilizaram-se da aplicação de um conjunto de técnicas de PLN para identificar e extrair as informações relevantes contidas nos detalhamentos de casos de uso que descrevem a aplicação em desenvolvimento. Através da utilização destas técnicas, tornou-se possível a identificação de novos relacionamentos e conceitos, resultando em uma ontologia final, gerada e refinada de modo semi-automático, minimizando com isto a necessidade de intervenção humana na construção de ontologias para o apoio ao processo de desenvolvimento de software.

1.1 Definição do Problema

Na literatura, referente à Inteligência Artificial (IA), encontra-se diversas propostas para auxiliar o processo de geração automática de uma ontologia. No contexto de Engenharia de Software (ES), a proposta é utilizar o conhecimento contido nas descrições detalhadas de casos de uso, que descrevem os requisitos do sistema de software em construção, para enriquecer e refinar ontologias que representem a semântica do problema. Os casos de uso contêm requisitos que descrevem as funções e restrições de um sistema de software. Casos de uso são comumente utilizados para elicitação de requisitos de software [SOM08] e são descritos em linguagem natural. A interpretação e recuperação das informações contidas nestas descrições detalhadas não são triviais, embora exista grande riqueza de informações nestes documentos.

1.2 Questão de Pesquisa

Com a finalidade de propor uma solução ao problema, surge a questão de pesquisa:

"Como criar e refinar uma ontologia de forma semi-automática a partir do diagrama de classes que descreve o modelo de domínio e da descrição detalhada dos casos de uso gerados no processo de desenvolvimento de software?".

1.3 Objetivos

Maedche e Staab [MAE03] definem a construção de uma ontologia como "uma tarefa tediosa que pode facilmente resultar em um gargalo na aquisição do conhecimento", representando um processo manual oneroso. Observando a definição dos autores da área, os objetivos deste trabalho incluem:

- Facilitar o trabalho de criação, refinamento e enriquecimento da ontologia;
- Garantir o mapeamento do conhecimento de modo mais abrangente.
- Desenvolver uma abordagem para realizar a identificação de conceitos e relacionamentos da ontologia;
- Trazer maior formalismo ao processo de desenvolvimento de software, pois através da linguagem OWL que descreve a ontologia, é possível o processamento por máquina.

1.4 Estrutura da dissertação

O texto está dividido em 7 Capítulos. No Capítulo 2, encontra-se descrita a fundamentação teórica necessária para compreensão do contexto em que está inserido este trabalho. O Capítulo 3 descreve os trabalhos relacionados a esta pesquisa e apresenta as considerações sobre cada trabalho. O Capítulo 4 apresenta as etapas da abordagem definida neste trabalho. O Capítulo 5 apresenta a ferramenta desenvolvida para apoiar a abordagem proposta neste trabalho. Esse Capítulo também inclui o diagrama de classes, contendo as principais classes da ferramenta para melhor entendimento. O Capítulo 6 descreve dois exemplos de utilização da abordagem, para dois sistemas descritos na literatura, os resultados obtidos e as considerações finais sobre

os resultados. Nesse Capítulo também é apresentada a avaliação da abordagem através da aplicação de uma *survey*, respondida por 10 participantes. Por fim, o Capítulo 7 apresenta sugestões de trabalhos futuros e a conclusão final desta pesquisa.

2. FUNDAMENTAÇÃO TEÓRICA

Este Capítulo apresenta a fundamentação teórica necessária para entendimento deste trabalho. O Capítulo inicia com a definição do termo ontologia, contextualizando os principais conceitos da área. Apresenta ainda a revisão teórica das áreas de Aprendizagem de Ontologia e Evolução da Ontologia (termos traduzidos de *Ontology Learning* e *Ontology Evolution* respectivamente). Para o entendimento destas áreas de estudo, estão contextualizadas as várias técnicas listadas na literatura que viabilizam a AO. Por fim, encontra-se descrito resumidamente o *Rational Process Unified* (RUP), contexto que está inserido esta pesquisa.

2.1 Ontologia

O termo grego ontologia advém da filosofia e significa o “conhecimento do ser”. As ontologias são modelos sobre vocabulários e seus significados, definidos explicitamente, expressivamente e semanticamente, possibilitando a interpretação por máquina [DAC03]. Fornecem um modelo explícito para estruturação de conceitos e representação de suas relações [SUN08]. Em IA, a representação de conceitos e relacionamentos desempenha papel fundamental na gestão do conhecimento, pois fornecem um modelo de representação para solucionar problemas decorrentes da incapacidade das máquinas de compreender a Linguagem Natural (LN) [SUN08].

No contexto da computação, uma das definições mais citadas é fornecida por Gruber, onde: “Ontologia é uma especificação explícita de uma conceituação” [GRU93]. Uma definição formal para ontologia é especificada por uma estrutura conforme apresenta a Figura 1.

$$\mathcal{O} := (C, \leq_C, R, \sigma_R, \leq_R, A, \sigma_A, T)$$

Figura 1 - Especificação formal da ontologia [CIM06].

A estrutura apresentada por Cimiano [CIM06] define:

- O identificador *C* para conceitos, *R* relações, *A* atributos e *T* tipos de dados;
- ≤ *c*, representa a hierarquia ou taxonomia de conceitos;

- σ_R, representa a assinatura da relação, que corresponde ao relacionamento entre dois conjuntos de conceitos *C*, o domínio e o *range*;
- $\leq_R$, representa a hierarquia das relações;
- σ_A, representa a assinatura dos atributos, que corresponde ao relacionamento entre os conjuntos *C* e *T*.

As ontologias descrevem alguns componentes como classes, indivíduos, atributos e relacionamentos. As classes representam conceitos do domínio (ou parte do mundo), e podem estar organizadas hierarquicamente, contendo outras subclasses. Elas descrevem um grupo de indivíduos que compartilham propriedades [W3C10]. Os conceitos são representados por termos que descrevem a sua semântica em um determinado domínio. Os indivíduos representam os objetos, e são instâncias das classes e propriedades. Os relacionamentos ou relações são responsáveis pela criação da taxonomia hierárquica da ontologia, estabelecendo uma estrutura de árvore onde descrevem a relação entre objetos. Eles descrevem a semântica do domínio, representando a relação de inclusão (é superclasse ou subclasse de), ou descrevem outros relacionamentos específicos do domínio modelado, refinando a semântica da ontologia. Como exemplo, é possível citar: “Java” é um tipo de “Linguagem_Orientada_a_Objetos” que é um tipo de “Linguagem_de_Programação”. Os atributos são as descrições dos objetos, possuindo um nome e valor. No exemplo do objeto Java, um atributo válido é a versão.

O compartilhamento das informações é muito difícil, à medida que são comumente utilizados diferentes conceitos para um mesmo domínio, o que dificulta o reuso da informação. Em IA, o conceito de ontologia está relacionado à especificação de conceitos, definindo termos e relacionamentos entre eles [HEN01]. Na computação, a formalização do conhecimento, tornando-o acessível através da ontologia, possibilita o compartilhamento, utilização e reutilização da informação nela contida, entre sistemas e indivíduos. As ontologias possibilitam o formalismo semântico e permitem o desenvolvimento de aplicações utilizando-as como base de conhecimento e fornecendo melhor representação dos dados. São utilizadas com a finalidade de permitir a interoperabilidade entre aplicações, disponibilizando conceitos comuns e formais do domínio [CHE07]. No contexto de ES, as ontologias podem ser importantes aliadas para explicitar formalmente a semântica dos artefatos durante o processo de desenvolvimento

de software, servindo como base de conhecimento da aplicação. Elas podem também ser utilizadas para explicitar decisões durante o processo de desenvolvimento de software.

Péres [PER04] apresenta a representação do conhecimento através de ontologias aplicadas em diversos domínios, entre eles:

- Comércio eletrônico (*e-commerce*);
- Medicina;
- Engenharia;
- Gestão de conhecimento.

O processo de desenvolvimento de uma ontologia envolve a escolha da linguagem de representação que será utilizada. A descrição das ontologias é realizada por meio de linguagens baseadas na lógica, consistentes e que suportam a representação de conhecimento. Na literatura encontram-se descritas várias linguagens de marcação para representação da ontologia, conforme apresentado abaixo [HAR03] [PER04] [W3C10]:

- SHOE (*Simple HTML Ontology Extension*): Primeira linguagem de marcação para ontologias. A linguagem é uma extensão do HTML e combina frames e regras. Posteriormente a linguagem foi adaptada para o XML.
- XOL (*XML-based Ontology exchange Language*): linguagem de marcação que estende o XML e apresenta um subgrupo do protocolo OKBC (*Open Knowledge Base Connectivity*).
- RDF (*Resource Description Framework*): linguagem base desenvolvida pela W3C para descrever recursos da web.
- RDFS (*Resource Description Framework Schema*): linguagem desenvolvida pela W3C que estende RDF.
- OIL (*Ontology Interchange Language*): linguagem baseada na XOL que estende uma estrutura básica para possibilitar a captura de uma lógica de descrição.
- DAML-OIL (*Darpa Agent Markup Language*): linguagem desenvolvida como uma extensão do XML e RDFS.

- OWL (*Ontology Web Language*): linguagem que deriva e substitui a DAML-OIL.

Atualmente, a linguagem comumente utilizada é a OWL. A linguagem OWL é atualmente indicada pela W3C por adicionar maior vocabulário para a descrição de classes e propriedades como, por exemplo, as relações de disjunção e cardinalidade entre as classes [W3C10].

A seguir, apresenta-se uma breve descrição das áreas de aprendizagem de ontologia (AO) e evolução da ontologia (EO).

2.2 Aprendizagem de Ontologia (AO)

A aprendizagem de ontologia (AO), assim como qualquer outra aprendizagem, só é útil e possível se motivada pelo contexto e necessidades dos indivíduos [LOO06]. A AO é a área de conhecimento relacionada à necessidade da ontologia em aprender novos conceitos durante o decorrer do tempo. Para que o conhecimento possua sentido, ele deve estar organizado de modo estruturado. A aprendizagem de novos conceitos relevantes é necessária devido a alterações em um domínio.

A aprendizagem de ontologia inclui as seguintes tarefas [CIM06]:

- Aquisição de terminologia relevante;
- Identificação de termos que são sinônimos e variações lingüísticas;
- Formação de conceitos;
- Organização hierárquica do conhecimento (taxonomia das classes);
- Aprendizagem de relacionamentos entre conceitos, propriedades ou atributos, conforme domínio de conhecimento da ontologia;
- Organização hierárquica dos relacionamentos (taxonomia dos relacionamentos);
- Instanciação dos axiomas;
- Definição de axiomas gerais.

A Figura abaixo ilustra as etapas descritas em camadas.

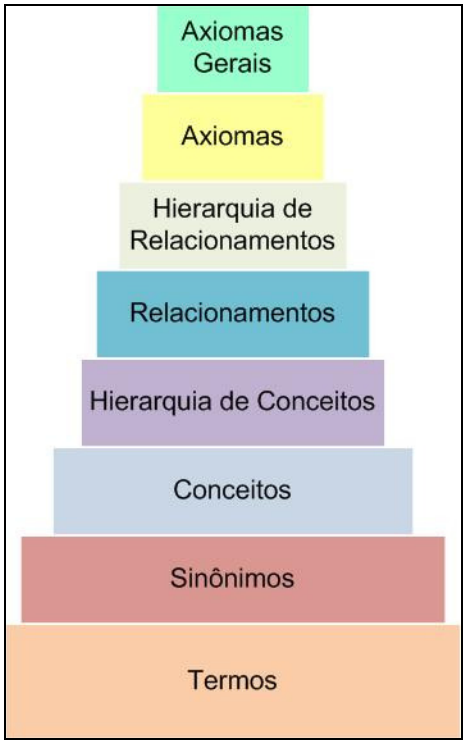

Figura 2 - Aprendizagem de ontologia adaptado de [CIM06].

A extração de termos durante a aprendizagem da ontologia é fundamental [CIM06]. A etapa consiste na identificação de um conjunto de termos relevantes para o domínio. Após a identificação dos termos, a etapa de reconhecimento de sinônimos consiste na tarefa de reconhecer palavras que representam os mesmos conceitos no domínio. A formação de conceitos a partir dos termos relevantes consiste na definição intencional e extensão dos conceitos. Posteriormente, a etapa de hierarquia de conceitos prevê a definição hierárquica de um determinado conjunto de conceitos.

A aprendizagem de relacionamentos envolve a tarefa de identificar as relações entre os conceitos no domínio apropriado, mesmo não havendo relações taxonômicas. A próxima etapa é denominada de hierarquia de relações e determina a ordem entre os relacionamentos. A tarefa de esquemas de axiomas refere-se à identificação de conceitos, relações ou pares de conceitos aplicados ao sistema que devem ser instâncias de axiomas. Os axiomas gerais correspondem à aprendizagem de axiomas e não apenas as instanciações.

Maedche [MAE02] propõe a aprendizagem de ontologia como um processo orientado que inclui as fases de importação, extração, corte e refinamento. A Figura 3 apresenta o ciclo da aprendizagem de ontologia.

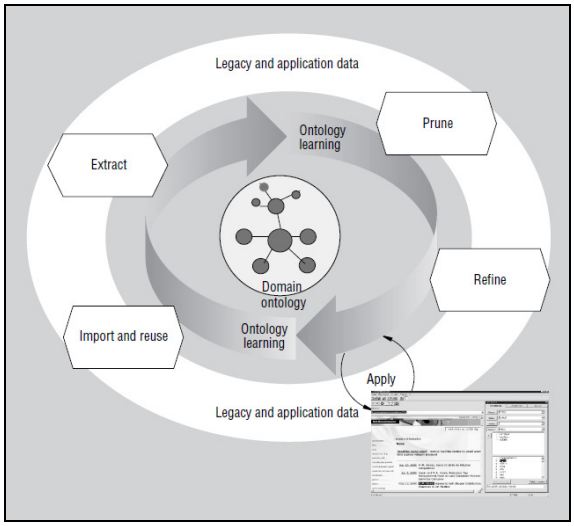

Figura 3 - Ciclo da aprendizagem de ontologia [MAE01].

A fase de importação sugere a reutilização ou a fusão de ontologias existentes ou a definição de regras de mapeamento entre estas estruturas conceituais. A extração de modelos corresponde à maior fase e refere-se à aprendizagem através da identificação de vários tipos de dados relevantes. Ela corresponde à inclusão de novos domínios ou a manutenção e atualização da ontologia. Na fase de corte, a ontologia resultante das fases de importação, reutilização e extração, sofre a “supressão” dos novos conceitos, com a finalidade de representar melhor o domínio. A última fase corresponde ao refinamento com menor granularidade da ontologia [MAE01] [MAE02].

Maedche relata a realização da validação da ontologia resultante do processo de aprendizagem a partir da cobertura dos conceitos, obtida no domínio da aplicação alvo [MAE02].

A aprendizagem da ontologia inclui várias técnicas [SUN08]:

- Técnicas de aprendizagem de máquina;
- Mineração de dados;
- Processamento da Linguagem Natural (PLN).

O entendimento dos formalismos e técnicas disponíveis é necessário para aplicação destas durante o processo de AO. A aprendizagem da ontologia inclui várias técnicas. A técnica relevante para a criação da abordagem apresentada neste trabalho, de acordo com o domínio desta pesquisa, é a utilização do PLN no processamento dos documentos gerados durante o desenvolvimento de software para posterior identificação de termos relevantes e candidatos a conceitos na geração e refinamento da ontologia. A seguir estão descritas cada uma destas técnicas.

2.2.1 Aprendizagem de Máquina

A aprendizagem de máquina pode ser definida como o reconhecimento e detecção de padrões dentro de um conjunto de dados [CIM06]. O reconhecimento de padrões pode ser utilizado para descrever os dados ou realizar previsões.

A aprendizagem de máquina supervisionada é comumente utilizada para predizer categorias. Existem algoritmos de aprendizagem que possibilitam a construção de um modelo para grupos de treino. A classificação é uma tarefa supervisionada, onde os dados de treinamento são marcados. O resultado são conjuntos de dados treinados e classificados. Os algoritmos de classificação possuem diferentes paradigmas. Alguns destes paradigmas são apresentados abaixo:

- Redes neuronais: rede de neurônios interconectados, formando um sistema dinâmico, onde o novo estado é função do estado anterior. Uma rede neural pode ser formalizada matematicamente, independente da topologia modelada [BIT06].
- Árvores de decisão: construída de forma iterativa, cada etapa executa um algoritmo de aprendizado responsável por criar um novo “ramo” para cada valor possível, onde um dos elementos restantes é escolhido [CIM06].
- Classificadores Bayesianos: modelagem estatística baseada no teorema de Bayes [TAN06]. O aprendizado é realizado através de modelos probabilísticos, onde assume a hipótese de que todos os elementos são independentes e possuem a mesma relevância [CIM06].

Métodos de aplicação de aprendizagem de máquina em grandes bases de dados são chamados de mineração de dados [ALP04]. A seguir apresenta-se uma breve descrição sobre a área de estudo de mineração de dados.

2.2.2 Mineração de Dados

A mineração de dados é uma etapa da área de Descoberta de Conhecimento em Banco de Dados (*Knowledge Discovery from Data - KDD*). Refere-se à extração ou mineração de conhecimento em uma grande base de dados [HAN01].

A descoberta de conhecimento (KDD) compreende as seguintes etapas [HAN01]:

- Limpeza dos dados: a etapa consiste na remoção de ruídos e dados inconsistentes;
- Integração dos dados: a etapa consiste na combinação de múltiplas fontes de dados;
- Seleção dos dados: a etapa consiste na seleção dos dados relevantes do Banco de Dados (BD);
- Transformação dos dados: a etapa consiste na transformação ou consolidação dos dados de modo apropriado à realização da mineração;
- Mineração dos dados: processo de métodos inteligentes que são aplicados de modo a extrair padrões de dados;
- Avaliação de padrões: a etapa consiste na identificação de padrões que representam o conhecimento de interesse.
- Apresentação do conhecimento: a etapa prevê técnicas de visualização e apresentação do conhecimento extraído.

O processo de descoberta de conhecimento em uma grande base de dados não é trivial, pois envolve uma seqüência de passos conforme descritos acima. A validade das informações extraídas é outro importante aspecto a ser considerado. A identificação de padrões potencialmente úteis nos dados também está relacionada a esta área de estudo. A mineração de dados utiliza algoritmos desenvolvidos nas áreas de pesquisa em IA, no que tange a área de estudo em aprendizagem de máquina. Possui também algoritmos na área de pesquisa em banco de dados, à medida que manipula dados volumosos, e por fim na área de estatística, no que diz respeito à validação e avaliação dos resultados obtidos após execução.

2.2.3 Processamento da Linguagem Natural - PLN

A linguagem natural (LN) constitui o primeiro meio com que os indivíduos se comunicam, questionam, expressam desejos, crenças e atitudes [CIM06]. De modo geral, a LN apresenta as seguintes entidades:

- Verbos: descrevem eventos, estados, ações, crenças ou atitudes em uma sentença;
- Nomes próprios: representam nomes individuais;

- Nomes: representam classes, tipicamente pessoas, animais ou coisas;
- Adjetivos: representam o valor dos atributos das classes e modificam os nomes;
- Advérbios: representam a descrição da forma e modificam o verbo;
- Frases Preposicionais: representam a condição no tempo/espaço.

O PLN consiste na aplicação seqüencial da análise das diferentes atividades apresentadas abaixo [CIM06]:

- Pré-processamento: envolvem as etapas de *tokenização*, normalização, POS *(Part-of-Speech)*, *stemming*, reconhecimento de entidades nomeadas (reconhecimento de nomes de entidades) e resolução de co-referência (tipo simples de co-referência, relacionada a nomes de entidades). A *tokenização* consiste na detecção dos limites das palavras na sentença. A normalização consiste na transformação em um formato padrão. A POS é a tarefa de atribuir a cada *token* a correspondência de parte de um discurso. O *stemming* é aplicado em etapas de normalização e mapeamento de variações morfológicas.
- Análise sintática: compreende técnicas de agrupamento de palavras e dependência gramatical para aumentar o significado sintático.
- Análise semântica: consiste na análise do significado semântico.
- Interpretação contextual: consiste na interpretação do contexto, ou seja, o significado relacionado à intenção do texto.

O processo de extração de informações em bases textuais possui basicamente duas partes [ZAM02]: a etapa de extração de fatos individuais através da análise local do texto e a etapa de integração dos fatos, produzindo novos ou maiores fatos, através de inferência.

2.3 Evolução de Ontologia (EO)

Ao longo do tempo, são necessárias mudanças no domínio da aplicação ou nas necessidades do usuário do sistema, resultando na evolução da ontologia. Existe uma

crescente necessidade de definição de um método de manutenção de ontologias, tornando-se um campo interessante a ser pesquisado. A Evolução da Ontologia (EO) pode ser definida como a adaptação temporal da ontologia de acordo com o surgimento das mudanças, propagando as alterações de modo consistente a todos os artefatos dependentes [STO04]. Para que seja possível a evolução, é necessária a aprendizagem de novos conceitos ao longo do tempo.

A EO compreende nove atividades [ENK07]:

- Mapeamento de ontologias, morfismo, alinhamento, para permitir a interoperabilidade;
- Articulação, tradução, evolução, controle de versão, para garantir a coerência mediante as alterações da ontologia;
- Integração e fusão, para unificar os conceitos e relacionamentos da ontologia original com a finalidade de compreender as necessidades do domínio.

A evolução da ontologia consiste em sofrer modificações ao longo de um determinado período de tempo, visando à manutenção do conhecimento. Essas alterações podem ocasionar mudanças no vocabulário e terminologias ou em possíveis relações com outras ontologias. O processo de EO pode ocasionar problemas, como a geração de inconsistência entre os relacionamentos da ontologia. Para que um sistema de EO seja eficiente, é necessário definir um método para possibilitar ao usuário expressar seu pedido de mudança e o modo como a mudança solicitada pode ser realizada [STO03]. Com isto, a EO prevê um conjunto de atividades técnicas e gerenciais. Stojanovic [STO04] adapta da terminologia utilizada na área de banco de dados dois conceitos importantes:

- Sistema de gestão de ontologia: envolve um conjunto de métodos e técnicas necessárias para criar, modificar, definir versões, consultar e armazenar ontologias.
- Modificação da ontologia: consiste em alterar a ontologia sem levar em consideração as consistências necessárias.

Alguns interesses devem ser preservados durante a EO. Chen [CHE07] descreve a EO com as finalidades listadas abaixo:

- Identificar as mudanças de conceitos, papéis e indivíduos ao longo do tempo;
- Definir novos conceitos e relacionamentos e a compatibilidade com a ontologia atualizada;
- Descartar conceitos antigos e avaliar potenciais impactos nas ontologias e nas aplicações envolvidas.

De acordo com Stojanovic [STO04] o processo de EO prevê seis fases: alteração de captura da informação, alteração de representação, alteração da semântica, alteração de implementação, alteração de propagação e alteração da validação.

O controle de versões durante o processo de EO é um importante método de acompanhamento da ontologia. A identificação e o controle das novas versões podem ser realizados através das técnicas de detecção da diferença estrutural, conceitual ou do conjunto de transformações.

2.3.1 Gestão da EO

O ambiente empresarial evolui de modo dinâmico e contínuo. As modificações estão relacionadas à necessidade de alterações do domínio da aplicação e na adição de funcionalidades ao aplicativo. Stojanovic [STO04] apresenta três razões básicas para modificar um sistema:

- O ambiente de domínio do sistema pode sofrer modificações e invalidar o conhecimento em determinadas áreas ou necessitar adicionar funcionalidades ao passar dos anos.
- Os processos internos sofrem reengenharia constante com a finalidade de obter melhor desempenho, exigindo adaptações do sistema e conseqüentemente da base de ontologia.
- Os usuários do sistema solicitam modificações para corresponder as suas exigências, justificando a adaptação do sistema à atualidade. Eles podem ser classificados como usuários finais, que utilizam aplicativos desenvolvidos com uma base ontológica, ou engenheiros de ontologia, que são responsáveis pelo desenvolvimento e alteração da ontologia.

O desenvolvimento de uma ontologia pode ser considerado um processo dinâmico, pois começa a partir de uma ontologia inicial bruta, e posteriormente deve ser revista e refinada em seus detalhes, gerando uma ontologia final que satisfaça as necessidades previstas. Acima, estão descritas as alterações que ocorrerão possivelmente em um sistema, e para que uma ontologia possa servir de base para este sistema, o conhecimento que existe mapeado nela deverá ser adaptado (aprendendo novos conceitos) e evoluir de acordo com o sistema, para não tornar-se obsoleta.

Problemas poderão advir da EO, em conseqüência das alterações, tornando importante a definição de uma metodologia e ferramentas adequadas para auxiliar na realização desta tarefa. Stojanovic [STO04] identifica dois desafios a serem compreendidos em uma ferramenta que auxilie na execução automática da EO:

- Complexidade: uma ontologia pode possuir relacionamentos, o que pode significar em modificações cumulativas.
- Dependência: uma ontologia pode possuir outras ontologias de base, o que significa que a alteração de uma ontologia pode resultar na necessidade de modificação das suas ontologias base.

2.4 Rational Unified Process - RUP

Um processo pode ser definido como um conjunto de passos ordenados com intuito de atingir uma meta. O RUP busca a utilização de algumas das melhores práticas de desenvolvimento, atribuindo tarefas e responsabilidades em uma empresa de desenvolvimento de software, visando à produção de software com qualidade [BOO06].

O RUP está definido como um processo iterativo, enfatizando a construção de modelos definidos na UML, e disponibilizando a representação semântica do software que está sendo desenvolvido. Ele é composto por ciclos, onde a conclusão de cada ciclo é realizada na passagem pelas fases e resulta em uma versão do produto. Cada fase no ciclo de vida de desenvolvimento do software é o período de tempo entre dois marcos de progresso do processo. Em uma fase ocorrem várias e sucessivas iterações que passam pelo fluxo do processo. Cada iteração corresponde a um ciclo completo de desenvolvimento. A Figura 4 apresenta o ciclo de vida de desenvolvimento de software.

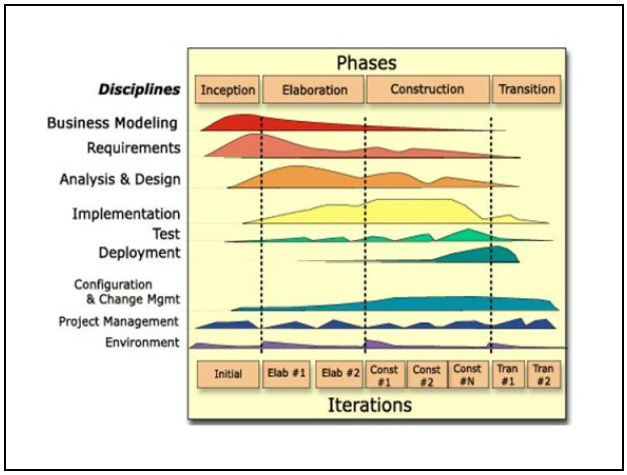

Figura 4 - Ciclo de vida do desenvolvimento de software [JAC99].

Durante a fase de concepção, o foco está na captação de requisitos de alto nível, na compreensão do problema e no plano do projeto inicial. Durante a fase de elaboração, o foco está na análise do domínio do problema, definição da arquitetura do sistema e no desenvolvimento do plano do projeto. Durante a fase de construção, o foco é a implementação do sistema e no teste do software. Já a última fase é a transição, que se caracteriza pela entrega do produto ao usuário.

O RUP define nove disciplinas. Durante a modelagem de negócio, a estrutura e a dinâmica da empresa são identificadas. O fluxo de trabalho de requisitos descreve os requisitos, identificando as necessidades de usuários e clientes. A análise e projeto refinam os requisitos do fluxo de trabalho anterior, contribuindo para a definição de várias visões da arquitetura.

A disciplina de implementação é responsável por planejar as integrações do sistema, implementar os subsistemas, testar as implementações e integrá-las. O teste define os casos de testes, procedimentos e medidas para acompanhamento dos erros. Nesta disciplina, os testes são realizados exaustivamente e os resultados obtidos são analisados em cada teste executado, em comparação aos requisitos definidos por clientes e usuários. A disciplina de entrega compreende o treinamento do usuário final, notas de versão ou outras necessidades relacionadas à entrega do produto. O gerenciamento da configuração é responsável pelo controle das modificações e manutenção da integridade dos artefatos do projeto. A disciplina de gerenciamento de projeto descreve as estratégias para o trabalho de modo iterativo. E por último, a disciplina denominada ambiente compreende a infra-estrutura necessária para tornar possível o desenvolvimento do sistema.

A abordagem descrita neste trabalho propõe a criação da ontologia inicial a partir do modelo de domínio representado através do diagrama de classes. Posteriormente, propõe o refinamento da ontologia inicial realizado de modo semi-automatizado através da leitura e interpretação dos detalhamentos de casos de uso que descrevem os requisitos do sistema, presentes na disciplina de requisitos do RUP. Portanto, a abordagem desenvolvida propõe a utilização da ontologia como forma de manter e formalizar o conhecimento, viabilizando a sua criação e refinamento de forma semi-automática, utilizando-se do diagrama de classes e descrições detalhadas de casos de uso, artefatos gerados no RUP.

2.5 Tecnologias Utilizadas

Para implementação da ferramenta de apoio a abordagem proposta neste trabalho, foi necessária a utilização de tecnologias existentes para viabilizar a geração e refinamento da ontologia durante o desenvolvimento de software. Tendo visto o estado da arte em IA, no que diz respeito aos principais conceitos que tangem o Processamento da Linguagem Natural, foi pesquisado um *parser* e algoritmos para auxiliar nas atividades propostas na abordagem. Abaixo está detalhado o analisador utilizado, e outras tecnologias necessárias para a implementação da abordagem proposta neste trabalho.

2.5.1 Stanford Parser

Um *parser* de linguagem natural é um programa analisador de linguagem natural, responsável por analisar a estrutura gramatical das sentenças organizadas em orações [NAT10].

O *Stanford parser* [NAT10] é um pacote implementado em linguagem de programação *Java* disponível atualmente para a versão JDK 1.5 ou posterior. O *parser* foi desenvolvido para a análise de dependências lexicais de frases e fornece como saída as relações gramaticais estruturadas em forma de árvore.

O *Stanford POS-Tagger* apresenta a proposta de retornar as dependências gramaticais através da representação de uma rede de dependência, apresentando aproximadamente 97,24% de *acurácia* sobre a classificação do texto marcado [TOU10].

O analisador *Stanford* permite a entrada de textos simples, fornecendo como saída a marcação de partes do discurso, uma estrutura de árvore ou as relações de

dependência gramatical. Para a frase "*I am testing the program.*", a saída gerada pelo *parser* será um texto marcado conforme apresenta a Figura 5.

```
I_PRP am_VBP testing_VBG the_DT program_NN ._.
```

Figura 5 - Exemplo de texto analisado pelo *Stanford POS-Tagger*.

O retorno do Stanford *POS-Tagger* à análise do texto é apresentado com algumas siglas, conforme descrição abaixo:

- ***_PRP***, sigla que representa o pronome pessoal (*personal pronoun*);
- ***_VBP***, sigla que representa o verbo conjugado no tempo presente (*verb, present*);
- ***_VBG***, sigla que representa o verbo no gerúndio (*verb, gerund*);
- ***_DT***, sigla que representa um determinante (*determiner*);
- ***_NN***, sigla que representa um nome (*noun, singular*);
- ***_.***, representa o ponto-final da sentença (*sentence-final punctuation*).

Outras formas de abreviação serão apresentadas pelo *parser* de acordo com o texto analisado. Adjetivos, preposições, advérbios, entre outros também são identificados pelo *Stanford POS-Tagger* e representados através de siglas que estão descritas conforme artigo e documentação disponível na ferramenta [NAT10] [NAT10a] [MAR08].

O *Stanford parser* realiza a análise das sentenças e apresenta uma estrutura de árvore conforme apresenta a Figura 6.

```
(ROOT
  (S
    (NP (PRP I))
    (VP (VBP am)
      (VP (VBG testing)
        (NP (DT the) (NN program))))
    (. .)))
```

Figura 6 - Saída da sentença de exemplo analisada pelo *Stanford parser* em estrutura de árvore.

A análise das dependências gramaticais é apresentada conforme ilustra a Figura 7.

```
nsubj(testing-3, I-1)aux(testing-3, am-2)
det(program-5, the-4)dobj(testing-3,
program-5)
```

Figura 7 - Exemplo de texto analisado pelo *Stanford parser* retornando as estruturas de dependência gramatical.

Para a análise da sentença exemplo, a representação das dependências gramaticais está apresentada conforme segue:

- ***nsubj***, representa o sujeito sintático da frase (*nominal subject*);
- ***aux***, representa o verbo auxiliar da frase (*auxiliary*);
- ***dobj***, representa o objeto direto da frase (*direct object*).

De acordo com a análise da sentença exemplo, tem-se a estrutura identificada pelo *Stanford parser*:

- ***I***, sujeito da sentença (quem realiza a ação);
- ***am***, verbo da sentença;
- ***program***, objeto direto – completa o verbo (quem testa, testa alguma coisa).

2.5.2 WordNet

A *WordNet* é um grande banco de dados lexicais para o idioma inglês, criado pelo Departamento de Ciência da Computação da Universidade de Princeton [WOR10].

A *WordNet* distingue substantivos, verbos, adjetivos e advérbios por possuírem regras gramaticais distintas, e estabelece as relações léxico-semânticas e semântico-conceituais entre as palavras. As palavras são agrupadas em conjuntos de sinônimos denominados *synsets*, cada um representando um conceito distinto. Os *synsets* podem conter ou um grupo de palavras sinônimas ou seqüências de palavras que co-ocorrem e formam um significado específico. Um exemplo de saída da *WordNet* para a palavra “*sale*” é apresentado na Figura 8.

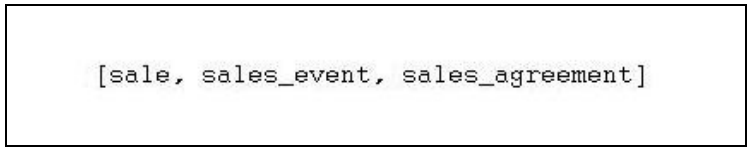

Figura 8 - Exemplo de saída da *WordNet* para a palavra *sale*.

Uma palavra pode ter mais de um sentido, sendo denominada de polissêmica ou compartilhar o mesmo sentido comum com outra palavra, sendo um sinônimo. As variações nas relações semânticas estão incorporadas na *WordNet* e estão descritas como ponteiros entre as palavras e os sentidos da palavra. Abaixo estão listadas as relações semânticas [MIL95]:

- **Sinonímia:** relação de base da *WordNet*, uma vez que está organizada em conjuntos de sinônimos (*synsets*) para representar o sentido das palavras;
- **Antonímia:** representa o sentido oposto das palavras e é importante na organização de adjetivos e advérbios;
- **Hiperonímia:** palavras que representam o sentido de um todo (significado abrangente) – super-nome;
- **Hiponímia:** é oposto da hiperonímia e representa o sentido de parte ou item de um todo (significado mais específico) – sub-nome. As relações de hiperonímia e hiponímia possibilitam a organização hierárquica dos significados dos substantivos, pois geralmente para cada hipônimo, existe apenas um hiperônimo;
- **Holonímia:** representa a relação de inclusão entre duas palavras, onde denota o todo - nome completo;
- **Meronímia:** é o oposto de holonímia e representa a relação de parte de um todo – parte do nome. As relações de holonímia e meronímia se distinguem das relações de hiperonímia e hiponímia por não haver transferência de propriedades semânticas, não estabelecendo o significado de “é um subtipo de” (hierarquia);
- **Troponímia:** representa a relação de sentido de parte ou item de um todo (significado mais específico) para verbos - nome da forma;

- **Acarretamento:** representa a relação de acarretamento entre verbos, onde a verdade de uma sentença implica na verdade de outra.

Para que as frases tenham sentido, elas são compostas de palavras significativas [MIL95]. Na identificação do sentido das palavras, a *WordNet* observa as categorias sintáticas (substantivo, adjetivo, advérbio e verbo) e as relações semânticas, estabelecidas de acordo com o contexto que estão inseridas.

A utilização da *WordNet* neste projeto de pesquisa é importante durante as fases de geração e refinamento da ontologia inicial, auxiliando na identificação de conceitos a partir de termos extraído dos documentos do processo de desenvolvimento de software. De acordo com Guarino [GUA97] a utilização da *WordNet* é de grande ajuda para a construção de ontologias, por permitir a generalidade, identificar ambigüidades e diferenças sutis, e garantir a legibilidade.

3. TRABALHOS RELACIONADOS

Na literatura, são encontrados trabalhos que sugerem a utilização da ontologia para o apoio às atividades de desenvolvimento de software. As ontologias desenvolvidas no domínio de ES geralmente são aplicadas com a finalidade de solucionar problemas decorrentes da existência de ambigüidade durante o processo de desenvolvimento de software.

A proposta desenvolvida em [TAL08] utiliza uma ontologia de componentes de software. O autor relata a necessidade de modificação dos dados do projeto periodicamente, com a finalidade de representar o progresso e alterações no desenvolvimento do projeto, mudança nos requisitos, adição de funcionalidades, melhorias incrementais e re-configuração. A ontologia é gerada com a finalidade de solucionar o problema da dificuldade de compreensão dos serviços de um componente, sua utilização e suas pré-condições e pós-condições operacionais. O autor prevê a criação de uma ontologia que envolve o conhecimento genérico de componentes de software em ES e a modelagem de sub-ontologias que expressam as necessidades específicas do projeto. Para criação da ontologia, a abordagem sugere o *framework Jena* [JEN10] e o formato OWL para representar o conhecimento.

O trabalho de Wongthongthaml *et al.* [WON07] apresenta uma comparação entre a modelagem de ontologias e o modelo de orientação a objetos (UML). A proposta retrata a transformação de um conjunto de conceitos de ES em uma ontologia no domínio de ES. Em cada projeto, haverá dados particulares modelados na ontologia. Toda a equipe de desenvolvimento pode, através da ontologia, consultar a semântica dos dados do projeto, e através deste conhecimento modelado, levantar questões de debate, analisar problemas, propor revisões ou novas soluções de software. O autor propõe a geração de uma ontologia através da similaridade do conceito de classes e suas relações existentes em diagramas de classes. A abordagem apresenta notações gráficas para representar uma ontologia.

Em outro trabalho publicado por Wongthongthaml *et al.* [WON06], a utilização de ontologias é apresentada com a finalidade de capturar e organizar o conhecimento de um determinado domínio, e facilitar a comunicação em um projeto de ES. As informações mapeadas em conceitos e relacionamentos da ontologia restringem as informações, possibilitando a utilização e manutenção do conhecimento sobre o projeto. O autor propõe novamente uma ontologia de ES genérica e uma metodologia de criação de instâncias

com informações específicas do projeto a partir de documentos UML, mas não apresenta um software para automação do processo. Em [WON06] é sugerido como trabalho futuro um processo para obtenção de conhecimento e viabilização da criação de instâncias da ontologia automaticamente, a partir dos dados contidos nos documentos de software.

O trabalho apresentado por Nicola *et al.* [NIC09] propõe uma metodologia para construção de ontologias derivada do Processo Unificado (UP), amplamente aceito na comunidade de ES. O autor denominou a metodologia de UPON e justifica a necessidade da utilização de uma metodologia para geração de ontologias para auxiliar a tarefa de modelagem das ontologias executada pelos engenheiros de ontologia. O trabalho apresenta a metodologia de construção de ontologias seguindo os princípios subjacentes e as fases básicas do UP. O UPON apresenta os objetivos de reduzir o tempo e o custo da construção de ontologias, melhorar a qualidade, e disponibilizar a busca semântica das informações para os usuários da ontologia. A captura dos requisitos na metodologia UPON consiste na especificação das necessidades semânticas modeladas na ontologia de acordo com a visão do usuário. A atividade necessita da interação dos analistas (que vão efetuar a modelagem do sistema), dos engenheiros do conhecimento e dos usuários da aplicação. Nicola *et al.* [NIC09] sugerem as diretrizes a seguir (tarefas dos engenheiros de conhecimento e especialistas de domínio):

1. Determinar o domínio de interesse e as possibilidades de alcance.
2. Definir o objetivo do negócio. Determinar a finalidade comercial, ou definir cenários com usuários e objetivos e especificar a razão da existência de uma ontologia. O especialista de domínio define a seqüência de atividades no cenário especificado.
3. O especialista de domínio especifica uma terminologia específica da aplicação.
4. Determinar questões conceituais que deverão estar compreendidas na modelagem da ontologia.
5. Modelar os casos de uso relacionados às questões conceituais estabelecidas na tarefa 4.

O engenheiro de ontologias tem como principal contribuição para a metodologia a modelagem conceitual. A análise conceitual consiste no refinamento da ontologia gerada a partir do cumprimento das tarefas anteriormente descritas.

Nicola *et al.* [NIC09] relatam ainda que, embora existam métodos de aprendizagem de ontologias, ainda persiste a necessidade de validação manual da ontologia construída, o que gera um esforço significativo. Neste sentido, a metodologia proposta por Nicola *et al.* apóia o engenheiro de ontologias nesta tarefa. O trabalho sugere como posterior etapa a automação da metodologia.

3.1 Considerações sobre os trabalhos relacionados

Neste Capítulo estão descritos quatro trabalhos que utilizam ontologias em ES. O primeiro trabalho [TAL08] necessita da utilização de uma ontologia inicial genérica de componente de software para após instanciar sub-ontologias que modelam o domínio específico. Em [WON06] e [WON07] os autores propõe uma ontologia que modela o domínio de ES e apresentam uma metodologia para instanciação dos conceitos relativos ao projeto a partir de documentos UML. Esta é a grande diferença destes trabalhos com a abordagem proposta nesta dissertação, pois de acordo com os autores, uma ontologia genérica inicial sempre é necessária, enquanto que na abordagem deste trabalho (apoiada pela ferramenta *OntSoft*) a ontologia inicial é gerada, refinada e enriquecida durante a engenharia de requisitos no processo de desenvolvimento de software.

A abordagem apresentada em [NIC09] sugere um processo para construção de ontologias com base no UP. A grande diferença do trabalho de Nicola *et al.* [NIC09] para a abordagem desenvolvida e descrita nesta dissertação é que o foco deste trabalho está na automação do processo de construção de ontologias inserido no contexto do RUP de desenvolvimento de software. Em contrapartida, Nicola *et al.* [NIC09] apresentam um processo para construção de ontologias, visando o apoio aos engenheiros de ontologias na tarefa de acelerar a consolidação e avaliação da ontologia gerada automaticamente ou auxiliar a construção de ontologias de domínio úteis e eficazes modeladas manualmente através da utilização de uma metodologia.

Embora existam algumas semelhanças entre os trabalhos apresentados neste Capítulo e a abordagem proposta nesta dissertação, os trabalhos não estão inseridos exatamente no foco desta dissertação. Apesar disto, um acordo entre as abordagens apresentadas nos trabalhos [TAL08] [NIC09] e a abordagem definida nesta dissertação é a utilização da linguagem OWL, recomendada pela W3C, para descrição da ontologia construída.

4. DEFINIÇÃO DA ABORDAGEM

O objetivo da abordagem apresentada neste trabalho é possibilitar a representação do conhecimento contido no modelo de domínio e nos detalhamentos de casos de uso de software, através de ontologias. Para isto, está previsto a captura e codificação do conhecimento de forma semi-automatizada, através da identificação de conceitos e relacionamentos, facilitando o trabalho de criação e refinamento da ontologia. A utilização de ontologias no RUP de desenvolvimento de software pode trazer muitos benefícios, por proporcionar a formalização do conhecimento, possibilitando o melhor entendimento dos conceitos relacionados à solução de software no decorrer do seu desenvolvimento.

De acordo com [NOL07a], as ontologias podem auxiliar o desenvolvimento de software. Porém, o processo de geração e manutenção de ontologias não é trivial [MAE03]. A criação de uma ontologia envolve o profundo conhecimento do domínio que está sendo modelado.

A abordagem deste trabalho propõe a aplicação de técnicas de IA para viabilizar a criação e aprendizagem da ontologia. A ontologia é gerada a partir do diagrama de classes que descreve o modelo de domínio e, por fim, enriquecida e refinada a partir das descrições detalhadas de casos de uso, possibilitando com isto a representação do conhecimento, e permitindo a identificação dos artefatos relacionados semanticamente. A criação desta abordagem visa viabilizar o processo oneroso de criação da ontologia, e permitir o refinamento da ontologia proposta, representando com menor granularidade os conceitos relevantes do domínio da aplicação de software em desenvolvimento. A Figura 9 apresenta a abordagem dividida em duas fases.

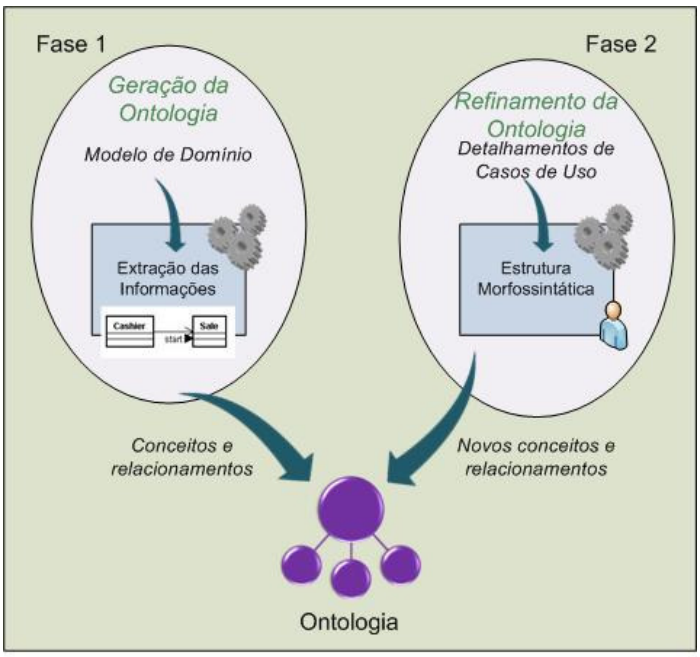

Figura 9 - Abordagem para geração e refinamento da ontologia.

A primeira fase corresponde à geração da ontologia inicial e a segunda fase refere-se ao refinamento e enriquecimento da ontologia inicial. As fases da abordagem estão detalhadas a seguir.

4.1 Criação da Ontologia Inicial

Para a criação da ontologia inicial, está previsto o mapeamento das informações contidas no modelo de domínio para uma ontologia. O modelo de domínio está descrito como um diagrama de classes definido na UML (*Unified Modeling Language*) que modela a estrutura geral do problema. O modelo de domínio retrata o cenário geral do sistema, visando à descrição do problema que o software pretende solucionar.

Para entendimento desta abordagem, a Figura 10 apresenta o diagrama de classes que define o modelo de domínio da aplicação *ProxGer* [LAR07] apresentada como exemplo de uso no Capítulo 6 desta dissertação.

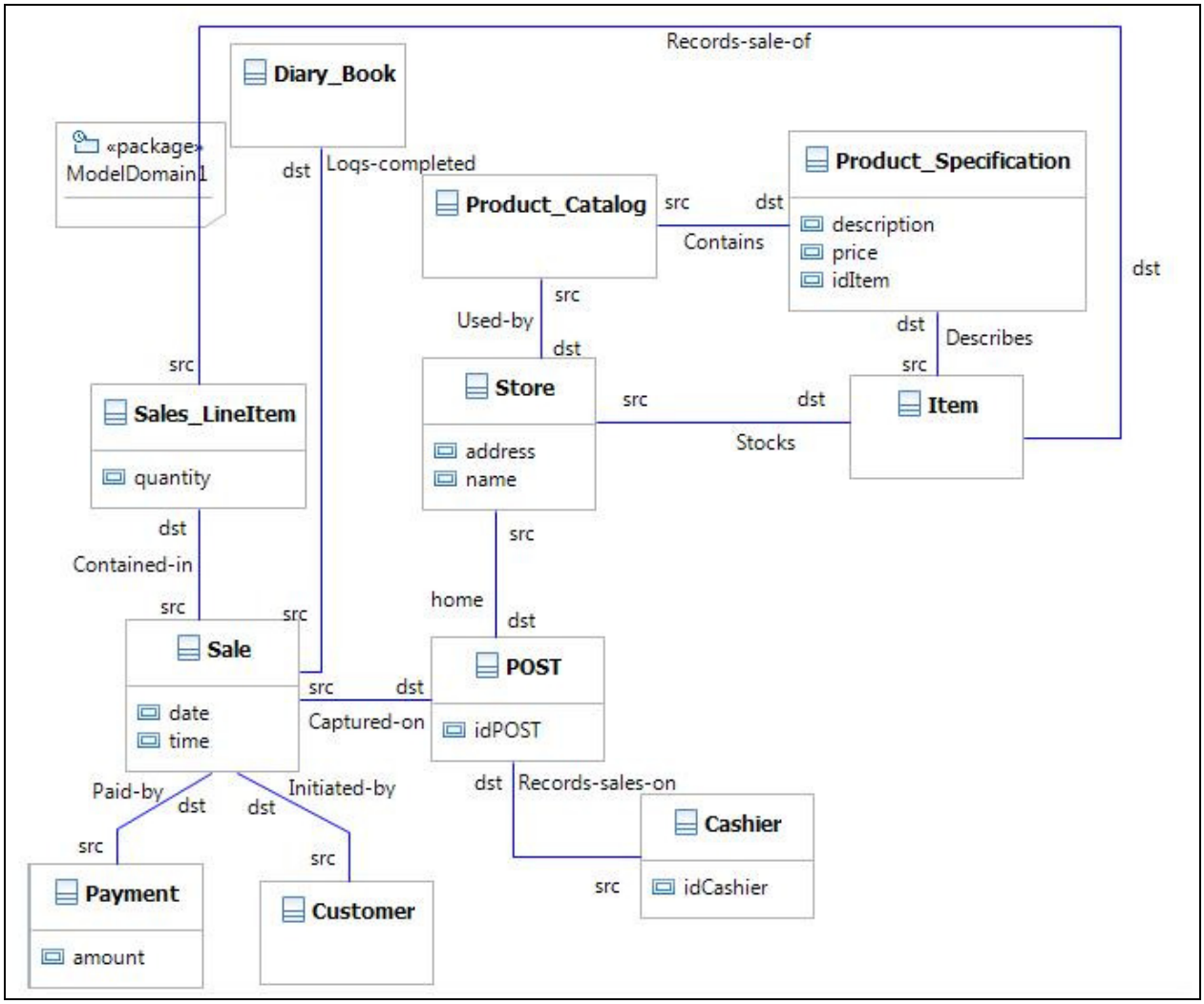

Figura 10 - Modelo de domínio da aplicação *ProxGer* [LAR07].

A partir do diagrama de classes é possível modelar a ontologia inicial da aplicação. Um diagrama de classes da UML pode ser representado por um arquivo XMI e interpretado através de um *parser* desenvolvido para interpretação de arquivos XML. Um trecho do arquivo XMI que representa o diagrama de classes da Figura 10 é apresentado na Figura 11.

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package xmi:version="2.1"
xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:uml="http://www.eclipse.org/uml2/3.0.0/UML" xmi:id="_XrjFwb7lEd
-9QYDMueSHEw" name="ModelDomain1">
  <elementImport xmi:id="_XxWy0L7lEd-9QYDMueSHEw">
    <importedElement xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#Boolean"/>
  </elementImport>
  <elementImport xmi:id="_XxqU0L7lEd-9QYDMueSHEw">
    <importedElement xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
  </elementImport>
  <elementImport xmi:id="_XxqU0b7lEd-9QYDMueSHEw">
    <importedElement xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#UnlimitedN
atural"/>
  </elementImport>
  <elementImport xmi:id="_XxqU0r7lEd-9QYDMueSHEw">
    <importedElement xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#Integer"/>
  </elementImport>
  <packagedElement xmi:type="uml:Class" xmi:id="_b1rj4L7lEd-
9QYDMueSHEw" name="Product_Catalog"/>
  <packagedElement xmi:type="uml:Class" xmi:id="_c1Kq8L7lEd-
9QYDMueSHEw" name="Product_Specification">
    <ownedAttribute xmi:id="_ccZeQL7oEd-9QYDMueSHEw"
name="description" aggregation="composite"/>
    <ownedAttribute xmi:id="_fxyFsL7oEd-9QYDMueSHEw" name="price"
aggregation="composite"/>
    <ownedAttribute xmi:id="_g2Bw8L7oEd-9QYDMueSHEw" name="idItem"
aggregation="composite"/>
  </packagedElement>
  <packagedElement xmi:type="uml:Class" xmi:id="_d6c4AL7lEd-
```

Figura 11 - Trecho do arquivo XMI referente ao diagrama de classes.

A extração da ontologia inicial parte da simplificação da definição formal da ontologia {*C, R, A, T*} com a estrutura do diagrama de classes de um modelo de domínio, que define classes, relacionamentos e atributos. De acordo com o modelo de domínio apresentado na Figura 6, podem-se extrair os seguintes conjuntos:

- *C = { Diary_Book, Product_Catalog, Product_Specification, Sale_Line_Item, Store, Item, Sale, POST, Payment,Customer, Cashier };*
- *R = { Records-sale-of, Contains, Describes, Logs-completed, Used-by, Stocks, Contained-in, home, Captured-on, Paid-by, Initiated-by, Records-sale-on };*
- *A = { quantity, description, price, idItem, address, name, date, time, idPOST, amount, idCashier };*

Considerando ainda o sistema *ProxGer* [LAR07], é possível identificar os seguintes relacionamentos (notação σ *R* (relação) = {domínio, *range*} [CIM06]):

- *σ R (Records-sale-of) = (Sale_Line_Item, Item);*
- *σ R (Contains) = (Product_Catalog, Product_Specification);*
- *σ R (Describes) = (Item, Product_Specification);*
- *σ R (Logs-completed) = (Sale, Diary_Book);*
- *σ R (Used-by) = (Product_Catalog, Store);*
- *σ R (Stocks) = (Store, Item);*
- *σ R (Contained-in) = (Sale, Sale_Line_Item);*
- *σ R (home) = (Store, POST);*
- *σ R (Captured-on) = (Sale, POST);*
- *σ R (Payd-by) = (Payment, Sale);*
- *σ R (Initiated-by) = (Customer, Sale);*
- *σ R (Records-sale-on) = (Cashier, POST).*

Através da extração destas informações, a ontologia inicial é gerada. Para criação e alteração da ontologia, é utilizado o *framework Jena*, desenvolvido em linguagem de programação *Java*, por disponibilizar um ambiente de programação e um mecanismo de inferência baseado em regras [JEN10]. A linguagem utilizada para geração da ontologia inicial é o OWL (indicação da W3C conforme apresenta o Capítulo 2).

A Figura 12 apresenta as etapas da geração da ontologia inicial.

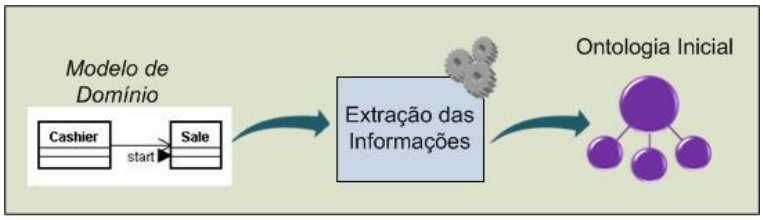

Figura 12 - Geração da ontologia inicial.

A modelagem de uma ontologia a partir do modelo de domínio descrito em um diagrama de classes está previsto no trabalho desenvolvido em Noll [NOL07] e serve de insumo para esta pesquisa.

4.2 Refinamento da Ontologia

O refinamento da ontologia está previsto na abordagem como uma seqüência de passos para extração de informações relevantes identificadas a partir dos documentos que descrevem o modelo de domínio da aplicação e documentos que descrevem os detalhamentos de casos de uso gerados durante a disciplina de requisitos descrita no RUP.

Durante a disciplina de requisitos, são expressos os requisitos funcionais (necessidades de usuários e clientes) através do modelo de casos de uso [JAC99]. Um caso de uso descreve o comportamento do sistema, podendo conter diferentes sequências de comportamento [COC01]. Na UML, um caso de uso está definido como um conjunto de seqüências de ações executadas pelo sistema com a finalidade de produzir um resultado observável por um ator [BOO06].

A fase de concepção do desenvolvimento de software inclui a identificação dos casos de uso mais importantes, delimitando o domínio do sistema. Na fase de elaboração, os casos de uso são especificados com melhor detalhamento. Durante a fase de construção, as iterações estão voltadas para a construção do software, e para isto são identificados os requisitos restantes que devem ser implementados. A última fase do ciclo de vida de desenvolvimento inclui a captura dos requisitos, caso haja alguma alteração ou implementação adicional do sistema.

A forma mais simples de descrição de um caso de uso envolve a identificação dos tipos de interação e os atores envolvidos [SOM08]. Um detalhamento de caso de uso aceito pelo RUP possui os seguintes campos:

- Identificação: nome do caso de uso;
- Atores: nome dos atores. Um ator representa um conjunto de papéis que desempenha ao interagir com o caso de uso;
- Pré-condições: definição das condições prévias para a execução do caso de uso (opcional);
- Pós-Condições: definição das condições posteriores à execução do caso de uso (opcional);
- Fluxo Básico: seqüência típica de eventos definida na ação do ator e resposta do sistema;

- Fluxo Alternativo: seqüência alternativa em resposta à ação de um ator. Pode não existir ocorrências de fluxo alternativo em um determinado caso de uso.

O campo *stakeholders* também pode aparecer no detalhamento de caso de uso. Um sistema pode conter diversos *stakeholders* que descrevem necessidades diferentes. A Tabela 1 apresenta o detalhamento de caso de uso adaptado de [LAR07] utilizado como exemplo de uso.

Tabela 1 - Exemplo de detalhamento de caso de uso adaptado de [LAR07].

Use Case UC1:	**Process Sale**
Primary Actor:	Cashier
Stakeholders and Interests:	Cashier: Wants accurate, fast entry, and no payment errors, as cash drawer short ages are deducted from his/her salary. Salesperson: Wants sales commissions updated. Customer: Wants purchase and fast service with minimal effort. Wants easily visible display of entered items and prices. Wants proof of purchase to support returns. Company: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server components (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory. Manager: Wants to be able to quickly perform override operations, and easily debug Cashier problems. Government Tax Agencies: Want to collect tax from every sale. May be multiple agencies, such as national, state, and county. Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.
Preconditions:	Cashier is identified and authenticated.
Post conditions:	Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated. Payment authorization approvals are recorded.
Basic Flow:	1. Customer arrives at POS checkout with goods and/or services to purchase. 2. Cashier starts a new sale. 3. Cashier enters item identifier. 4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules. Cashier repeats steps 3-4 until indicates done. 5. System presents total with taxes calculated. 6. Cashier tells Customer the total, and asks for payment. 7. Customer pays and System handles payment. 8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory). 9. System presents receipt. 10. Customer leaves with receipt and goods (if any).
Extensions or Alternative Flows:	*a. At any time, Manager requests an override operation: 1. System enters Manager-authorized mode. 2. Manager or Cashier performs one Manager-mode operation. e.g., cash balance change, resume a suspended sale on another register, void a sale, etc. 3. System reverts to Cashier-authorized mode.

Observando a descrição de um caso de uso qualquer, é possível identificar certa estrutura na sua descrição. Ao contrário de um texto qualquer, o detalhamento segue uma estrutura conforme ilustrou a Tabela 1.

Os casos de uso tornaram-se fundamentais para a descrição de modelos de sistemas [SOM08]. O detalhamento de caso de uso traz informações importantes, como os requisitos essenciais que o software em desenvolvimento deverá contemplar. Estas informações estão descritas em Linguagem Natural (LN), o que pode resultar em ambigüidade de interpretação, característica da linguagem livre. Para possibilitar a extração das informações relevantes descritas em LN, a abordagem está dividida em etapas.

A primeira etapa do processo de refinamento da ontologia consiste em identificar as informações relevantes contidas na descrição do caso de uso. Estas informações, posteriormente, poderão ser categorizadas em conceitos e relacionamentos. Para isto, estão previstas as seguintes atividades:

1. Leitura individual de cada frase;
2. Identificação dos *tokens*. As atividades de *tokenização*, normalização e *stemming* estão previstas nas atividades do PLN e contempladas na utilização do *Stanford POS-tagger* [NAT10a] na ferramenta desenvolvida para apoio a esta abordagem (conforme apresenta o Capítulo 5).
3. Identificação da presença do caractere vírgula na frase e verificação da ocorrência de período composto.
4. Transformação dos períodos compostos em períodos simples. Para que isto seja possível, a identificação do sujeito da oração muitas vezes se faz necessária. Para identificar o sujeito, a ferramenta de apoio à abordagem *OntSoft* utiliza o *Stanford parser* [NAT10] (conforme apresenta o Capítulo 5).
5. Identificação das regras para interpretação das frases com anotações lingüísticas.
6. Descoberta das tuplas de enriquecimento e refinamento da ontologia. Para gerar as tuplas, algumas regras implementam novamente o *parser* para identificar o sujeito, ou utilizam da base de dados lexical

disponível através do *WordNet* [WOR10]. Maiores detalhes sobre a criação das regras são descritos na seção 4.2.1.

Sendo assim, ao executar o *Stanford POS-Tagger* no caso de uso de exemplo acima, tem-se a seguinte saída conforme apresenta a Figura 13.

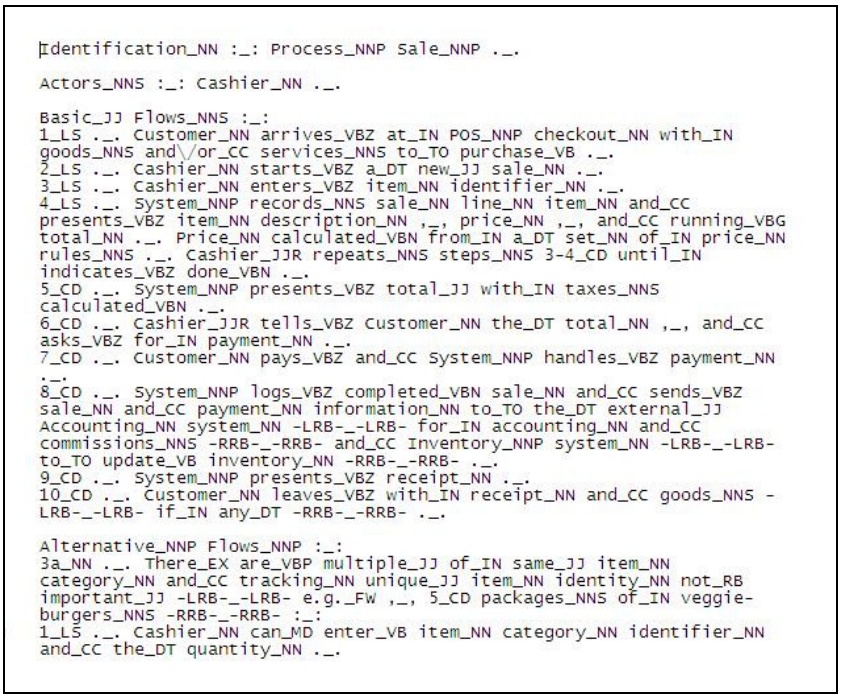

Figura 13 - Retorno da aplicação do *Stanford POS-Tagger* no trecho do detalhamento de Caso de Uso.

O *POS-tagging* é responsável pela categorização sintática das palavras de um texto. A análise da frase através da utilização do *Stanford parser* permite a identificação do sujeito da oração. Sendo assim, a aplicação do *parser* no trecho do detalhamento de caso de uso apresentado na Tabela 1 resulta no arquivo conforme apresenta a Figura 14.

```
[nsubj(Sale-2, Process-1)]
[]
[nsubj(arrives-2, Customer-1), dep(checkout-5, POS-4), prep_at(arrives-2,
checkout-5), prep_with(checkout-5, goods-7), conj_and/or(goods-7, services-9),
aux(purchase-11, to-10), xcomp(arrives-2, purchase-11)]
[nsubj(starts-2, Cashier-1), det(sale-5, a-3), amod(sale-5, new-4), dobj(starts
-2, sale-5)]
[nsubj(enters-2, Cashier-1), nn(identifier-4, item-3), dobj(enters-2,
identifier-4)]
[nsubj(records-2, System-1), nn(item-5, sale-3), nn(item-5, line-4), dobj
(records-2, item-5), conj_and(records-2, presents-7), nn(description,-9, item-
8), dobj(presents-7, description,-9), xcomp(presents-7, price,-10), conj_and
(price,-10, running-12), dobj(price,-10, total-13)]
[nsubj(calculated-2, Price-1), det(set-5, a-4), prep_from(calculated-2, set-5),
nn(rules-8, price-7), prep_of(set-5, rules-8)]
[amod(steps-3, Cashier-1), nn(steps-3, repeats-2), nsubj(indicates-6, steps-3),
measure(until-5, 3-4-4), advmod(indicates-6, until-5), acomp(indicates-6, done-
7)]
[nsubj(presents-2, System-1), dobj(presents-2, total-3), prep_with(presents-2,
taxes-5), partmod(taxes-5, calculated-6)]
[nsubj(tells-2, Cashier-1), iobj(tells-2, Customer-3), det(total,-5, the-4),
dobj(tells-2, total,-5), conj_and(tells-2, asks-7), prep_for(asks-7, payment-9)]
[nsubj(pays-2, Customer-1), nsubj(handles-5, System-4), conj_and(pays-2,
handles-5), dobj(handles-5, payment-6)]
[nsubj(logs-2, System-1), amod(sale-4, completed-3), dobj(logs-2, sale-4),
conj_and(logs-2, sends-6), nn(information-10, sale-7), conj_and(sale-7, payment
-9), dobj(sends-6, information-10), det(accounting-17, the-12), amod(accounting
-17, external-13), nn(accounting-17, Accounting-14), nn(accounting-17, system-
15), nn(accounting-17, (for-16), prep_to(sends-6, accounting-17), conj_and
(accounting-17, commissions)-19), nn(inventory)-25, Inventory-21), nn
(inventory)-25, system-22), nn(inventory)-25, (to-23), nn(inventory)-25, update
-24), conj_and(accounting-17, inventory)-25)]
[nsubj(presents-2, System-1), dobj(presents-2, receipt-3)]
[nsubj(leaves-2, Customer-1), nn(any)-8, receipt-4), conj_and(receipt-4, goods-
6), nn(any)-8, (if-7), prep_with(leaves-2, any)-8)]
[expl(are-2, There-1), acomp(are-2, multiple-3), amod(category-7, same-5), nn
(category-7, item-6), prep_of(multiple-3, category-7), conj_and(multiple-3,
tracking-9), amod(identity-12, unique-10), nn(identity-12, item-11), dobj
(tracking-9, identity-12), prep(tracking-9, not-13), amod((e.g.-15, important-
14), dep(not-13, (e.g.-15)]
[num(packages-2, 5-1), nsubj(enter-7, packages-2), nn(Cashier-5, veggie-
burgers):-4), prep_of(packages-2, Cashier-5), aux(enter-7, can-6), nn
(identifier-10, item-8), nn(identifier-10, category-9), dobj(enter-7,
identifier-10), det(quantity-13, the-12), conj_and(identifier-10, quantity-13)]
```

Figura 14 - Retorno do *Stanford parser* aplicado ao trecho detalhamento de caso de uso apresentado na Tabela 1.

Assim, para o refinamento da ontologia inicial, duas importantes informações são obtidas a partir do detalhamento de caso de uso: identificação da categoria gramatical de cada palavra e reconhecimento do sujeito de cada frase.

A Figura 15 apresenta um trecho do detalhamento de caso de uso e as informações identificadas através do *POS-tagger e do parser Stanford*.

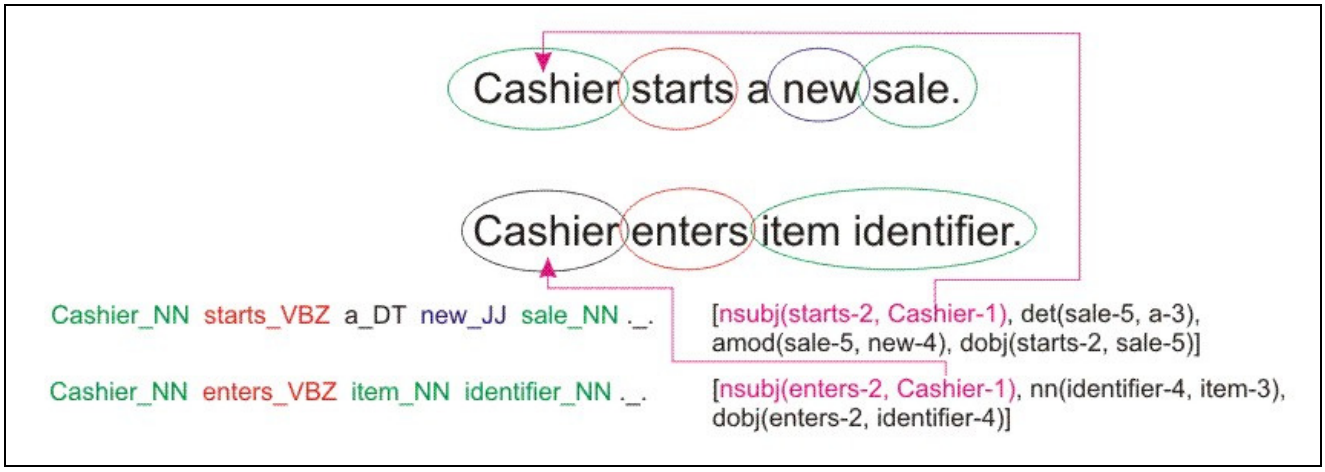

Figura 15 - Trecho do detalhamento do caso de uso e a extração das informações obtidas através do *POS-tagger* e *parser Stanford*.

A partir da identificação destas estruturas, é possível a indução de uma série de relacionamentos e conceitos candidatos a refinar a ontologia. Considerando o exemplo do trecho do detalhamento de caso de uso, é possível identificar alguns relacionamentos. Utilizando a notação σ *R* (relação) = {domínio, *range*} [CIM06]:

- *σ R (starts) = (cashier, sale);*
- σ R (enters) = (cashier, item).

De acordo com o exemplo, é possível observar os substantivos (NN) como prováveis conceitos, identificando domínio e *range* respectivamente, e o verbo (VB) aparecendo como a relação que os une, porque os verbos tendem as ser modelados como relações [MAE03]. A identificação dos conceitos e relacionamentos ocorre a partir do mapeamento de regras que prevêem a estrutura gramatical das frases e sugere tuplas para refinar e enriquecer a ontologia inicial. A seguir está detalhada a criação das regras utilizadas neste trabalho.

4.2.1 Definição das regras

Com a finalidade de possibilitar a geração das tuplas de refinamento da ontologia inicial, foi necessário estabelecer regras sobre as orações que descrevem os casos de uso. As regras têm a finalidade de identificar conceitos e relacionamentos que possibilitem o refinamento da ontologia inicial. Elas retornam tuplas descritas na estrutura – (*conceito_1, relacionamento, conceito_2)*, onde:

- O conceito_1 identificado sugere um possível conceito existente na ontologia inicial ou sugere a criação de um novo conceito na

ontologia, enriquecendo o conhecimento do domínio o qual representa.

- O relacionamento identificado sugere um relacionamento entre dois conceitos;
- O conceito_2 sugere um possível conceito anteriormente modelado na ontologia inicial (existente no modelo de domínio da aplicação) ou sugere a criação de um novo conceito na ontologia.

O detalhamento textual do caso de uso *process sale* [LAR07] foi utilizado como modelo para definição das regras iniciais da abordagem. O processo inicial de criação das regras foi realizado contemplando as seguintes etapas:

- Processamento do detalhamento de caso de uso pelo *POS-tagger Stanford*, retornando como saída o texto etiquetado;
- Processamento do detalhamento de caso de uso pelo *parser Stanford*, retornando o texto marcado em sujeito (dentre outras informações que não apresentam interesse no contexto desta abordagem);
- Identificação das estruturas nome + verbo + nome inicialmente;
- Considerações das estruturas gramaticais: adjetivo, advérbio, verbo modal e preposições que interferem na semântica dos substantivos e verbos;
- Tratamento dos sinais de pontuação: ocorrência de dois pontos e vírgula;
- Tratamento das conjunções coordenativas (CC) *and* e *or*;
- Identificação das regras que necessitam utilizar a informação de sujeito da oração informada através do *parser*;
- Identificação da estrutura gramatical nome + verbo na terceira pessoa do tempo verbal presente, e chamada da *WordNet* para substituição do verbo para o modo infinitivo;

- Generalização das regras através da definição de zero ou mais ocorrências das estruturas gramaticais: preposição (IN), determinante (DT), *to* (TO) e presença de números cardinais (CD).

A realização da identificação e formação das regras (expressões regulares) é responsável por gerar as tuplas de refinamento e enriquecimento da ontologia inicial. A geração das regras deste trabalho levou em consideração a estrutura gramatical da língua inglesa. A sintaxe é o estudo da combinação de palavras geridas por um conjunto de regras responsáveis pela construção de frases na linguagem natural. As orações em língua inglesa dividem-se em sujeito e predicado, e podem ser formadas por períodos simples e compostos. O sujeito pode ser um nome ou um pronome. Um predicado exprime a noção de estado ou movimento do sujeito.

Um período simples é encontrado no detalhamento de caso de uso do autor Larman [LAR07]:

"Cashier starts a new sale."

Onde a oração apresenta a estrutura:

- "*Cashier*": sujeito da oração.
- "*starts a new sale*": predicado da oração.

A frase é um período simples por possuir apenas um único sujeito e um único predicado. Ao unirmos mais de um predicado na oração, tem-se um período composto. Os períodos compostos são identificados através do símbolo de pontuação vírgula ou da conjunção *e* (classe gramatical). As orações abaixo extraídas do detalhamento de caso de uso denominado *Process Sale* apresentam exemplos de períodos compostos.

"Cashier tells Customer the total, and asks for payment."

"*Customer pays and System handles payment."*

Na primeira oração, o substantivo (ou nome) *Cashier* é o sujeito dos dois predicados: *tells Customer the total* e *asks for payment*. Já a segunda frase apresenta dois sujeitos, o sujeito *Customer* para o primeiro predicado, e o sujeito *System* para o segundo predicado.

Durante a criação das regras que interpretam e geram tuplas para sugestão de refinamento da ontologia, estas duas estruturas simples de frases devem estar contempladas:

Para a primeira frase, a solução encontrada é quebrar a frase na vírgula, separando o processamento das duas orações. Para isto, a abordagem prevê a etapa de pré-processamento das frases, antes de aplicá-las alguma regra. Sendo assim, a primeira frase do exemplo anterior *"Cashier tells Customer the total, and asks for payment"* gera duas orações:

- *Cashier tells Customer* – primeira oração, que apresenta a estrutura sujeito *Cashier* e predicado *tells Customer*.
- *Cashier asks for payment* – segunda oração, onde o sujeito da oração anterior escreve o sujeito da segunda oração. Para formar esta oração, foi necessário aplicar o *tagger Stanford parser* para identificar o sujeito, e formar a segunda oração estruturada em forma de sujeito e predicado.

Sendo assim, a etapa de pré-processamento foi necessária para transformar os períodos compostos em períodos simples, e no exemplo anterior, para estruturar a frase em oração, com sujeito e predicado, foi necessário o processamento da frase pelo *parser* para identificar o sujeito.

Na outra frase utilizada para exemplificar o período composto "*Customer pays and System handles payment"* não foi necessário o pré-processamento, pois a estrutura foi prevista através de regras, onde a conjunção coordenativa *and* (representada pelo *POS-tagger* como *CC – Coordinating conjunction*) declara a possibilidade de após a primeira oração, existir uma segunda oração. No caso do exemplo escolhido, a segunda oração que forma o período possui o sujeito explícito, não sendo necessário submeter à frase ao processamento do *parser* para identificar o sujeito. Sendo assim, seguindo a notação σ *R* (relação) = {domínio, *range*} [CIM06], as duas frases utilizadas de exemplo de período composto neste trabalho formam as seguintes tuplas:

- σ R (*tells*) = (*Cashier*, *Customer*) – representado na aplicação de apoio à abordagem com notação (*Cashier,tells,Customer_total*);
- σ R (*asks_for*) = (*Cashier*, *payment*) – representado na aplicação de apoio à abordagem com notação (*Cashier,asks_for,payment*);
- σ R (*do*) = (*Customer, pays*) – representado na aplicação de apoio à abordagem com notação (*Customer,do,pays*);

- σ R (*handles*) = (*System*, *payment*) – representado na aplicação de apoio à abordagem com notação (*System,handles,payment*).

A estruturação das regras também contou com outra importante consideração da análise gramatical. Um verbo pode ou não vir acompanhado de um advérbio (que modifica o verbo, adjetivo ou advérbio) e um substantivo pode ou não vir acompanhado de um adjetivo (modificador do nome). Sendo assim, as classes gramaticais advérbio e adjetivo etiquetadas pelo *parser Stanford POS-tagger* como RB e JJ respectivamente, são consideradas diferencialmente nas regras.

Os advérbios podem concatenar-se ao verbo ou ao nome, conforme posição que se encontra na oração. Para isto, durante a programação das regras foi estabelecida a seguinte estrutura:

- Se o advérbio (RB) encontra-se posicionado antes do verbo (VB), concatena-se com o verbo ao formar a tupla.
- Se o advérbio (RB) encontra-se posicionado após a ocorrência do verbo (VB) na oração, concatena-se com o nome (NN) mais próximo do verbo, para formar a tupla.
- Se o advérbio (RB) encontra-se posicionado antes da ocorrência do adjetivo (JJ) na oração, concatena-se com o adjetivo (JJ) para formar a tupla.

Um exemplo da ocorrência do advérbio (RB) no caso de uso *process sale* extraído de Larman [LAR07], é apresentado logo abaixo:

- A oração *"Tax is correctly calculated."*, após processada pelo *parser Stanford POS-Tagger* retorna a estrutura anotada **Tax_NNP is_VBZ correctly_RB calculated_VBN**, formando a tupla (Tax,is,correctly_calculated) onde, *Tax* representa um conceito na ontologia, *correctly_calculated* outro conceito na ontologia e *is* o relacionamento entre os conceitos.

A classe gramatical adjetivo, por sua vez, caracteriza um substantivo ou nome, designando-o características como modo ou estado. Para representar esta classe na ontologia, as regras prevêem o relacionamento *can_be* quando da ocorrência do adjetivo antes do substantivo na oração. Um exemplo da ocorrência do adjetivo (JJ) no caso de uso *process sale* extraído de Larman [LAR07], é apresentado logo abaixo:

- A oração *"Cashier starts a new sale."*, após processada pelo *parser Stanford POS-Tagger* retorna a estrutura anotada **Cashier_NN starts_VBZ a_DT new_JJ sale_NN**, formando as tuplas (*Cashier,starts,sale*) e (*sale,can_be,new*), onde *Cashier* representa um conceito na ontologia, *sale* outro conceito na ontologia e *starts* o relacionamento entre os conceitos. Sendo *new* pertencente à classe adjetivo, cria-se outro relacionamento que apresenta o estado do conceito *sale*, que pode estar caracterizado como *new*. A motivação do uso de *can_be* está no fato de que em outro ponto da descrição pode haver outro adjetivo modificando o mesmo substantivo. Dessa forma, optou-se por não denominar diretamento o relacionamento como *is*.

A classe gramatical da preposição "*to"* (TO), entre verbos também deverá ser mantida na descrição do nome do relacionamento. Assim, para a sentença *"Government Tax Agencies **Want to collect** tax from every sale"*, a tupla resultante será *(Government_Tax_Agencies,Want_to_collect,tax_sale)*.

As regras ainda apresentam algumas considerações especiais, principalmente para as ocorrências das preposições ou conjunções identificadas pelo *parser* através da etiqueta IN. Nos exemplos abaixo (frase extraída de Larman [LAR07]):

"Price calculated from a set of price rules."

"Cashier asks for payment."

Quando identificada a existência de IN após o verbo, e IN etiqueta as preposições *from* ou *for*, concatena-se o *from* ou o *for* com o verbo, formando as tuplas que segue: (Price,calculated_**from**,set_rules) e (Cashier,asks_**for**,payment).

O existencial *there* também necessita de tratamento especial. Convencionou-se na abordagem que ao identificar a etiqueta EX, será criada a tupla conforme segue:

- Para a frase *"There are product rebates"*, a tupla retornada pela aplicação de apoio à abordagem é (System,has,product_rebates), onde o existencial *there* (EX) é substituído por *System* e o verbo *are* por *has*.

Para as frases que apresentam a estrutura **NN** (nome) + **VBN** (verbo na forma nominal - particípio), a regra para identificar a tupla resultante desta formação envia o

verbo nominal para a *WordNet* [WOR10] que retorna o verbo no infinitivo, resultando na tupla conforme exemplo abaixo:

- A frase *"Commissions **recorded**"*, resulta na tupla *(Commissions,do,record)*.

Porém, para as ocorrências de frases estruturadas com **nome + verbo + nome (se existir) + forma nominal verbal particípio** (NN+VB+VBN), o verbo nominal concatena com o nome (se existir), ou representa sozinho um conceito (*range*) na tupla formada.

A frase *"Salesperson wants sales commissions **updated**"*, resulta na tupla *(Salesperson,wants,sales_commissions_updated)* e apresenta o verbo nominal particípio *"updated"* concatenado com as demais ocorrências nominais.

Os verbos na forma nominal gerúndio indicam uma ação que está sendo realizada (em andamento) e tiveram a sua ocorrência tratada diferentemente nas regras. Assim, convencionou-se que para as frases estruturadas na forma **nome + verbo + nome + verbo nominal gerúndio** (NN+VB+NN+VBG), o verbo nominal concatena com o nome mais próximo. Para exemplificar esta estrutura de frase, é encontrada na descrição detalhada do caso de uso *process sale* [LAR07] a sentença:

"System detects anomalies preventing recovery"

Onde:

- *System = nome próprio(NNP);*
- *detects = verbo (VBZ);*
- *anomalies = nome no plural (NNS);*
- *preventing = verbo na forma nominal gerúndio (VBG);*
- *recovery = nome (NN).*

Esta sentença, após processada pela regra adequada, concatena a ocorrência do verbo no gerúndio "*preventing*" com as ocorrências nominais (anterior e/ou posterior), resultando na tupla *(System,detects,anomalies_preventing_recovery)*.

Durante a criação das regras, verificou-se a necessidade de tratar as frases que terminam com a seqüência gramatical **preposição + nome** (IN + NN) **ou TO + nome** (TO + NN) diferentes na formação do conceito da tupla gerada, evitando a formação de *n-*

gramas desnecessariamente. Sendo assim, para a sentença original *"Cashier cancels sale on System"*, onde:

- *Cashier = nome (NN);*
- *cancels = verbo (VBZ);*
- *sale = nome (NN);*
- *on = preposição (IN);*
- *System = nome (NN).*

A tupla resultante é (*Cashier,cancels,sale*), onde:

- *Cashier = conceito (domínio);*
- *cancels = relacionamento entre dois conceitos (domínio e range);*
- *sale = conceito (range).*

Neste exemplo, portanto, *"on System"* são desconsiderados na definição da abordagem para formação da tupla, por não acrescentar informação relevante para a ontologia. Outro exemplo é a frase *"System signals error to Cashier"*, onde:

- *System = nome próprio (NNP);*
- *signals = verbo (VBZ);*
- *error = nome (NN);*
- *to = para (TO);*
- *Cashier = nome próprio (NNP).*

A tupla resultante do processamento desta sentença é (*System,signals,error*), onde:

- *System = conceito (domínio);*
- *signals = relacionamento entre dois conceitos (domínio e range);*
- *error = conceito (range).*

Neste outro exemplo, as palavras "*to Cashier"* não foram consideradas na tupla resultante, por pertencerem à classe gramatical preposição (para) e nome próprio respectivamente.

Erros de classificação gramatical foram identificados na etiquetagem do *parser Stanford POS-Tagger*, o que acarretou em problemas na criação das tuplas. Um exemplo que ocorreu comumente no processamento do caso de uso *process sale* [LAR07] é o reconhecimento do verbo como NNS (substantivo no plural). A oração *"System records sale line item and presents item"*, extraída de Larman [LAR07], depois de anotada pelo *parser Stanford POS-Tagger*, classifica a palavra *records* na classe gramatical substantivo no plural (NNS). Embora seja possível classificar a palavra como um substantivo, ela pertence à classe gramatical verbo no contexto em que está inserida. Para estas estruturas, onde o *parser* identifica um ou mais substantivos seguidos de um NNS e de um ou vários substantivos (NN+NNS+NN), a abordagem classifica o NNS como sendo o relacionamento entre os conceitos. As tuplas formadas neste caso, para a frase de exemplo proposta são, respectivamente, *(System,records,sale_line_item)* e *(System,presents,item_description)*.

4.2.2 Pré-processamento das sentenças

A etapa de pré-processamento prevê a realização da limpeza das sentenças. A limpeza consiste na remoção dos caracteres que possuem classes gramaticais que não são de interesse para leitura da regra que irá gerar a tupla. As classes gramaticais que são eliminadas durante o pré-processamento estão listadas a seguir:

- PP$ e POS (*possessive pronoun*): pronomes possessivos não são considerados nas regras, de modo que a etapa de pré-processamento elimina da sentença a ocorrência desta classe gramatical.
- PDT (*determiner pronoun*): determinantes (como por exemplo, meu, minha, este, esta) não são considerados nas regras.
- PRP (*personal pronoun*): pronomes pessoais são desconsiderados caso ocorram na sentença.
- Símbolos $, #, “”, ‘’: os símbolos como o dólar, *Sharp*, aspas duplas e aspas simples também são tratados antes das regras, e eliminados da sentença antes do processamento.
- LS (*list item maker*) e SYM (*symbol mathematical/scientific*): os símbolos de marcação são eliminados por não representar relevância na formação das tuplas.

- WDT (*which determiner*): removido determinante *which* antes do processamento pelas regras por não possuir importância para formação das tuplas.
- WP (*which pronoun*) e WP$ (*possessive which pronoun*): removido *which* pronome antes do processamento da frase pelas regras por não possuir importância para formação das tuplas.
- WRB (*which adverb*): removido também o *which* (pertencente à classe gramatical advérbio) antes do processamento da frase pelas regras por não possuir importância para formação das tuplas.
- Colon (*punctuation*): removida a ocorrência do sinal de pontuação dois pontos na sentença.

Na etapa de pré-processamento também convencionou-se desconsiderar o texto existente entre parênteses, por geralmente tratar-se de exemplos e não possuir importância para a geração das tuplas. Sendo assim, a sentença abaixo:

"Customer tells Cashier they have a tax-exempt status (e.g., seniors, native peoples)."

Após pré-processada, retorna a frase assim: *"Customer tells Cashier they have a tax-exempt status."* Somente após a etapa de pré-processamento, onde as frases são reescritas, é que serão submetidas ao processamento das regras.

A formação das sentenças do campo s*takeholders* da descrição do caso de uso apresenta uma característica particular, pois descrevem o sujeito seguido do sinal de pontuação dois pontos, e as frases subseqüentes iniciando com um verbo. Por causa desta estrutura de frase foi necessário estabelecer um tratamento diferenciado, reescrevendo as frases com o sujeito, cada vez que a frase iniciar com um verbo. Um exemplo desta estrutura está exemplificado na Figura a seguir:

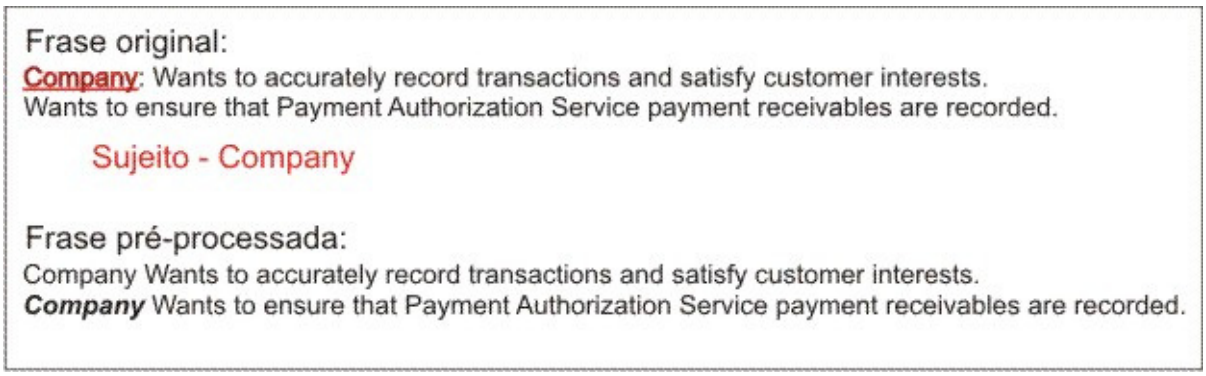

Figura 16 - Exemplo da sentença pré-processada.

O próximo Capítulo apresenta a ferramenta de apoio à abordagem deste trabalho.

5. FERRAMENTA DE APOIO À ABORDAGEM

A ferramenta *OntSoft* é um aplicativo de apoio à abordagem proposta. Seu código foi escrito em linguagem de programação Java, e integra recursos como o *parser POS-Tagger* e o *tagger* de *Stanford* [NAT10] [NAT10a], o tesauro *WordNet* [WOR10], e a biblioteca *Jena* [JEN10], para apoio ao processo de criação e refinamento da ontologia em linguagem OWL.

5.1 Ferramenta *OntSoft*

A ferramenta de apoio à abordagem proposta neste trabalho propõe o processamento do texto de entrada, que descreve o detalhamento de casos de uso, resultando em um conjunto de saída que representa a tupla (conceito, relacionamento, conceito) que tem o objetivo de refinar e enriquecer a ontologia inicial.

5.1.1 Funcionamento do *OntSoft*

A ferramenta *OntSoft* funciona de acordo com o esquema apresentado na Figura 17:

1. Recebe como entrada um arquivo texto (que descreve o detalhamento de casos de uso);
2. Cada frase do arquivo é lida e anotada pelo *parser POS-tagger*, responsável pela anotação lingüística da classe gramatical de cada palavra;
3. A frase anotada então passa pelo módulo de pré-processamento, responsável pela eliminação das classes gramaticais que não possuem relevância para o processamento das regras, limpeza de pontuação e reestruturação da frase com a identificação do sujeito entre outras;
4. A etapa de processamento do *tagger Stanford parser* é chamada quando é necessária a identificação do sujeito da oração, para reescrevê-la com a estrutura sujeito-predicado;
5. A frase pré-processada é então encaminhada ao interpretador, que identifica a regra adequada à sua estrutura gramatical;

6. A etapa de processamento através do tesauro *WordNet* é chamada somente em algumas regras, quando necessário (no Capítulo 4 é apresentada em detalhes a utilização do tesauro para formação das tuplas);
7. Cada sentença retorna uma ou mais tuplas que representam (conceito, relacionamento, conceito).

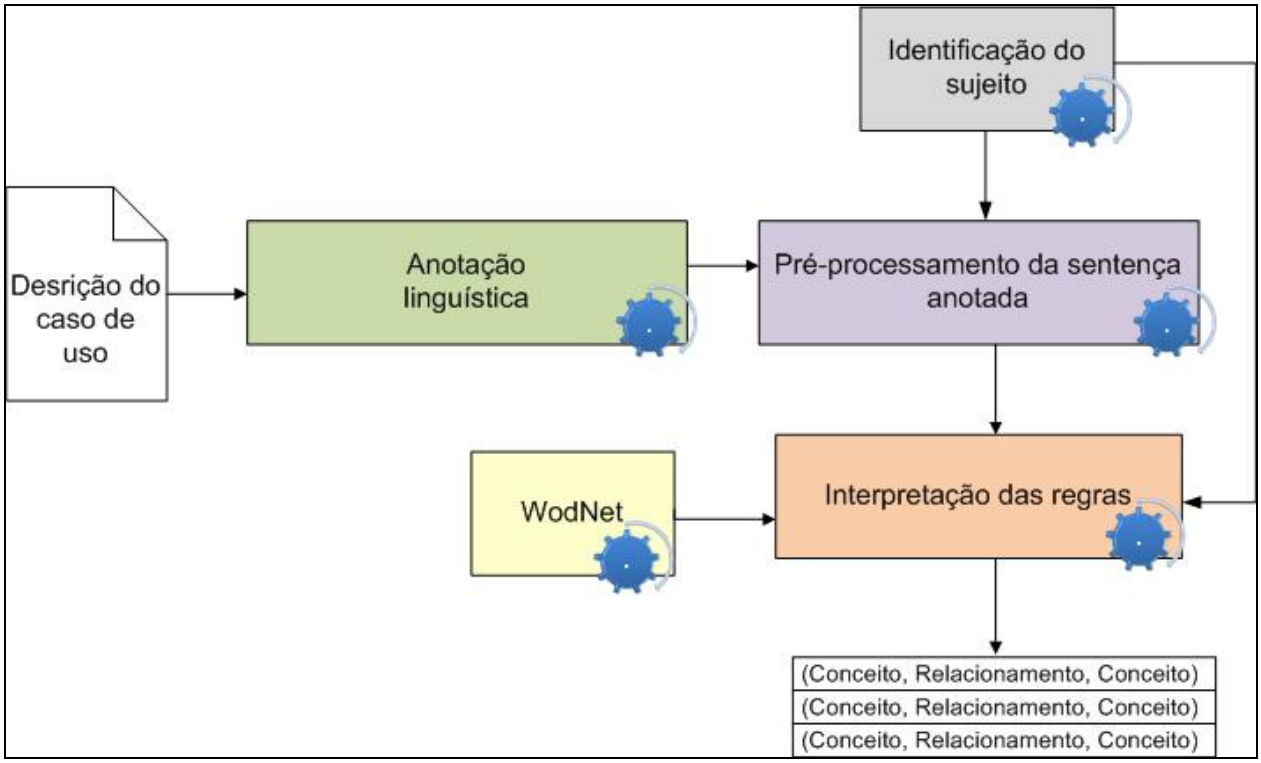

Figura 17 - Funcionamento do *OntSoft.*

5.1.2 Arquitetura do *OntSoft*

Para desenvolver a ferramenta, foi necessária a implementação de um conjunto de classes. A Figura 18 apresenta o diagrama de classes da aplicação *OntSoft*.

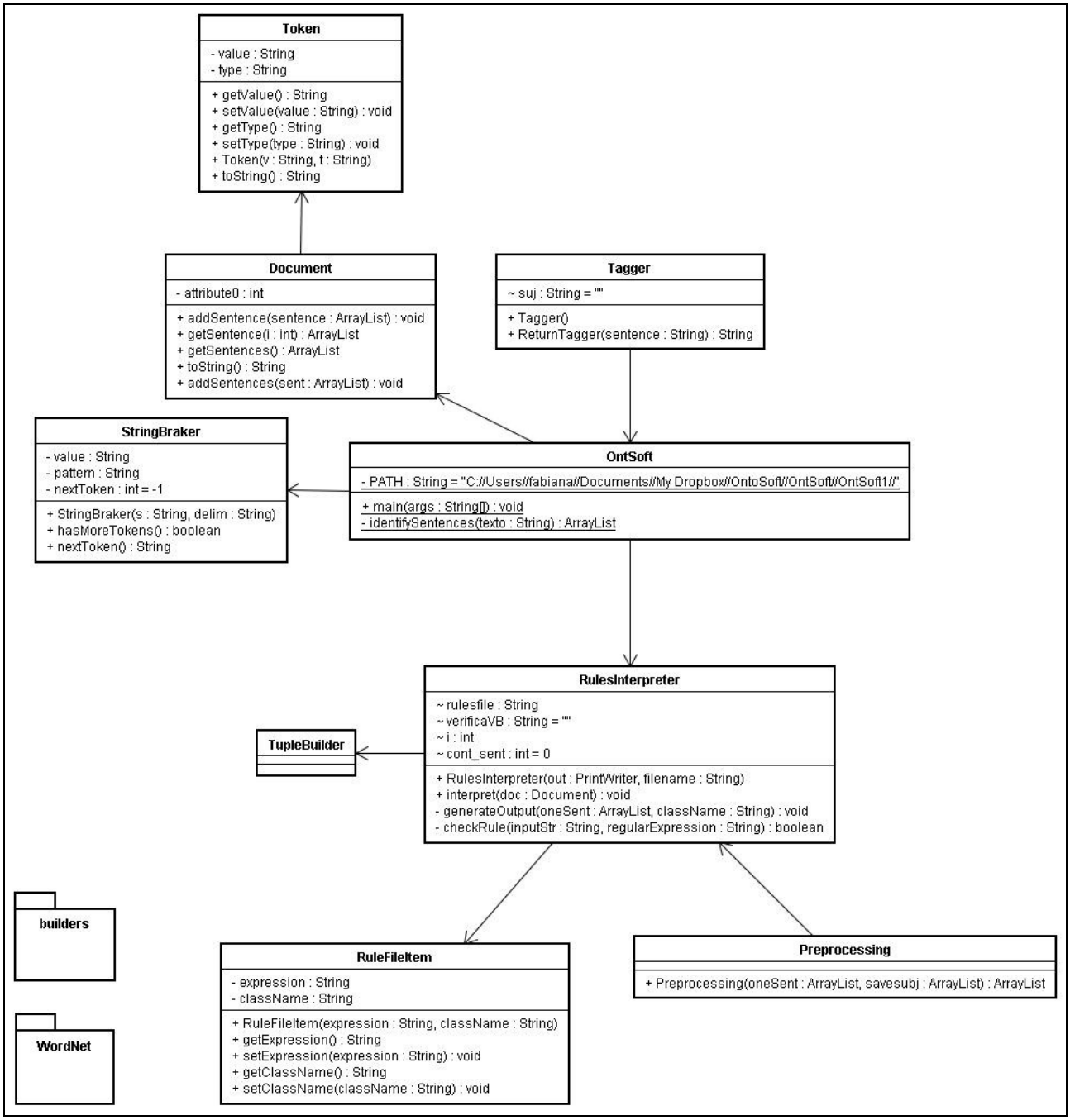

Figura 18 – Principais classes da aplicação.

As principais classes que constituem o aplicativo *OntSoft* encontram-se listadas abaixo:

- *OntSoft*: Classe responsável pela leitura do arquivo do detalhamento de caso de uso;
- *Tagger*: Classe responsável pelas anotações lingüísticas do arquivo;
- *Document*: Classe responsável por separar as anotações lingüísticas do *parser* (classe gramatical) do texto (*token*);

- *Preprocessing*: O módulo de pré-processo encontra-se detalhado na próxima subseção, e no diagrama de classes está descrito pela classe *preprocessing*, responsável por preparar a sentença antes de ser "casada" com as expressões regulares que definem cada regra;
- *Rules Interpreter*: Classe responsável por verificar se a sentença "casa" (classe *Matcher*) com alguma expressão regular definida no arquivo em formato de texto denominado *rules*.
- *Tuple Builder*: Classe que contém as seqüências de caracteres válidos (etiquetas do *parser*). Exemplo: "JJ" para adjetivo, "VB" para verbo e etc. são aceitos e lidos pelo programa;
- *Builders*: Representa todas as classes que implementam as regras definidas no arquivo de regras chamado *rules*. A saída destas classes é uma ou mais tuplas referentes ao processamento da sentença.
- *String Braker*: Classe que identifica o limite de cada sentença do arquivo texto que contém a descrição detalhada de uso.
- *WordNet:* O pacote *WordNet* contém as classes que realizam a chamada para o banco de dados lexical.

Para exemplificar o fluxo de funcionamento da ferramenta *OntSoft*, para a sentença *"Commissions recorded."* [LAR07], a classe *OntSoft* lê o arquivo de entrada (detalhamento do caso de uso), realiza a leitura da sentença e encaminha ao *Stanford POS-Tagguer* que retorna o texto assim anotado: *"Commissions_NNS recorded_VBN ._."*. A sentença anotada é enviada para as classes Document e Token, onde são separadas as anotações em valores e tipos. Neste exemlo, os valores seriam *strings* contendo as palavras *commissions* e *recorded*, e os tipos (classe gramatical que pertence cada palavra) *NNS* e *VBN*. A classe RulesInterpreter é chamada e então é enviada para a classe Preprocessing que realiza a limpeza e re-escrita da sentença (se necessário) e depois é identificada a regra que "casa" com a sentença. A regra identificada pela classe RuleFileItem é instanciada (cada classe implementa uma única regra da lista) prevê a estrutura **NN+VBN** para este exemplo. Esta regra identifica a estrutura gramatical da sentença através da classe TupleBuilder. A regra deste exemplo ainda chama a *WordNet*, que retorna o verbo no infinitivo *record*, e então a tupla *(commissions,do,record)* é retornada do processamento.

O fluxo de funcionamento para a sentença de exemplo *"System signals error and rejects entry."* [LAR07] apresenta a diferença de a regra que "casa" com a sentença prever a existência da conjunção coordenativa *and*. Neste exemplo, como o sujeito da frase é o mesmo para os dois períodos, a classe *Tagger* é chamada com a finalidade de identificar o sujeito. Com isto, as tuplas são retornadas: *(system,signals,error)* e *(system,rejects,entry)*.

Outros exemplos ainda necessitam que o sujeito seja identificado durante o pré-processamento, e então para a sentença de exemplo *"Cashier restarts System, logs in, and requests recovery of prior state."* [LAR07], a classe Preprocessing chama a classe Tagger para identificar o sujeito e formar as sentenças: *"Cashier restarts System."*, *"Cashier logs System."* e *"Cashier requests recovery of prior state."*. Cada sentença retornada da classe Preprocessing é "casada" com a sua respectiva regra e retornam as seguintes tuplas: *(Cashier,restarts,System), (Cashier,logs,System), (Cashier,requests,recovery)* e *(state,can_be,prior).*

O Capítulo a seguir faz uma breve apresentação sobre os dois sistemas usados para exemplo de uso neste trabalho: o sistema PDV *ProxGer* [LAR07] utilizado para definir e testar as regras da abordagem e o sistema *Time Card* [ARR01] utilizado para testar estas regras.

6. EXEMPLO DE USO

O sistema PDV *ProxGer* [LAR07] foi utilizado como modelo tanto para criação quanto para teste das regras de geração das tuplas a partir do texto de entrada. Para teste destas regras também foram utilizadas as descrições detalhadas de casos de uso do sistema *Time Card* [ARR01]. Foram utilizados apenas os detalhamentos de casos de uso destes dois sistemas devido ao fato de existir grande dificuldade de encontrar exemplos completos de sistemas em domínio público. Neste Capítulo, encontram-se detalhados os dois sistemas, as ontologias que representam o modelo de domínio das aplicações, os termos mais freqüentes extraídos a partir das descrições de casos de uso e a análise sobre a *survey* aplicada nesta pesquisa.

6.1 Visão Geral do Sistema *ProxGer* (extraído de Larman [LAR07])

Um sistema PDV é uma aplicação de computador utilizada para registrar vendas e gerenciar pagamentos. Consiste em um sistema tipicamente utilizado em lojas de departamento e inclui componentes de hardware, como um leitor de código de barras, e um software para rodar o sistema. Tem interfaces com várias aplicações de serviços, como, por exemplo, uma aplicação de cálculo de impostos e uma aplicação de controle de estoque. Esses sistemas devem ser relativamente tolerantes a falhas, ou seja, mesmo que os serviços remotos fiquem temporariamente não disponíveis (como, por exemplo, o controle de estoque), eles devem ser capazes de capturar as vendas e tratar pelo menos os pagamentos em dinheiro. Um sistema PDV deve dar suporte de forma incremental a múltiplos e variados terminais e interfaces no lado do cliente.

6.1.1 Ontologia que representa o Modelo de Domínio do sistema *ProxGer*

A ontologia a seguir foi modelada a partir do Modelo de Domínio do sistema PDV *ProxGer*.

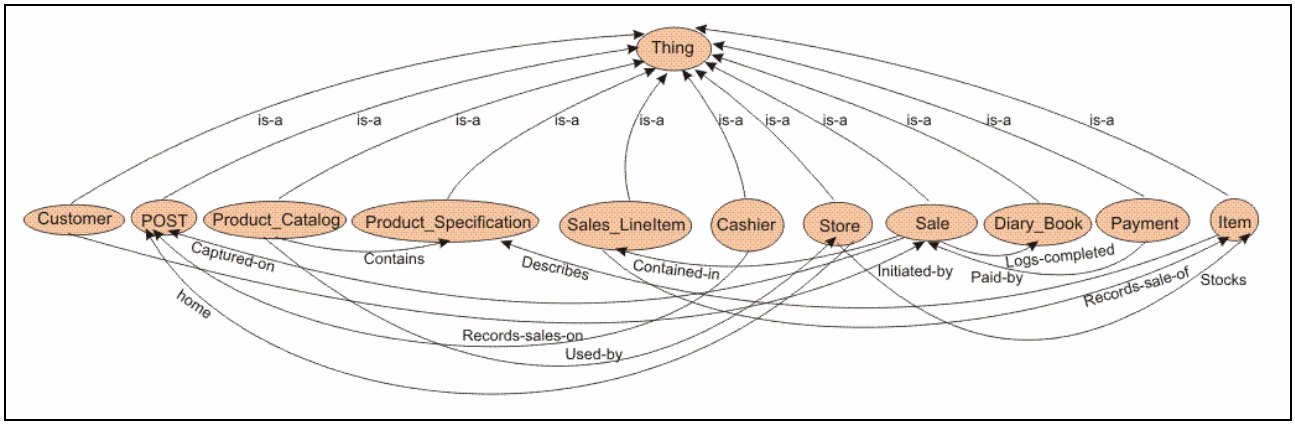

Figura 19 – Ontologia que descreve o modelo de domínio do sistema *ProxGer*.

Apenas para facilitar a leitura, as tuplas abaixo descrevem os relacionamentos laterais da Figura 19:

- (Sale,Captured-on,POST);
- (Sale,Contained-in,Sales_LineItem);
- (Product_Catalog,Contains,Product_Specification);
- (Item,Describes,Product_Specification);
- (Customer,Initiated-by,Sale);
- (Sale,Logs-completed,Diary_Book);
- (Payment,Payd-by,Sale);
- (Sales_LineItem,Records-sale-of,Item);
- (Cashier,Records-sales-on,POST);
- (Store,Stocks,Item);
- (Product_Catalog,Used-by,Store);
- (Store,home,POST).

6.1.2 Extração de termos – Sistema *ProxGer*

Inicialmente, o mapeamento do diagrama de classes que descreve o modelo de domínio da aplicação *ProxGer* identificou uma série de conceitos e relacionamentos. Na Tabela 2 a seguir, encontram-se listados estes conceitos e relacionamentos previamente identificados.

Tabela 2 – Lista inicial de conceitos e relacionamentos – Sistema PDV

Conceitos	Relacionamentos
Sale	Captured-on
POST	Contained-in
Sales_LineItem	Contains
Product_Catalog	Describes
Product_Specification	Initiated-by
Item	Logs-completed
Customer	Payd-by
Diary_Book	Records-sale-of
Payment	Records-sales-on
Cashier	Stocks
Store	Used-by

A geração das regras levou em consideração o modelo de estrutura de frase escrito por Larman [LAR07] e definido no detalhamento de caso de uso denominado *process sale*. O detalhamento do caso de uso está descrito no Anexo A, conforme formatação lida pela ferramenta de apoio à abordagem proposta neste trabalho. O processamento do detalhamento de caso de uso *process sale* [LAR07] gera um conjunto de tuplas que descrevem a estrutura (conceito, relacionamento, conceito). Portanto, as tuplas identificadas sugerem conceitos e relacionamentos a partir de termos extraídos do texto descrito em linguagem natural. Na verdade, estes termos sugerem conceitos que poderão existir na ontologia final. Os termos identificados podem ser classificados em uni gramas, bi gramas, tri gramas e *n* gramas, de acordo com o número de seqüência de palavras que os formam.

Sendo assim, abaixo está relacionada uma lista com exemplos de termos candidatos a conceitos que aparecem com maior número de ocorrências nas tuplas geradas (conceito, relacionamento, conceito) extraídos da descrição detalhada de caso de uso *process sale*. Também está relacionada o número de vezes que estes termos candidatos a conceitos aparecem nas tuplas e se o termo é um conceito pertencente à ontologia inicial, de acordo com sua classificação: uni gramas, bi gramas, tri gramas ou *n* gramas (mais de 3 palavras). Os termos bi gramas, tri gramas e *n* gramas algumas vezes apresentam uma de suas palavras como conceitos mapeados na ontologia inicial. Quando isto ocorre, na Tabela 3 a informação da coluna correspondente é apontada com o valor “sim” e ao lado o nome do conceito correspondente é discriminado.

Tabela 3 – Termos candidatos: *Process Sale.*

Classificação	Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Uni gramas	Cashier	62	sim
	System	58	não
	Customer	20	sim
	Sale	17	sim
	Error	10	não
	Manager	8	não
	Price	8	não
	Alternate	5	não
	Company	5	não
Bi gramas	Customer_payment	6	sim – Customer e payment
	item_ID	6	sim - Item
	gift_receipt	3	não
	Manager_override	3	não
	override_operation	3	não
	price_entry	3	não
Tri gramas	Payment_Authorization_Service	4	sim - Payment
	item_entry_mode	2	sim - item
	limit_for_Cashiers	2	sim - Cashier
	manual_category_code	2	não
	retrieval_POS_register	2	sim - POS
N gramas	easily_display_entered_items	2	sim - item (plural)
	item_identifier_for_removal	2	sim - item
	tax_calculation_system_service	2	não

O processamento do detalhamento de caso de uso resultou ainda na identificação de tuplas repetidas. Isto ocorreu por motivo da repetição de frases no fluxo de extensão ou devido aos tratamentos especiais previstos nas regras gerarem, determinadas vezes, tuplas iguais a outras existentes. A fim de exemplificar, para as sentenças existentes em *process sale*, *"System signals error to Cashier"* e *"System signals error"* o resultado do processamento das regras gera a mesma tupla *(System,signals,error)*. Isto ocorre devido ao fato da oração *"System signals error to Cashier"* entrar na regra especial que desconsidera a terminação final da frase: preposição *"to"* seguida de um nome (TO + NN). A lista das tuplas repetidas encontra-se no Apêndice A desta dissertação.

6.1.2.1 Corpus e as tuplas identificadas

O número total de sentenças do detalhamento de caso de uso *process sale* [LAR07] é 150 frases. As 150 sentenças geram, ao reescrever períodos compostos em períodos simples, 144 sentenças que são processadas pelas regras e formam tuplas. As sentenças definidas no detalhamento de caso de uso como o exemplo *"*b. At any time, System fails:"* não são lidas pelas regras e consequentemente não geram tuplas. Este exemplo explica o porquê da abordagem processar através das regras menos sentenças do que as existentes na descrição detalhada. No Apêndice A encontram-se listadas todas as sentenças que não são processadas pelas regras.

A quantidade total de tuplas identificadas a partir do processamento das sentenças através das regras é 218, sendo que 195 tuplas são distintas. Dentre as 150 sentenças existentes no arquivo, 22 destas não são lidas por nenhuma expressão regular que implementa as regras propostas neste trabalho. A Tabela 4 apresenta esta análise.

Tabela 4 – Total de sentenças e tuplas: Sistema *ProxGer.*

Descrição	Quantidade
Total de sentenças	150
Total de sentenças processadas	144
Total de sentenças duplicadas	23
Total de tuplas identificadas	218
Total de tuplas distintas	195

6.1.3 Ontologia Final – Sistema *ProxGer*

A ontologia final, gerada a partir da descrição detalhada do caso de uso *process sale*, referente ao sistema *ProxGer* [LAR07], encontra-se modelada conforme apresenta a Figura 20. O refinamento da ontologia que representa o sistema *ProxGer*, considerou as tuplas identificadas a partir das regras, e consideradas relevantes por pelo menos 60% dos participantes, segundo a *survey* respondida por dez participantes (mais informações sobre as respostas da *survey* conforme apresenta a subseção 6.4). Para melhor visualização da ontologia final, os novos conceitos e relacionamentos estão representados com cores diferentes da ontologia inicial.

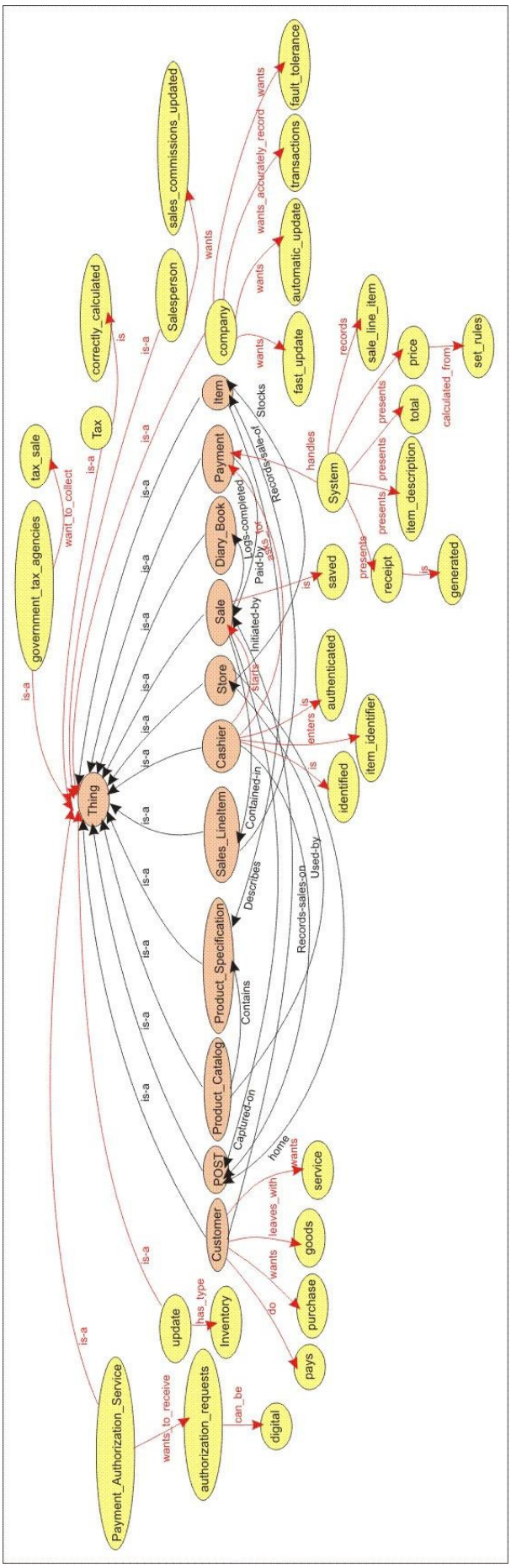

Figura 20 – Ontologia final gerada a partir dos detalhamentos de caso de uso do sistema *ProxGer*.

6.2 Visão Geral do Sistema *Time Card* (extraído de [ARR01])

O Sistema *Time Card* [ARR01] é um sistema de cartão ponto e prevê a utilização do aplicativo para gravar as horas faturáveis e não-faturáveis realizadas pelos funcionários de uma empresa. O sistema tem como requisito inicial o acesso ao aplicativo tanto no escritório quanto em casa. O sistema prevê a criação de um usuário e senha para cada funcionário, solicitando a mudança da senha na primeira vez que o usuário "*logar*" no sistema.

6.2.1 Ontologia que representa o Modelo de Domínio do Sistema *Time Card*

A ontologia a seguir foi modelada a partir do Modelo de Domínio do Sistema *Time Card*.

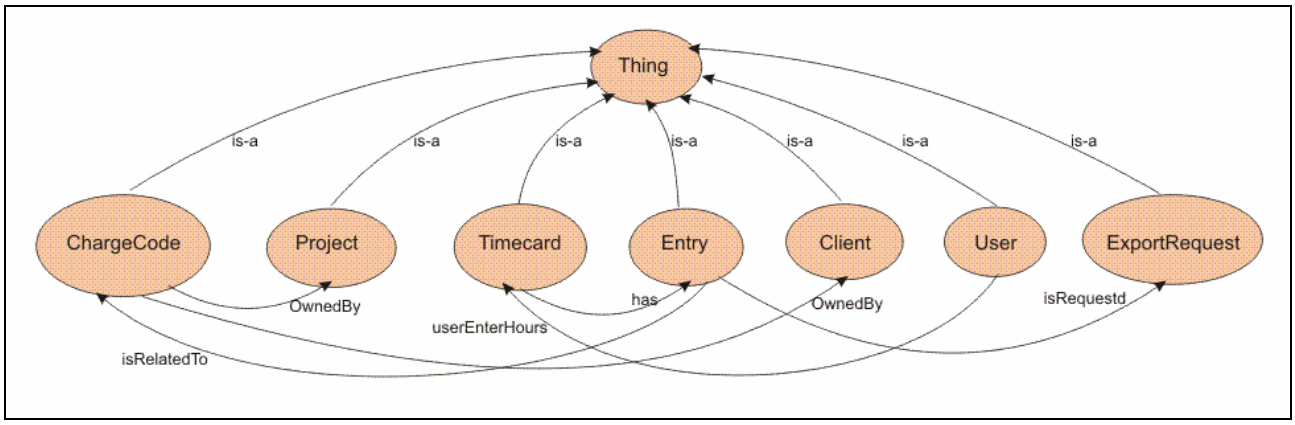

Figura 21 – Ontologia que descreve o modelo de domínio do sistema *Time Card*.

Apenas para facilitar a leitura, as tuplas abaixo descrevem os relacionamentos laterais da Figura 21:

- (TimeCard,has,Entry);
- (Entry,isRelatedTo,ChargeCode);
- (Entry,isRequested,ExportRequest);
- (ChargeCode,OwnedBy,Client);
- (ChargeCode,OwnedBy,Project);
- (User,userEnterHours,Timecard).

A seção a seguir apresenta a lista dos termos com maior número de ocorrência nas tuplas, candidatos a conceitos da ontologia.

6.2.2 Extração de termos – Sistema *Time Card*

A ontologia inicial que descreve o modelo de domínio da aplicação *Time Card* apresenta os conceitos e relacionamentos, conforme listado na Tabela 5.

Tabela 5 – Conceitos e relacionamentos da ontologia inicial – Sistema *Time Card.*

Conceitos	**Relacionamentos**
TimeCard	has
Entry	isRelatedTo
ChargeCode	IsRequested
ExportRequest	OwnedBy
Client	UserEnterHours
Project	
User	

São seis descrições de casos de uso referentes ao sistema *Time Card*. No Anexo A, encontram-se detalhados estes casos de uso. Os detalhamentos de caso de uso do sistema *Time Card* geraram menos tuplas que a descrição detalhada do caso de uso *process sale*. Isto ocorreu porque a estrutura das sentenças do caso de uso utilizado para gerar as regras é diferente da estrutura de frases utilizadas para detalhar o sistema *Time Card*. Um exemplo bastante perceptível à primeira vista é que o autor dos casos de uso utilizados para teste utilizou-se muito da classe gramatical dos verbos modais, e também iniciou geralmente as frases do fluxo principal, alternativo e extensão com o determinante "*the"*.

Os detalhamentos do sistema *Time Card* foram utilizados para testar as expressões regulares que implementam as regras desenvolvidas utilizando de modelo o sistema de PDV *ProxGer*. Através do processamento dos seis detalhamentos de casos de uso que descrevem o sistema *Time Card* foi extraído tuplas com a finalidade de refinar a ontologia inicial do sistema. A Tabela 6 apresenta alguns dos termos com maior número de ocorrência nas tuplas identificadas, organizados em uni gramas, bi gramas, tri gramas e *n* gramas.

Tabela 6 – Termos candidatos: Sistema *Time Card.*

Classificação	Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
	Employee	25	não

Uni gramas	Administrator	18	não
	User	18	sim
	System	17	não
	Unable	8	não
	Due	7	não
	Administrative	7	não
	Failure	6	não
	Project	5	sim
Bi gramas	Administrative_user	11	sim - user
	Charge_code	9	sim
	Logged_user	6	sim - user
Tri grama	View_existing_employees	2	não
n grama	Record_time_use_case	4	não

6.2.2.1 Corpus e as tuplas identificadas

O número total de sentenças dos seis detalhamentos de casos de uso do sistema *Time Card* é 143 sentenças, e se encontram assim distribuídas:

- Caso de uso *change password*: possui 21 sentenças;
- Caso de uso *create charge code*: possui 27 sentenças;
- Caso de uso *create employee*: possui 20 sentenças;
- Caso de uso *export time entries*: possui 18 sentenças;
- Caso de uso *login*: possui 20 sentenças;
- Caso de uso *record time*: possui 37 sentenças.

As 143 sentenças após processadas geram 131 tuplas. Das 129 tuplas identificadas, 23 destas são duplicadas gerando, portanto, um total de 106 tuplas distintas. Esta análise desconsidera sentenças duplicadas em cada detalhamento de caso de uso. As sentenças duplicadas comumente ocorrem nos fluxos alternativos e nos fluxos de exceção.

A Tabela 7 apresenta a análise dos números descritos acima.

Tabela 7 – Total de sentenças e tuplas: Sistema *Time Card.*

Descrição	Quantidade
Total de sentenças	143
Total de sentenças duplicadas	25
Total de tuplas identificadas	131
Total de tuplas distintas	106

A subseção a seguir apresenta a ontologia refinada, considerando os resultados obtidos através da *survey* (detalhes sobre a análise da *survey* encontram-se disponíveis na subseção 6.4).

6.2.3 Ontologia Final – *Time Card*

A ontologia final, gerada a partir dos seis detalhamentos de casos de uso que descrevem o sistema *Time Card* [ARR01] está representada na Figura 22. A ontologia foi modelada considerando as tuplas avaliadas na *survey* (conforme apresenta a subseção 6.4) e foram modelados todos os conceitos e relacionamentos considerados relevantes por pelo menos 60% dos participantes da pesquisa.

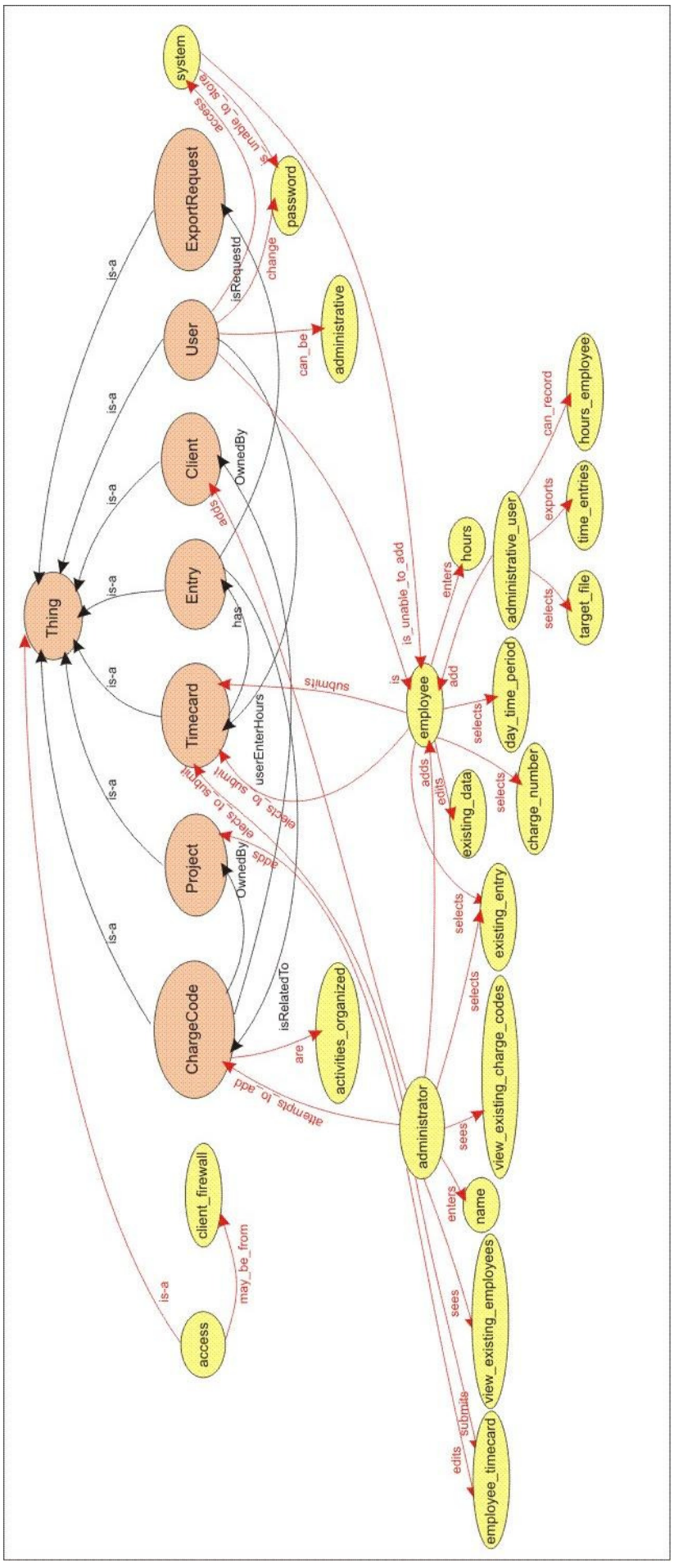

Figura 22 – Ontologia final gerada a partir dos detalhamentos de caso de uso do Sistema *Time Card*.

6.3 Considerações

As descrições detalhadas dos casos de uso dos sistemas utilizados apresentam aproximadamente o mesmo número total de sentenças. Enquanto o detalhamento de caso de uso *process sale* que descreve o sistema de PDV *ProxGer* [LAR07] possui 150 sentenças e produz 195 tuplas distintas, o sistema *Time Card* [ARR01] está descrito por 143 sentenças, gerando 106 tuplas distintas. Isto ocorre porque o sistema PDV foi utilizado como modelo para gerar as expressões regulares que implementam as regras, enquanto que o sistema *Time Card* foi utilizado para verificar a aplicação das regras em outra descrição de caso de uso, com suas características e peculiaridades descritivas próprias.

As descrições detalhadas dos casos de uso do sistema *Time Card*, para os campos de fluxo principal e alternativo ou exceção foram escritas muitas vezes utilizando a voz passiva. É importante salientar que qualquer passo do fluxo, escrito em voz passiva está mal formado, por suprimir informações importantes como o que, quem, onde e quando especificamente [ARL05]. Já os fluxos do sistema de PDV *ProxGer* foi escrito pelo autor na voz ativa, o que contribuiu para a interpretação das sentenças pelas regras, gerando a identificação de um número maior de tuplas.

Alguns dos termos que apresentam maior freqüência na formação das tuplas foram indicados como conceitos relevantes pelos participantes da *survey*. A Tabela a seguir apresenta estes termos que pertencem à ontologia final, refinada de acordo com as respostas obtidas a partir da aplicação da *survey* (informações detalhadas sobre a *suvey* encontram-se no item 6.4).

Tabela 8 – Conceitos identificados como relevantes.

Sistema	Conceitos
ProxGer	system
	price
	company
	payment_authorization_service
Time Card	employee
	system
	administrator
	administrative
	view_existing_employee

Embora alguns termos com maior taxa de ocorrência nas tuplas tenham sido identificados como conceitos relevantes nas tuplas analisadas pelos participantes da *survey*, não é possível aponta-los como relevantes. Isto ocorre porque a maior parte dos termos com maior taxa de ocorrência nas tuplas não foi identificada pelos participantes da *survey* como conceitos pertencentes às tuplas relevantes para o refinamento e enriquecimento da ontologia final. Esta análise indica que, a partir dos exemplos de uso utilizados neste trabalho, a utilização do cálculo da ocorrência dos termos na formação das tuplas não determina a sua relevância para o refinamento da ontologia.

6.4 Pesquisa Empírica

Um estudo empírico é o ato ou operação com a finalidade de descobrir algo novo ou estudar uma hipótese que envolve a pesquisa, coleta e análise de dados para determinar a significância dos dados [BAS99]. Isto envolve estratégias de pesquisa, estudos qualitativos, análises, entre outros. A primeira etapa da pesquisa envolve a coleta e análise dos dados originais [SJO07]. Na literatura estão descritos vários tipos de pesquisa empírica. Os objetivos, propriedades ou resultados almejados podem determinar o tipo mais adequado. Neste trabalho, a estratégia empírica utilizada para análise dos resultados foi a *survey*, e está descrita neste Capítulo.

6.4.1 Survey

Uma pesquisa investiga as relações e resultados [SJO07]. A *survey* é um instrumento de coleta de informações, com a finalidade de descrever, comparar ou explicar o conhecimento, atitudes e comportamento [SHU02] [SHU08]. Não existem variáveis nesta abordagem. O objetivo da *survey* é produzir descrições estatísticas quantitativas ou numéricas a respeito do estudo, analisar os dados a partir de perguntas e respostas de uma amostra da população ou servir como um estudo preliminar de uma pesquisa posteriormente aprofundada.

Utiliza-se *surveys* em ES para coleta de um conjunto de dados de um evento ocorrido para uma população específica, ou para determinar técnicas ou relacionamentos, possibilitando o levantamento do número de variáveis a serem analisadas. A coleta de dados preliminares pode ser através do método qualitativo ou quantitativo. O método qualitativo coleta os dados em formato de textos, imagens ou sons e a análise ocorre independente de medições precisas. Já o método quantitativo coleta os dados numéricos

e aplica estatísticas para analisá-los. São utilizados dados quantitativos para medir um aspecto particular em um processo [FOR01].

A *survey* deste trabalho foi aplicada para preenchimento de 10 participantes, com a finalidade de avaliar os resultados obtidos a partir da extração automática de tuplas dos detalhamentos de casos de uso que descrevem os dois sistemas utilizados como exemplo de uso (conforme descrição apresentada nos itens 6.1 e 6.2 respectivamente). Para o sistema *ProxGer* [LAR07] não foi considerado o campo de fluxo alternativo para análise, devido ao grande número de sentenças e tuplas geradas, o que acarretaria em um tempo muito alto para a realização da *survey*. O preenchimento da *survey* durou, em média, 1 hora e seguiu o seguinte procedimento pelos participantes da pesquisa:

1. Ler as instruções de cada um dos sistemas.
2. Verificar as duas ontologias que descrevem o modelo de domínio dos dois sistemas.
3. Ler os detalhamentos de casos de uso para cada um dos sistemas descritos brevemente.
4. Informar nas planilhas de cada sistema, respectivamente, as tuplas consideradas relevantes, de acordo com a sua percepção, no refinamento e enriquecimento da ontologia inicial.

As respostas de cada participante estão descritas no Apêndice C deste documento. De acordo com as respostas obtidas a partir da aplicação da *survey*, foi possível analisar a razão das tuplas relevantes em relação ao total identificado. A Tabela 9 apresenta a freqüência de acerto da ferramenta para cada detalhamento de caso de uso dos exemplos de uso, considerando como válida as tuplas que possuem relevância para pelo menos 6 participantes da pesquisa (60% da amostra).

Tabela 9 – Freqüência de acerto das tuplas (dados extraídos da análise das respostas da *survey*).

Sistema	**Nome do Caso de Uso**	**Freqüência de acerto**
ProxGer	*Process sale*	50%
Time Card	Change password	20%
	Create charge code	27%
	Create employee	23%

	Export time entries	13%
	Login	18%
	Record time	44%

Considerando a Tabela 9, é possível verificar que as tuplas identificadas a partir da descrição detalhada de caso de uso *process sale* apresentou maior freqüência de aprovação se comparado com a freqüência de acerto obtida dos seis detalhamentos de casos de uso que descrevem o sistema *Time Card*. O sistema *Time Card* apresenta uma média de acerto de 24%, em contrapartida ao índice de 50% obtidos da análise dos resultados das respostas da *survey*. Como anteriormente referenciado, qualquer passo do fluxo de detalhamento de caso de uso, quando escrito em voz passiva, está mal formado [ARL05]. Como os detalhamentos de casos de uso do sistema *Time Card* muitas vezes estavam descritos em voz passiva, dificultou a identificação e a formação das tuplas. Isto acarretou no menor índice de acerto determinado pela relevância de cada tupla, de acordo com a percepção dos participantes da *survey*.

Das 28 tuplas identificadas como relevantes na análise do sistema *ProxGer* para pelo menos 6 participantes da pesquisa, a regra que apresenta maior freqüência (gera 6 tuplas das 28 consideradas) é a regra que interpreta a estrutura de frase nome(s) + verbo(s) + (adjetivo opcional) nome(s). Embora existam 6 ocorrências de aceitação de tuplas formadas pela mesma regra, isto equivale a apenas 21% da amostra, o que não caracteriza melhor formação de uma regra. O mesmo ocorre com as tuplas indicadas com maior relevância pelos participantes da pesquisa para o sistema *Time Card*, pois foram formadas por regras diversas.

Das 28 tuplas identificadas como relevantes para o sistema *ProxGer*, duas delas possuem 100% de relevância e 21% das tuplas possuem 90% de relevância de acordo com as respostas obtidas através da *survey*. Das tuplas identificadas relevantes a partir dos detalhamentos de casos de uso do sistema *Time Card*, 4 delas foram consideradas 100% relevantes. Do total de tuplas geradas automaticamente, para o sistema *ProxGer* apenas 1 tupla foi considerada irrelevante (0%). Já para o sistema *Time Card*, 7 tuplas foram consideradas irrelevantes.

As tuplas que possuem o relacionamento *can_be*, formadas a partir da identificação de um adjetivo que qualifica um substantivo, denotam (de acordo com esta

abordagem) um estado que o conceito pode ou não apresentar. Para exemplificar, a tupla (sale, can_be, new) determina que uma venda possa ser uma nova venda, embora o conceito venda não represente necessariamente uma nova venda. Das 8 tuplas que sugerem o relacionamento *can_be* para o refinamento do sistema *ProxGer*, apenas 1 destas tuplas foi considerada relevante. Para o sistema *Time Card* também foi identificado apenas 1 tupla relevante que representa o relacionamento *can_be*, apesar de serem sugeridas 25 tuplas com este relacionamento.

Muitos conceitos e relacionamentos novos foram identificados nas tuplas consideradas relevantes. Portanto, é possível concluir que através da abordagem descrita neste trabalho (apoiada pela ferramenta *OntSoft*) foi possível enriquecer a ontologia inicial, através da identificação automática de novas tuplas que representam os conceitos e relacionamentos relevantes para a solução de software em desenvolvimento. Assim, com a utilização da ferramenta de apoio *OntSoft*, foi possível desonerar o processo de criação da ontologia, apresentando um bom resultado para os exemplos de uso. A Tabela 10 a seguir apresenta alguns destes novos conceitos e relacionamentos identificados.

Tabela 10 – Novos conceitos e relacionamentos identificados.

Sistema	Novos conceitos	Novos Relacionamentos
ProxGer	*update*	*has_type*
	inventory	*can_be*
	authorization_requests	*asks_for*
	digital	*presents*
	company	*wants*
	price	*enters*
Time Card	*administrative*	*can_be*
	password	*change*
	administrator	*adds*
	hours	*submits*
	employee	*exports*

Para o sistema *ProxGer*, através da aplicação da *survey* foram identificados 86% de novos conceitos relevantes, que não existiam na ontologia inicial. Todos os relacionamentos identificados para refinar e enriquecer a ontologia não existiam na ontologia inicial do sistema *ProxGer*. Para o sistema *Time Card*, foram identificados 80% de novos conceitos, ou seja, dos conceitos identificados como relevantes, 80% deles não

estavam mapeados na ontologia inicial. Todos os relacionamentos identificados para refinar e enriquecer a ontologia, também não constava na ontologia inicial.

As tuplas que foram identificadas não sofreram nenhum tipo de análise de relevância automática antes da validação humana realizada através da *survey*. Algum critério, como o estudo de um “ponto de corte”, seria muito útil e interessante se fosse aplicado antes da sugestão da tupla para o usuário final realizar a validação, pois traria a melhoria da taxa de acertos. Embora seja possível e importante a melhoria da taxa de acerto, os resultados obtidos através da abordagem proposta podem ser considerados satisfatórios a partir da análise da aplicação da *survey*. Isto ocorre porque a abordagem alcançou o objetivo principal de auxiliar o refinamento da ontologia, minimizando a intervenção humana para a sua construção.

7. CONCLUSÃO

A extração de termos conceitualmente relevantes em um determinado domínio a partir de um corpus em linguagem natural não é um processo simples [LOP10]. A linguagem natural, por ser suscetível a erros e ambigüidades, dificulta o processamento e a interpretação do texto. Termos compostos, que correspondem a mais de duas unidades lexicais (tratados no texto como termos *n* gramas) representam outra dificuldade para extração automática.

A identificação de um ou mais termos possíveis candidatos a conceitos, assim como a definição de uma métrica que defina quando um ou *n* termos representam semanticamente um conceito para o domínio da aplicação proposta não é trivial. Neste trabalho a extração dos termos e relacionamentos foi realizada através de uma abordagem lingüística. A abordagem lingüística utiliza-se de anotações lingüísticas, ou seja, identificação dos termos a partir de suas características morfológicas, sintáticas e semânticas.

No domínio de ES, o objetivo é extrair tuplas que expressem o relacionamento entre conceitos com a finalidade de refinar e enriquecer uma ontologia inicial, muito útil por servir de modelo para os demais relacionamentos posteriormente identificados, modelada a partir da definição de um Modelo de Domínio. A extração destas tuplas ocorreu a partir da descrição detalhada de casos de uso. Os detalhamentos de casos de uso geralmente não possuem volume de texto relativamente grande para aplicar técnicas estatísticas. Embora não possua grande volume de informação textual, os detalhamentos de casos de uso são escritos de forma estruturada, seguindo *templates* que especificam campos para nome, fluxo principal, alternativo entre outros. Se por um lado um texto escrito em linguagem natural apresenta problemas inerentes à linguagem livre, pode-se afirmar que um texto escrito na seção do fluxo principal de um caso de uso descreve uma ação. Esta característica torna o problema de extração de conceitos e relacionamentos um pouco mais viável se comparado à extração de termos em textos livres.

A extração da estrutura (conceito, relacionamento, conceito) teve a finalidade de enriquecer e refinar a ontologia inicial. No entanto, o processo de aprendizagem da ontologia envolve o cumprimento das atividades de aquisição de terminologia, identificação dos termos que são sinônimos, formação de conceitos, taxonomia do conhecimento, aprendizagem de relacionamentos entre conceitos, propriedades ou atributos, taxonomia dos relacionamentos, instanciação e definição dos axiomas [CIM06].

Maedche [MAE02] apresenta a aprendizagem da ontologia como um processo que envolve a definição de regras ou reutilização de uma ontologia, inclusão ou manutenção de novos domínios, corte de conceitos irrelevantes (com a finalidade de representar melhor o domínio) e refinamento da ontologia.

De acordo com o Cimiano [CIM06], para que o processo de aprendizagem da ontologia esteja completo é necessária a satisfação de todas as atividades descritas. Neste trabalho, a abordagem previu a geração de uma ontologia inicial, modelada a partir do diagrama de classes que descreve o modelo de domínio. Imediatamente após, a abordagem propôs o refinamento e enriquecimento através da descrição textual dos detalhamentos de casos de uso que detalham a aplicação de software. A partir das informações que servem de insumo, foi possível a identificação de conceitos e relacionamentos, porém, a relevância destes conceitos e relacionamentos, a identificação de instancias e axiomas (regras que são sempre verdadeiras) não são satisfeitas na abordagem apresentada neste trabalho.

Uma *survey* foi aplicada nesta pesquisa com a finalidade de avaliar as tuplas geradas automaticamente pela ferramenta. A abordagem descrita neste trabalho ainda necessita da validação humana para indicação da relevância dos conceitos e relacionamentos propostos para o refinamento da ontologia. Embora ainda persista a necessidade da intervenção humana para o refinamento da ontologia, é possível afirmar que a abordagem alcançou resultados satisfatórios, pois através da análise da *survey*, foi identificado 50% de acerto para as tuplas sugeridas para o refinamento da ontologia inicial do exemplo de uso do sistema *ProxGer* [LAR07]. Uma limitação apresentada neste trabalho é a necessidade do modelo de domínio e dos detalhamentos de casos de uso estar escrito em língua inglesa.

7.1 Trabalhos Futuros

Para continuidade deste trabalho é sugerido o estudo aprofundado da aplicação de técnicas estatísticas combinadas, para estabelecer uma métrica de relevância para as tuplas geradas. Para a utilização das técnicas estatísticas, é possível realizar a análise sobre toda documentação e outros diagramas do sistema. No trabalho de [LOP10] é sugerido o cálculo da freqüência relativa dos termos, realizado através da divisão do número de vezes que o termo é extraído pelo número total de ocorrências do termo. A autora [LOP10] ainda sugere um ponto de corte entre 5% e 10%. Para os exemplos de

uso utilizados nesta dissertação, foi observado que a frequencia da ocorrência dos termos não determina a relevância do mapeamento destes termos em conceitos na ontologia que descreve a solução de software. Identificar uma medida de relevância é necessário para minimizar a taxa de erros da abordagem apresentada neste trabalho.

Testes exaustivos, aplicados em diferentes descrições detalhadas de casos de uso, de diversas modelagens de sistemas, é um trabalho interessante de ser realizado. A dificuldade em executar testes com outros exemplos está no fato de não existirem muitos exemplos completos de modelagem de sistemas na literatura para realização dos testes. Outro trabalho interessante, e talvez a maior lacuna a ser preenchida na viabilização da automação da geração de ontologias no processo de desenvolvimento, é a identificação da relevância dos relacionamentos importantes dentro do domínio da aplicação modelada, e que devem fazer parte da ontologia que descreve o sistema em desenvolvimento. Para que isto seja possível, um estudo sobre a possibilidade de se estabelecer um modelo único de descrição de caso de uso também é válido. O autor Somé [SOM07] afirma a necessidade de expressar restrições de seqüência nos casos de uso. A importante contribuição do autor é a proposta de uma definição da semântica formal para a representação formal dos casos de uso (instruções sobre a forma da linguagem natural) e a definição da estrutura (seções) do documento [SOM11]. Esta pesquisa ainda comporta muitos estudos, como a generalização desta abordagem para a língua portuguesa, com a criação de novas regras que possibilitem a identificação de tuplas, considerando as categorias morfossintáticas pertinentes à linguagem.

Por fim, através da abordagem proposta nesta dissertação, tornou-se viável a construção e o refinamento da ontologia que descreve o sistema de software em construção, possibilitando a aquisição e o compartilhamento do conhecimento através da utilização de ontologias no desenvolvimento de software.

REFERÊNCIAS BIBLIOGRÁFICAS

[ALP04] Alpaydin, E. “Introduction to Machine Learning”. MIT Press, 2004, 415p.

[ARL05] Arlow, J.; Neustadt, I. “UML 2 and the Unified Process: practical object-oriented analysis an design”. Addison-Wesley, 2005, 2nd ed. 592p.

[ARR01] Arrington, C. T. “Enterprise Java with UML”. John Wiley & Sons, 2001, 480p.

[ASS10] Assawamekin, N.; Sunetnanta, T.; Pluempitiwiriyawej, C. “Ontology-based Multiperspective Requirements Traceability Framework”. *Journal Knowledge and Information Systems*, vol. 25 – 3, 2010, pp. 493-522.

[BAS99] Basili, V. R.; Shull, F.; Lanubile, F. “Building Knowledge through Families of Experiments”. *IEEE Transactions on Software Engineering*, vol. 25 - 4, Jul-Ago 1999, pp. 456-473.

[BIT06] Bittencourt, G. “Inteligência Artificial: Ferramentas e Teorias”. UFSC, Brasil, 2006, 371p.

[BOO06] Booch, G.; Rumbaugh; J.; Jacobson, I. “UML Guia do Usuário”. Editora Campus, 2006, 474p.

[CHE07] Chen, C.; Mattews, M. "Extending Description Logic for Reasoning about Ontology Evolution". *IEE/WIC/ACM International Conference on Web Intelligence*, 2007, pp. 452-456.

[CIM06] Cimiano, P. “Ontology Learning and Population from Text: Algorithms, Evaluation and Applications”. USA: Springer Science, 2006, 347p.

[COC01] Cockburn, A. “Escrevendo Casos de Uso Eficazes: Um guia prático para desenvolvedores de software”. Brasil: Bookman, 2001, 254p.

[DAC03] Daconta, M. C.; Obrst, L. J.; Smith, K. T. “The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management”. USA: Wiley Publishing , 2003, 281p.

[ENK07] Enkhsaikhan, M.; Wong, W.; Liu, W.; Reynolds, M. "Measuring Data-Driven Ontology Changes using Text Mining". *Australasian Data Mining Conference (AusDM’07)*, 2007, pp 39-46.

[FOR01] Forrest, S.; Carver, J.; Travassos, G. “An Empirical Methodology for Introducing Software Processes”. *ESEC/FSE*, 2001, pp. 288-296.

[GRU93] Gruber, T. R. “Towards Principles for the Design of Ontologies Used for Knowledge Sharing”. *International Journal of Human and Computer Studies*, 1993, vol. 43, pp. 907-928.

[GUA97] Guarino, N. “Understanding, Building and Using Ontologies”. *International Journal of Human-Computer Studies*, vol. 46, Issue 2-3, 1997, pp. 293-310.

[HAN01] Han, J.; Kamber, M. "Data Mining: Concepts and Techniques". USA: Morgan Kaufmann Publishers, 2001, 770p.

[HAR03] Harmelen, F. V.; Davies, J.; Fensel, D. "Towards the Semantic Web: Ontology-Driven Knowledge Management". England: John Wiley & Sons Ltda, 2003, 288p.

[HEN01] Hendler, J. "Agents and the Semantic Web". *IEEE Intelligent Systems*, 2001, pp. 30-37.

[JAC99] Jacobson, I.; Booch, G.; Rumbaugh, J. “The Unified Software Development Process”. Addison-Wesley, 1999, 463p.

[JEN10] Jena – A Semantic Web Framework for Java. Desenvolvido por HP Labs Semantic Web Programme. Capturado em: http://jena.sourceforge.net/ index.html, Fevereiro de 2010.

[LAR07] Larman, C. “Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design”. Prentice Hall, 2007, 507p.

[LOO06] Loos, B. "Scaling Natural Language Understanding via User-driven Ontology Learning". In: 3rd Workshop on Scalable Natural Language Understanding, New York City, USA, 2006, pp. 33-40.

[LOP10] Lopes, L.; Vieira, R.; Finatto, M.J.; Martins, D. “Extracting compound terms from domain corpora”. *Journal of the Brazilian Computer Society*, vol. 16 – 4, 2010, pp. 247-259.

[MAE01] Maedche, A.; Staab, S. “Ontology Learning for the Semantic Web”. *IEEE Intelligent Systems*, 2001, pp. 72-79.

[MAE02] Maedche, A. “Ontology Learning for the Semantic Web”. Kluwer Academic Publishers, 2002, 244p.

[MAE03] Maedche, A.; Motik, B.; Stojanovic; L.; Studer, R.; Volz, R. "An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies". *ACM Proceedings of the 12th international conference on World Wide Web*, 2003, pp. 439-448.

[MAR08] Marneffe, M. C.; Manning, C. D. “Stanford Typed Dependencies Manual”. 2008, 20p.

[MIL95] Miller, G. A. “WordNet: a lexical database for English”. *Communications of the ACM*, 1995, vol. 38, Issue 11. pp. 39-41.

[NAT10] Natural Language Processing the Group. Capturado em: http://nlp.stanford.edu/software/lex-parser.shtml, Junho de 2010.

[NAT10a] Natural Language Processing the Group. Capturado em: http://nlp.stanford.edu/software/parser-faq.shtml, Junho de 2010.

[NIC09] Nicola, A.; Missikoff, M.; Navigli, R. “A Software Engineering Approach to Ontology Building”. *Information Systems*. 2009, 34, pp. 258-275.

[NOL07] Noll, R. P. "Rastreabilidade Ontológica sobre o Processo Unificado", Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2007, 120p.

[NOL07a] Noll, R. P.; Ribeiro, M. B. "Ontological Traceability over the Unified Process". In: 14th Annual *IEEE* International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), USA, 2007, pp. 249-255.

[PER04] Péres, A. G.; López, M. F.; Corcho, O. "Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web". London: Springer-Verlag, 2004, 403p.

[SHU02] Shull, F.; Basili, V.; Carver, J.; Maldonado, J.; Travassos, G.; Mendonça, M.; Fabbri, S. “Replicating Software Engineering Experiments: Addressing the Tacit Knowledge Problem”. *International Symposium on Empirical Software Engineering (ISESE’02)*, 2002, 10p.

[SHU08] Shull, F.; Singer, J.; Sjoberg, D. I. K. “Guide To Advanced Empirical Software Engineering”. London: Springer Verlag, 2008, 394p.

[SJO07] Sjoberg, D. I. K.; Duba, T.; Jorgensen, M. “The Future of Empirical Methods in Software Engineering Research”. *Future of Software Engineering (FOSE’07)*, 2007, 21p.

[SOM07] Somé, S. S. “Specifying Use Case Sequencing Constraints using Description Elements”. *IEEE Sixth International Workshop on Scenarios and State Machines (SCESM'07)*, May 2007, 7p.

[SOM08] Sommerville, I. “Software Engineering, eighth edition”. São Paulo: Pearson Addison Wesley, 2008, 552p.

[SOM11] Somé, S. S. “Petri Nets Based Formalization of Textual Use Cases”. Capturado em: http://www.site.uottawa.ca/eng/school/publications/techrep/2007/TR-2007-11.pdf, Jan 2011.

[STO03] Stojanovic, L.; Maedche, A.; Stojanovic, N.; Studer, R. "Ontology Evolution as Reconfiguration-Design Problem Solving". *K-CAP*, Sanibel, Island, 2003, pp. 162-171.

[STO04] Stojanovic, L. "Methods and Tools for Ontology Evolution". Tese de Doutorado, University of Karlsruhe, 2004, 249p.

[SUN08] Sung, S.; Chung, S.; McLeod, D. "Efficient Concept Clustering for Ontology Learning using an Event Life Cycle on the Web". *Proceedings of the 2008 ACM Symposium on Applied Computing*, 2008, pp. 2310-2314.

[TAL08] Talevski, A.; Wongthongtham, P.; Komchaliaw, S. "Towards a Software Component Ontology". *AIIDE Proceedings of iiWAS*, 2008, pp. 503-507.

[TAN06] Tan, P.; Steinbach, M.; Kumar, V. "Introduction to Data Mining". USA: Person Education, 2006, 769p.

[TOU10] Toutanova, K.; Klein, D.; Manning, C. D.; Singer, Y. “Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network”. *NAACL ’03 Proceedings of the 2003 Conference of the North American Chapter of the Association for Computacional Linguistics on Human Language Technology*, vol. 1, 2003, pp. 173-180.

[WON06] Wongthongthaml, P.; Chang, E.; Dillon, T. "Enhancing Software Engineering Project Information through Software Engineering Ontology Instantiations". *WI’06 International Conference on Web Intelligence*, 2006, 5p.

[WON07] Wongthongthaml, P.; Chang, E.; Dillon, T. "Ontology Modeling Notations for Software Engineering Knowledge Representation". *IEEE DEST 2007 International Conference on Digital Ecosystems an Technologies*, 2007, pp. 339-345.

[WOR10] WordNet – A lexical database for English. Capturado em: http://wordnet.princeton.edu, Junho de 2010.

[W3C10] W3C. “World Wide Web Consortium”. Capturado em: http://www.w3.org/TR/owl-features, Junho de 2010.

[YAN08] Yang, H.; Callan, J. "Metric-Based Ontology Learning". *ONISW’08*, California, USA, 2008, pp. 01-08.

[ZAM02] Zambenedetti, C. "Extração de Informação sobre Base de Dados Textuais". Dissertação de Mestrado, Programa de Pós-Graduação em Computação, UFRGS, 2002, 144p.

APÊNDICE A - TUPLAS IDENTIFICADAS

A descrição detalhada do caso de uso *process sale* [LAR07] foi utilizada como modelo para definição das regras iniciais, e geram tuplas para refinar e enriquecer a ontologia inicial. A extração dos conceitos (*C*) e relacionamentos (*R*) contidos no texto parte da identificação da definição formal simplificada da ontologia {*C, R, A, T*} com as informações contidas na especificação detalhada dos casos de uso, descritos em linguagem natural. O resultado são tuplas que representam (conceito, relacionamento, conceito), conforme apresenta a Tabela abaixo.

Tuplas identificadas: Caso de uso *Process Sale.*

Sentença Original	Sentença Pré-processada	Tupla(s) da Ontologia
Cashier: wants accurate, fast entry, and no payment errors, as cash drawer short ages are deducted from his/her salary.	Cashier wants accurate.	(Cashier,wants,accurate)
Salesperson: wants sales commissions updated.	Salesperson wants sales commissions updated.	(Salesperson,wants,sales_commissions_updated)
Customer: Wants purchase and fast service with minimal effort.	Customer Wants purchase and fast service with minimal effort.	(Customer,Wants,purchase) (Customer,Wants,service) (service,can_be,fast) (effort,can_be,minimal)
Wants easily visible display of entered items and prices.	Customer Wants easily visible display of entered items and prices.	(Customer,Wants,easily_display_entered_items) (Customer,Wants,prices) (easily_display_entered_items,can_be,visible)
Wants proof of purchase to support returns.	Customer Wants proof of purchase to support returns.	(Customer,Wants,proof) (proof,support,returns)
Company: Wants to accurately record transactions and satisfy customer interests.	Company Wants to accurately record transactions and satisfy customer interests.	(Company,Wants_accurately_record,transactions) (Company,satisfy,customer_interests)
Wants some fault tolerance to allow sales capture even if server components (e.g., remote credit validation) are unavailable.	Company Wants some fault tolerance to allow sales capture even if server components.	(Company,Wants,fault_tolerance) (fault_tolerance,allow,sales) (sales,capture,even_server_components)
Wants automatic and fast update of accounting and inventory.	Company Wants automatic and fast update of accounting and inventory.	(Company,Wants,automatic_update) (Company,Wants,fast_update)

		(update,has_type,accounting)
		(update,has_type,inventory)
Manager: Wants to be able to quickly perform override operations, and easily debug Cashier problems.	Manager Wants to be able to quickly perform override operations.	(Manager,Wants_to_be,able)
		(able,quickly_perform,override_operations)
Government Tax Agencies: Want to collect tax from every sale.	Government Tax Agencies Want to collect tax from every sale.	(Government_Tax_Agencies,Want_to_collect,tax_sale)
Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol.	Payment Authorization Service Wants to receive digital authorization requests in the correct format and protocol.	(Payment_Authorization_Service,Wants_to_receive,authorization_requests)
		(Payment_Authorization_Service,Wants_to_receive,protocol)
		(Payment_Authorization_Service,Wants_to_receive,format)
		(authorization_requests,can_be,digital)
		(format,can_be,correct)
		(protocol,can_be,correct)
Wants to accurately account for their payables to the store.	Payment Authorization Service Wants to accurately account for payables to the store.	(Payment_Authorization_Service,Wants_to_accurately_account_for,payables)
Cashier is identified and authenticated.	Cashier is identified and authenticated.	(Cashier,is,identified)
		(Cashier,is,authenticated)
Sale is saved.	Sale is saved.	(Sale,is,saved)
Tax is correctly calculated.	Tax is correctly calculated.	(Tax,is,correctly_calculated)
Accounting and Inventory are updated.	Accounting and Inventory are updated.	(Accounting,are,updated)
		(Inventory,are,updated)
Commissions recorded.	Commissions recorded.	(Commissions,do,record)
Receipt is generated.	Receipt is generated.	(Receipt,is,generated)
Payment authorization approvals are recorded.	Payment authorization approvals are recorded.	(Payment_authorization_approvals,are,recorded)
2. Cashier starts a new sale.	Cashier starts a new sale.	(Cashier,starts,sale)
		(sale,can_be,new)
3. Cashier enters item identifier.	Cashier enters item identifier.	(Cashier,enters,item_identifier)
4. System records sale line item and presents item description, price, and running total.	System records sale line item and presents item description.	(System,records,sale_line_item)
		(System,presents,item_description)
4. System records sale line	System presents price.	(System,presents,price)

item and presents item description, price, and running total.		
4. System records sale line item and presents item description, price, and running total.	System running total.	(System,running,total)
Price calculated from a set of price rules.	Price calculated from a set of price rules.	(Price,calculated_from,set_rules)
5. System presents total with taxes calculated.	System presents total with taxes calculated.	(System,presents,total)
		(total,can_be,taxes_calculated)
6. Cashier tells Customer the total, and asks for payment.	Cashier tells Customer the total.	(Cashier,tells,Customer_total)
6. Cashier tells Customer the total, and asks for payment.	Cashier asks for payment.	(Cashier,asks_for,payment)
7. Customer pays and System handles payment.	Customer pays and System handles payment.	(Customer,do,pays)
		(System,handles,payment)
9. System presents receipt.	System presents receipt.	(System,presents,receipt)
10. Customer leaves with receipt and goods (if any).	Customer leaves with receipt and goods.	(Customer,leaves_with,receipt)
		(Customer,leaves_with,goods)
*a. At any time, Manager requests an override operation:	Manager requests an override operation.	(Manager,requests,override_operation)
1. System enters Manager-authorized mode.	System enters Manager-authorized mode.	(System,enters,mode)
		(mode,can_be,Manager-authorized)
2. Manager or Cashier performs one Manager-mode operation. e.g., cash balance change, resume a suspended sale on another register, void a sale, etc.	Manager or Cashier performs one Manager-mode operation.	(Manager,performs,operation)
		(Cashier,performs,operation)
		(operation,can_be,Manager-mode)
3. System reverts to Cashier-authorized mode.	System reverts to Cashier-authorized mode.	(System,reverts,Cashier-authorized_mode)
1. Cashier restarts System, logs in, and requests recovery of prior state.	Cashier restarts System.	(Cashier,restarts,System)
1. Cashier restarts System, logs in, and requests recovery of prior state.	Cashier logs System.	(Cashier,logs,System)
1. Cashier restarts System, logs in, and requests recovery of prior state.	Cashier requests recovery of prior state.	(Cashier,requests,recovery)
		(state,can_be,prior)
2. System reconstructs prior state.	System reconstructs prior state.	(System,reconstructs,prior_state)
2a. System detects anomalies preventing recovery:	System detects anomalies preventing recovery.	(System,detects,anomalies_preventing_recovery)
1. System signals error to the Cashier, records the error, and enters a clean state.	System signals error to the Cashier.	(System,signals,error)

1. System signals error to the Cashier, records the error, and enters a clean state.	System records the error.	(System,records,error)
1. System signals error to the Cashier, records the error, and enters a clean state.	System enters a clean state.	(System,enters,state)
		(state,can_be,clean)
2. Cashier starts a new sale.	Cashier starts a new sale.	(Cashier,starts,sale)
		(sale,can_be,new)
1a. Customer or Manager indicate to resume a suspended sale.	Customer or Manager indicate to resume a suspended sale.	(Customer,indicate_to_resume,suspended_sale)
		(Manager,indicate_to_resume,suspended_sale)
1. Cashier performs resume operation, and enters the ID to retrieve the sale.	Cashier performs resume operation.	(Cashier,performs_resume,operation)
1. Cashier performs resume operation, and enters the ID to retrieve the sale.	Cashier enters the ID to retrieve the sale.	(Cashier,enters,ID)
		(ID,retrieve,sale)
2. System displays the state of the resumed sale, with subtotal.	System displays the state of the resumed sale.	(System_displays_state,resumed,sale)
1. System signals error to the Cashier.	System signals error to the Cashier.	(System,signals,error)
2. Cashier probably starts new sale and re-enters all items.	Cashier probably starts new sale and re-enters all items.	(Cashier,probably_starts,sale)
		(Cashier,re-enters,items)
		(Cashier,probably_starts,items)
		(sale,can_be,new)
3. Cashier continues with sale (probably entering more items or handling payment).	Cashier continues with sale.	(Cashier,continues_with,sale)
2-4a. Customer tells Cashier they have a tax-exempt status (e.g., seniors, native peoples).	Customer tells Cashier have a tax-exempt status.	(Customer,tells,Cashier)
		(Cashier,have,status)
		(status,can_be,tax-exempt)
1. Cashier verifies, and then enters tax-exempt status code.	Cashier verifies.	(Cashier,do,verify)
1. Cashier verifies, and then enters tax-exempt status code.	Cashier enters tax-exempt status code.	(Cashier,enters,status_code)
		(status_code,can_be,tax-exempt)
2. System records status (which it will use during tax calculations).	System records status.	(System,records,status)
1. System signals error and rejects entry.	System signals error and rejects entry.	(System,signals,error)
		(System,rejects,entry)
2. Cashier responds to the	Cashier responds to the error.	(Cashier,responds,error)

error:		
2a. There is a human-readable item ID (e.g., a numeric UPC):	There is a human-readable item ID.	(System,has,item_ID) (item_ID,can_be,human-readable)
1. Cashier manually enters the item ID.	Cashier manually enters the item ID.	(Cashier,manually_enters,item_ID)
2. System displays description and price.	System displays description and price.	(System,displays,description) (System,displays,price)
2a. Invalid item ID: System signals error.	Invalid item ID System signals error.	(Invalid_item_ID_System,signals,error)
Cashier tries alternate method.	Cashier tries alternate method.	(Cashier,tries,method) (method,can_be,alternate)
2b. There is no item ID, but there is a price on the tag:	There is no item ID.	(System,has_no,item_ID)
2b. There is no item ID, but there is a price on the tag:	there is a price on the tag.	(System,has,price_tag)
1. Cashier asks Manager to perform an override operation.	Cashier asks Manager to perform an override operation.	(Cashier,asks,Manager) (Manager,perform,override_operation)
2. Managers performs override.	Managers performs override.	(Managers,performs,override)
3. Cashier indicates manual price entry, enters price, and requests standard taxation for this amount (because there is no product information, the tax engine can't otherwise deduce how to tax it).	Cashier indicates manual price entry.	(Cashier,indicates,price_entry) (price_entry,can_be,manual)
3. Cashier indicates manual price entry, enters price, and requests standard taxation for this amount (because there is no product information, the tax engine can't otherwise deduce how to tax it).	Cashier enters price.	(Cashier,enters,price)
2d. Otherwise, Cashier asks an employee for the true item ID or price, and does either manual ID or manual price entry (see above).	Cashier asks an employee for the true item ID or price.	(Cashier,asks,employee) (Cashier,asks,price) (Cashier,asks,item_ID) (item_ID,can_be,true) (price,can_be,true)
1. Cashier can enter item category identifier and the quantity.	Cashier can enter item category identifier and the quantity.	(Cashier,can_enter,item_category_identifier) (Cashier,can_enter,quantity)
3c. Item requires manual category and price entry (such as flowers or cards with a price on them):	Item requires manual category and price entry.	(Item,requires,category) (Item,requires,price_entry) (category,can_be,manual)

1. Cashier enters special manual category code, plus the price.	Cashier enters special manual category code.	(Cashier,enters,manual_category_code)
		(manual_category_code,can_be,special)
This is only legal if the item value is less than the void limit for Cashiers, otherwise a Manager override is needed.	a Manager override is needed.	(Manager_override,is,needed)
1. Cashier enters item identifier for removal from sale.	Cashier enters item identifier for removal from sale.	(Cashier,enters,item_identifier_for_removal)
		(item_identifier_for_removal,is_from,sale)
2. System removes item and displays updated running total.	System removes item and displays updated running total.	(System,removes,item)
		(displays,updated_running,total)
2a. Item price exceeds void limit for Cashiers:	Item price exceeds void limit for Cashiers.	(Item_price,exceeds,limit_for_Cashiers)
		(limit_for_Cashiers,can_be,void)
1. System signals error, and suggests Manager override.	System signals error.	(System,signals,error)
1. System signals error, and suggests Manager override.	System suggests Manager override.	(System,suggests,Manager_override)
2. Cashier requests Manager override, gets it, and repeats operation.	Cashier requests Manager override.	(Cashier,requests,Manager_override)
3-6b. Customer tells Cashier to cancel sale:	Customer tells Cashier to cancel sale.	(Customer,tells,Cashier)
		(Cashier,cancel,sale)
1. Cashier cancels sale on System.	Cashier cancels sale on System.	(Cashier,cancels,sale)
3-6c. Cashier suspends the sale:	Cashier suspends the sale.	(Cashier,suspends,sale)
1. System records sale so that it is available for retrieval on any POS register.	System records sale so that is available for retrieval on any POS register.	(System_records_sale,is,retrieval_POS_register)
		(retrieval_POS_register,can_be,available)
1. Cashier requests approval from Manager.	Cashier requests approval from Manager.	(Cashier,requests,approval)
		(approval,requests_from,Manager)
2. Manager performs override operation.	Manager performs override operation.	(Manager,performs,override_operation)
3. Cashier enters manual override price.	Cashier enters manual override price.	(Cashier,enters,override_price)
		(override_price,can_be,manual)
4. System presents new price.	System presents new price.	(System,presents,price)
		(price,can_be,new)
5a. System detects failure to communicate with external tax calculation system service:	System detects failure to communicate with external tax calculation system service.	(System,detects,failure)
		(failure,communicate_with,tax_calculation_system_service)
		(tax_calculation_system_service,can_be,external)

1. System restarts the service on the POS node, and continues.	System restarts the service on the POS node.	(System,restarts,service_POS_node)
1. System signals error.	System signals error.	(System,signals,error)
2. Cashier may manually calculate and enter the tax, or cancel the sale.	Cashier cancel the sale.	(Cashier,cancel,sale)
5b. Customer says they are eligible for a discount (e.g., employee, preferred customer):	Customer says are eligible for a discount.	(Customer,says_are,discount)
		(discount,can_be,eligible)
1. Cashier signals discount request.	Cashier signals discount request.	(Cashier,signals,discount_request)
2. Cashier enters Customer identification.	Cashier enters Customer identification.	(Cashier,enters,Customer_identification)
3. System presents discount total, based on discount rules.	System presents discount total.	(System,presents,discount_total)
3. System presents discount total, based on discount rules.	System based on discount rules.	(System,based_on,discount_rules)
5c. Customer says they have credit in their account, to apply to the sale:	Customer says have credit in account.	(Customer,says_have,credit)
1. Cashier signals credit request.	Cashier signals credit request.	(Cashier,signals,credit_request)
2. Cashier enters Customer identification.	Cashier enters Customer identification.	(Cashier,enters,Customer_identification)
3. Systems applies credit up to price=0, and reduces remaining credit.	Systems reduces remaining credit.	(Systems,reduces,remaining_credit)
6a. Customer says they intended to pay by cash but don't have enough cash:	Customer says intended to pay by cash but do n't have enough cash.	(Customer,says_intended_to_pay,cash)
		(Customer,do_n't_have,cash)
		(cash,can_be,enough)
1. Cashier asks for alternate payment method.	Cashier asks for alternate payment method.	(Cashier,asks_for,payment_method)
		(payment_method,can_be,alternate)
1a. Customer tells Cashier to cancel sale.	Customer tells Cashier to cancel sale.	(Customer,tells,Cashier)
		(Cashier,cancel,sale)
Cashier cancels sale on System.	Cashier cancels sale on System.	(Cashier,cancels,sale)
1. Cashier enters the cash amount tendered.	Cashier enters the cash amount tendered.	(Cashier,enters,cash_amount_tendered)
2. System presents the balance due, and releases the cash drawer.	System releases the cash drawer.	(System,releases,cash_drawer)
4. System records the cash payment.	System records the cash payment.	(System,records,cash_payment)
7b. Paying by credit:	System records the cash payment Paying by credit.	(System_records_cash_payment,Paying,by_credit)
1. Customer enters their credit account information.	Customer enters credit account information.	(Customer,enters,credit_account_information)
2. System displays their	System displays payment for	(System,displays,payment_verification)

payment for verification.	verification.	
3. Cashier confirms.	Cashier confirms.	(Cashier,do,confirm)
3a. Cashier cancels payment step:	Cashier cancels payment step.	(Cashier,cancels,payment_step)
1. System reverts to "item entry" mode.	System reverts to item entry mode.	(System,reverts,item_entry_mode)
4. System sends payment authorization request to an external Payment Authorization Service System, and requests payment approval.	System sends payment authorization request to an external Payment Authorization Service System.	(System,sends,payment_authorization_request)
4. System sends payment authorization request to an external Payment Authorization Service System, and requests payment approval.	System requests payment approval.	(System,requests,payment_approval)
4a. System detects failure to collaborate with external system:	System detects failure to collaborate with external system.	(System,detects,failure) (failure,collaborate_with,system) (system,can_be,external)
1. System signals error to Cashier.	System signals error to Cashier.	(System,signals,error)
2. Cashier asks Customer for alternate payment.	Cashier asks Customer for alternate payment.	(Cashier,asks,Customer_payment) (Customer_payment,can_be,alternate)
5.System receives payment approval, signals approval to Cashier, and releases cash drawer (to insert signed credit payment receipt).	System receives payment approval.	(System,receives,payment_approval)
5.System receives payment approval, signals approval to Cashier, and releases cash drawer (to insert signed credit payment receipt).	System signals approval to Cashier.	(System,signals,approval)
5.System receives payment approval, signals approval to Cashier, and releases cash drawer (to insert signed credit payment receipt).	System releases cash drawer.	(System,releases,cash_drawer)
5a. System receives payment denial:	System receives payment denial.	(System,receives,payment_denial)
1. System signals denial to Cashier.	System signals denial to Cashier.	(System,signals,denial)
2. Cashier asks Customer for alternate payment.	Cashier asks Customer for alternate payment.	(Cashier,asks,Customer_payment) (Customer_payment,can_be,alternate)
5b. Timeout waiting for response.	Timeout waiting for response.	(Timeout,waiting_for,response)
1. System signals timeout to Cashier.	System signals timeout to Cashier.	(System,signals,timeout)
2. Cashier may try again, or ask Customer for alternate payment.	Cashier ask Customer for alternate payment.	(Cashier,ask,Customer_payment) (Customer_payment,can_be,alternate)

6. System records the credit payment, which includes the payment approval.	System records the credit payment.	(System,records,credit_payment)
7. System presents credit payment signature input mechanism.	System presents credit payment signature input mechanism.	(System,presents,credit_payment_signature_input _mechanism)
8. Cashier asks Customer for a credit payment signature.	Cashier asks Customer for a credit payment signature.	(Cashier,asks,Customer_credit_payment_ signature)
Customer enters signature.	Customer enters signature.	(Customer,enters,signature)
7e. Cashier cancels payment step:	Cashier cancels payment step.	(Cashier,cancels,payment_step)
1. System reverts to "item entry" mode.	System reverts to item entry mode.	(System,reverts,item_entry_mode)
7f. Customer presents coupons:	Customer presents coupons.	(Customer,presents,coupons)
System records the used coupons for accounting reasons.	System records the used coupons for accounting reasons.	(System_records,used,coupons_for_accounting_ reasons)
1a. Coupon entered is not for any purchased item:	Coupon entered is not for any purchased item.	(Coupon,entered_is_not_for,purchased_item)
1. System signals error to Cashier.	System signals error to Cashier.	(System,signals,error)
9a. There are product rebates:	There are product rebates.	(System,has,product_rebates)
1. System presents the rebate forms and rebate receipts for each item with a rebate.	System presents the rebate forms and rebate receipts for each item with a rebate.	(System,presents,rebate_forms) (System,presents,rebate_receipts_item_rebate)
9b. Customer requests gift receipt (no prices visible):	Customer requests gift receipt.	(Customer,requests,gift_receipt)
1. Cashier requests gift receipt and System presents it.	Cashier requests gift receipt and System presents.	(Cashier,requests,gift_receipt) (System,presents,gift_receipt)
2. Cashier replaces paper.	Cashier replaces paper.	(Cashier,replaces,paper)
3. Cashier requests another receipt.	Cashier requests another receipt.	(Cashier,requests,receipt)

Dentre a lista de tuplas identificadas, o processamento do detalhamento de caso de uso resultou em algumas tuplas repetidas. A indicação de tuplas repetidas ocorre devido à ocorrência de sentenças iguais no fluxo, ou devido ao fato do pré-processamento gerar sentenças iguais a outras existentes no momento da transformação de um período composto em um período simples. Outra razão da abordagem gerar tuplas repetidas pode ser justificada pelo fato das regras possuírem tratamentos especiais para o caso de sentenças que finalizam com a estrutura **TO + NN**. A tabela a seguir apresenta as tuplas repetidas, identificadas durante o processamento do detalhamento de caso de uso *process sale*.

Identificação de tuplas repetidas: Sistema *ProxGer.*

Tuplas repetidas	Número de repetições
(Cashier,asks,Customer_payment)	1
(Cashier,cancel,sale)	2
(Cashier,cancels,payment_step)	1
(Cashier,cancels,sale)	1
(Cashier,enters,Customer_identification)	1
(Cashier,starts,sale)	1
(Customer,tells,Cashier)	2
(Customer_payment,can_be,alternate)	2
(sale,can_be,new)	2
(System,detects,failure)	1
(System,presents,price)	1
(System,releases,cash_drawer)	1
(System,reverts,item_entry_mode)	1
(System,signals,error)	6

A tabela a seguir apresenta as sentenças não processadas pelas regras implementadas.

Sentenças não-processadas: Sistema *ProxGer.*

Relação de sentenças não-processadas
Wants to ensure that Payment Authorization Service payment receivables are recorded.
May be multiple agencies, such as national, state, and county.
Customer arrives at POS checkout with goods and/or services to purchase.
System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
At any time, System fails:
To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario.
Sale not found.
Invalid item ID (not found in system):
Otherwise, Cashier asks an employee for the true item ID or price, and does either manual ID or manual price entry (see above).
There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers):
Customer asks Cashier to remove (i.e., void) an item from the purchase:
System presents a "suspend receipt" that includes the line items, and a sale ID used to retrieve and resume the sale.
The system supplied item price is not wanted (e.g., Customer complained about something and is offered a lower price):
System detects that the service does not restart.
Paying by cash:
Cashier deposits cash tendered and returns balance in cash to Customer.

If signature on paper receipt, Cashier places receipt in cash drawer and closes it.
Paying by check.
Paying by debit.
Before handling payment, Cashier records each coupon and System reduces price as appropriate. System records the used coupons for accounting reasons.
Printer out of paper.
If System can detect the fault, will signal the problem.

A identificação dos conceitos e relacionamentos para formação das tuplas foi realizada a partir das regras implementadas, utilizando como exemplo a descrição detalhada de caso de uso *process sale*. O próximo passo após a criação das regras foi testá-las em outros detalhamentos de casos de uso. Sendo assim, foram utilizados os detalhamentos de casos de uso do sistema *Time Card* [ARR01], com a finalidade de testar a cobertura das regras. Nas tabelas a seguir, estão descritas as tuplas resultantes do processamento dos detalhamentos de casos de uso do sistema *Time Card*, através da abordagem proposta.

Tuplas identificadas: Caso de uso *Change Password.*

Sentença Original	**Sentença Pré-processada**	**Tupla(s) da Ontologia**
The Change Password use case allows employees and administrative users to change their password.	The Change Password use case allows employees and administrative users to change password.	(Change_Password_use_case,allows,employees)
		(users,change,password)
		(users,can_be,administrative)
The user must have logged in to the system.	The user must have logged in to the system.	(user,must_have,logged_system)
Employees must be able to log in from any computer, including home, client sites, and on the road.	Employees must be able to log in from any computer.	(Employees,must_be,able)
		(able,log,computer)
Employees must be able to log in from any computer, including home, client sites, and on the road.	Employees including home.	(Employees,including,home)
This access may be from behind a client's firewall.	This access may be from behind a client firewall.	(access,may_be_from,client_firewall)
The user is notified that his or her password has been changed.	The user is notified that or password has been changed.	(user,is,notified)
		(password,has,been_changed)
The failure is logged by the system.	The failure is logged by the system.	(failure,is,logged_system)
The user is allowed to try again indefinitely.	The user is allowed to try again indefinitely.	(user,is_allowed_to_try,again_indefinitely)
New passwords do not match.	New passwords do not match.	(New_passwords,do,not)
System is unable to store new	System is unable to store new	(System,is_unable_to_store,password)

password due to a system or communications error.	password due to a system or communications error.	
		(System,communications,error)
		(System,is_unable_to_store,error)
		(password,can_be,new)
		(system,can_be,due)
		(error,can_be,due)
The user is notified of the error, complete with any available details.	The user is notified of the error.	(user,is,notified_error)

Tuplas identificadas: Caso de uso *Create Charge Code.*

Sentença Original	**Sentença Pré-processada**	**Tupla(s) da Ontologia**
The user must be logged in as an administrative user.	The user must be logged in as an administrative user.	(user,must_be,logged_user)
		(logged_user,can_be,administrative)
The administrator sees a view of existing charge codes for a selected project.	The administrator sees a view of existing charge codes for a selected project.	(administrator,sees,view_existing_charge_codes_selected_ project)
The new charge code appears in the view and may be used by employees.	The new charge code appears in the view and may be used by employees.	(new_charge_code,appears,view)
		(code,may_be_used,employees)
Administrator adds a new client and project.	Administrator adds a new client and project.	(Administrator,adds,client)
		(Administrator,adds,project)
		(client,can_be,new)
The administrator sees a view of existing clients.	The administrator sees a view of existing clients.	(administrator,sees,view_existing_clients)
The administrator enters the name of a new client.	The administrator enters the name of a new client.	(administrator,enters,name)
Employees will not be able to bill to the project until the administrator adds a charge code.	Employees will not be able to bill to the project until the administrator adds a charge code.	(Employees,will_not_be,able_bill_project_administrator)
		(able_bill_project_administrator,adds,charge_code)
The administrator sees a view of all existing charge codes.	The administrator sees a view of all existing charge codes.	(administrator,sees,view_existing_charge_codes)
Charge codes are activities organized by client and project.	Charge codes are activities organized by client and project.	(Charge_codes,are,activities_organized)
		(Charge_codes,are,project)
The administrator attempts to add a charge code that has the same activity as another charge code for the project.	The administrator attempts to add a charge code has the same activity as another charge code for the project.	(administrator,attempts_to_add,charge_code)
		(charge_code,has,activity_charge_code_for_project)

		(activity_charge_code_for_project,can_be,same)
The administrator is notified of the conflict.	The administrator is notified of the conflict.	(administrator,is,notified_conflict)
No change to existing data.	No change to existing data.	(change,existing,data)
System is unable to store data due to system or communications failure.	System is unable to store data due to system or communications failure.	(System,is_unable_to_store,data)
		(System,communications,failure)
		(System,is_unable_to_store,failure)
		(system,can_be,due)
		(failure,can_be,due)
The system is unable to store the new charge code.	The system is unable to store the new charge code.	(system,is,unable)
		(unable,store,charge_code)
		(charge_code,can_be,new)
The user is notified of the error, complete with any available details.	The user is notified of the error.	(user,is,notified_error)
The failure is logged by the system.	The failure is logged by the system.	(failure,is,logged_system)

Tuplas identificadas: Caso de uso *Create Employee.*

Sentença Original	**Sentença Pré-processada**	**Tupla(s) da Ontologia**
The Create Employee use case allows the administrative user to add an employee to the system, so that the employee may enter his or her hours into the time-tracking system.	The Create Employee use case allows the administrative user to add an employee to the system.	(Create_Employee_use_case,allows,user)
		(user,add,employee)
		(user,can_be,administrative)
The Create Employee use case allows the administrative user to add an employee to the system, so that the employee may enter his or her hours into the time-tracking system.	The Create Employee use case allows the administrative user to add an employee to the system.	(Create_Employee_use_case,allows,administrative_user)
		(administrative_user,add,employee)
The user must be logged in as an administrative user.	The user must be logged in as an administrative user.	(user,must_be,logged_user)
		(logged_user,can_be,administrative)
The administrative user adds a new employee.	The administrative user adds a new employee.	(administrative_user,adds,employee)
		(employee,can_be,new)
The administrator sees a view of all existing employees by name.	The administrator sees a view of all existing employees by name.	(administrator,sees,view_existing_employees)
The administrator adds an	The administrator adds an	(administrator,adds,employee)

employee, with a name and password.	employee.	
The new employee appears in the view.	The new employee appears in the view.	(new_employee,appears,view)
An email is sent to the employee, instructing him or her to log in and change his or her password.	An email is sent to the employee.	(email,is,sent_employee)
The administrator sees a view of all existing employees by name.	The administrator sees a view of all existing employees by name.	(administrator,sees,view_existing_employees)
Administrator is notified of the conflict.	Administrator is notified of the conflict.	(Administrator,is,notified_conflict)
No change to existing data.	No change to existing data.	(change,existing,data)
System is unable to add the employee due to a system or communication error.	System is unable to add the employee due to a system or communication error.	(System,is_unable_to_add,employee)
		(System,is_unable_to_add,communication_error)
		(system,can_be,due)
		(communication_error,can_be,due)
The system is unable to complete the addition, due to a system or communication error.	The system is unable to complete the addition.	(system,is,unable)
		(unable,complete,addition)
The system informs the administrative user of the error, complete with available details.	The system informs the administrative user of the error.	(system,informs,user_error)
		(user_error,can_be,administrative)
The view reverts to the previous state.	The view reverts to the previous state.	(view,reverts,state)
		(state,can_be,previous)

Tuplas identificadas: Caso de uso *Export Time Entries.*

Sentença Original	**Sentença Pré-processada**	**Tupla(s) da Ontologia**
The user must be logged in as an administrative user.	The user must be logged in as an administrative user.	(user,must_be,logged_user)
		(logged_user,can_be,administrative)
The administrative user exports the time entries.	The administrative user exports the time entries.	(administrative_user,exports,time_entries)
The administrative user selects a range of dates.	The administrative user selects a range of dates.	(administrative_user,selects,range)
The administrative user selects a subset of clients or all.	The administrative user selects a subset of clients or all.	(administrative_user,selects,subset)
The administrative user selects a subset of employees or all.	The administrative user selects a subset of employees or all.	(administrative_user,selects,subset)
The administrative user selects a target file.	The administrative user selects a target file.	(administrative_user,selects,target_file)

The data is exported to the file as XML.	The data is exported to the file as XML.	(data,is,exported_file)
System is unable to export the entries due to a system error.	System is unable to export the entries due to a system error.	(System,is,unable)
		(unable,export,entries)
		(system_error,can_be,due)
The administrative user selects a range of dates.	The administrative user selects a range of dates.	(administrative_user,selects,range)
The system is unable to export the entries.	The system is unable to export the entries.	(system,is,unable)
		(unable,export,entries)
The administrative user is notified of the error.	The administrative user is notified of the error.	(administrative_user,is_notified,error)

Tuplas identificadas: Caso de uso *Login.*

Sentença Original	**Sentença Pré-processada**	**Tupla(s) da Ontologia**
The Login use case allows employees and the administrative user to access the system.	The Login use case allows employees and the administrative user to access the system.	(Login_use_case,allows,employees)
		(user,access,system)
		(user,can_be,administrative)
Employees must be able to log in from any computer, including home, client sites, and on the road.	Employees must be able to log in from any computer.	(Employees,must_be,able)
		(able,log,computer)
Employees must be able to log in from any computer, including home, client sites, and on the road.	Employees including home.	(Employees,including,home)
This access may be from behind a client's firewall.	This access may be from behind a client firewall.	(access,may_be_from,client_firewall)
The user is authenticated as either an administrator or an employee.	The user is authenticated as either an administrator or an employee.	(user,is,authenticated_administrator)
		(user,is,employee)
The failure is logged by the system.	The failure is logged by the system.	(failure,is,logged_by_system)
The user is allowed to try again indefinitely.	The user is allowed to try again indefinitely.	(user,is_allowed_to_try,again_indefinitely)

Tuplas identificadas: Caso de uso *Record Time.*

Sentença Original	**Sentença Pré-processada**	**Tupla(s) da Ontologia**
The Record Time use case allows any employee to track his or her own hours.	The Record Time use case allows any employee to track or own hours.	(Record_Time_use_case,allows,employee)
Administrative users can	Administrative users can	(Administrative_users,can_record,hours_employee

record hours for any employee.	record hours for any employee.	)
The Record Time use case must be accessible from client sites and the employees'homes.	The Record Time use case must be accessible from client sites and the employees homes.	(Record_Time_use_case,must_be,client_sites)
		(Record_Time_use_case,employees,homes)
		(Record_Time_use_case,must_be,homes)
		(client_sites,can_be,accessible)
The employee selects a charge number from all available charge numbers, organized by client and project.	The employee selects a charge number from all available charge numbers.	(employee,selects,charge_number)
The employee selects a charge number from all available charge numbers, organized by client and project.	employee organized by client and project.	(employee,organized,client)
		(employee,organized,project)
The employee selects a day from the time period.	The employee selects a day from the time period.	(employee,selects,day_time_period)
The employee enters the hours worked as a positive decimal number.	The employee enters the hours worked as a positive decimal number.	(employee,enters,hours)
		(hours,worked,number)
		(number,can_be,positive_decimal)
Employee edits existing data.	Employee edits existing data	(Employee,edits,existing_data)
The employee selects an existing entry.	The employee selects an existing entry.	(employee,selects,existing_entry)
Employee submits timecard as complete.	Employee submits timecard as complete.	(Employee,submits,timecard)
		(timecard,can_be,complete)
The employee elects to submit the timecard.	The employee elects to submit the timecard.	(employee,elects_to_submit,timecard)
Administrator edits an employee's timecard.	Administrator edits an employee timecard.	(Administrator,edits,employee_timecard)
The administrator selects an employee from a list.	The administrator selects an employee from a list.	(administrator,selects,employee_list)
The administrator selects an existing entry.	The administrator selects an existing entry.	(administrator,selects,existing_entry)
Administrator submits an employee's timecard as complete.	Administrator submits an employee timecard as complete.	(Administrator,submits,employee_timecard)
		(employee_timecard,can_be,complete)
The administrator selects an employee from a list.	The administrator selects an employee from a list.	(administrator,selects,employee_list)
The administrator elects to submit the timecard.	The administrator elects to submit the timecard.	(administrator,elects_to_submit,timecard)
The submission is logged as an unusual activity.	The submission is logged as an unusual activity.	(submission,is,logged_activity)
		(logged_activity,can_be,unusual)

A tabela abaixo apresenta as tuplas duplicadas no processamento das seis descrições de casos de uso, e o número total de vezes que o aplicativo *OntSoft* retorna cada tupla repetida.

Lista de tuplas duplicadas: Sistema *Time Card.*

Tuplas repetidas	Número de repetições
(Employees,must_be,able)	1
(able,log,computer)	1
(Employees,including,home)	1
(access,may_be_from,client_firewall)	1
(failure,is,logged_system)	1
(user,is_allowed_to_try,again_indefinitely)	1
(system,can_be,due)	2
(user,must_be,logged_user)	2
(logged_user,can_be,administrative)	2
(administrator,is,notified_conflict)	1
(change,existing,data)	1
(system,is,unable)	3
(user,is,notified_error)	1
(user,can_be,administrative)	1
(administrator,sees,view_existing_employees)	1
(unable,export,entries)	1
(administrative_user,selects,range)	1
(administrative_user,selects,subset)	1

APÊNDICE B – TERMOS CANDIDATOS A CONCEITOS

Abaixo se encontra a lista dos termos candidatos a conceitos, extraídos da descrição detalhada de caso de uso *process sale*. As tabelas apresentam também o número de vezes que estes termos candidatos a conceitos aparecem nas tuplas e se o termo é um conceito da ontologia inicial, de acordo com sua classificação: uni gramas, bi gramas, tri gramas ou *n* gramas (mais de 3 palavras). As tabelas a seguir apresentam os termos extraídos do detalhamento de caso de uso *process sale* [LAR07].

Lista de uni gramas – *process sale.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Cashier	62	sim
System	58	não
Customer	20	sim
Sale	17	sim
Error	10	não
Manager	8	não
Price	8	não
Alternate	5	não
Company	5	não
Failure	4	não
New	4	não
Operation	4	não
Receipt	4	não
Total	4	não
Approval	3	não
Cash	3	não
Item	3	sim
Manual	3	não
State	3	não
Status	3	não
Able	2	não
Accounting	2	não
Category	2	não
Correct	2	não
Discount	2	não
External	2	não
Format	2	não
ID	2	não
Inventory	2	não
Items	2	sim (plural)
Method	2	não
Mode	2	não
payment	2	sim

Proof	2	não
Protocol	2	não
Sales	2	sim (plural)
Service	2	não
tax-exempt	2	não
Timeout	2	não
True	2	não
Update	2	não
Updated	2	não
Accurate	1	não
authenticated	1	não
available	1	não
Clean	1	não
Commissions	1	não
Confirm	1	não
Coupon	1	não
Coupons	1	não
Credit	1	não
Denial	1	não
description	1	não
Digital	1	não
Displays	1	não
Effort	1	não
Eligible	1	não
employee	1	não
Enough	1	não
Entry	1	não
Fast	1	não
generated	1	não
Goods	1	não
human-readable	1	não
identified	1	não
Manager-authorized	1	não
Manager-mode	1	não
Managers	1	não
Minimal	1	não
Needed	1	não
Override	1	não
Paper	1	não
payables	1	não
Pays	1	não
Prices	1	não
Prior	1	não
purchase	1	não
Quantity	1	não
Record	1	não
recorded	1	não
recovery	1	não
response	1	não
Returns	1	não
Salesperson	1	não

Saved	1	não
signature	1	não
Special	1	Não
Systems	1	Não
Tax	1	Não
transactions	1	Não
Verify	1	Não
Visible	1	Não
Void	1	Não

A tabela a seguir apresenta os termos candidatos a conceitos formados por duas palavras (bi grama). Estes termos, algumas vezes, apresentam uma de suas palavras já existentes como conceitos mapeados na ontologia inicial. Quando isto ocorre, na Tabela a informação da coluna da Tabela correspondente é apontada com o valor “sim” e ao lado o nome do conceito correspondente é discriminado.

Lista de bi gramas – *process sale.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Customer_payment	6	sim – Customer e payment
item_ID	6	sim - Item
gift_receipt	3	não
Manager_override	3	não
override_operation	3	não
price_entry	3	não
authorization_requests	2	não
cash_drawer	2	não
Customer_identification	2	sim - Customer
fault_tolerance	2	não
override_price	2	não
payment_approval	2	sim - payment
payment_method	2	sim - payment
payment_step	2	sim - payment
status_code	2	não
Suspended_sale	2	sim - sale
automatic_update	1	não
by_credit	1	não
cash_payment	1	sim - payment
Cashier-authorized_mode	1	sim - Cashier
correctly_calculated	1	não
credit_payment	1	sim - payment
credit_request	1	não
customer_interests	1	sim - customer
Customer_total	1	sim - Customer
discount_request	1	não

discount_rules	1	não
discount_total	1	não
fast_update	1	não
item_description	1	sim - item
item_identifier	1	sim - item
Item_price	1	sim - item
override_operations	1	não
payment_denial	1	sim - payment
payment_verification	1	sim - payment
price_tag	1	não
prior_state	1	não
product_rebates	1	não
purchased_item	1	sim - item
rebate_forms	1	não
remaining_credit	1	não
set_rules	1	não
System_records	1	não
tax_sale	1	sim - sale
taxes_calculated	1	não

Do mesmo modo que o apontado na tabela dos bigramas, quando uma das palavras pertencentes à lista de tri gramas ou *n* gramas representam conceitos mapeados na ontologia inicial, são indicados na coluna e linha da tabela correspondente. Abaixo estão listados os termos classificados como tri gramas e *n* gramas identificados no processamento das regras para formação das tuplas (conceito, relacionamento, conceito).

Lista de tri gramas – *process sale.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Payment_Authorization_Service	4	sim - Payment
item_entry_mode	2	sim - item
limit_for_Cashiers	2	sim - Cashier
manual_category_code	2	não
retrieval_POS_register	2	sim - POS
anomalies_preventing_recovery	1	não
cash_amount_tendered	1	não
credit_account_information	1	não
even_server_components	1	não
Government_Tax_Agencies	1	não
item_category_identifier	1	não
Payment_authorization_approvals	1	sim - Payment
payment_authorization_request	1	sim - payment
sale_line_item	1	sim – sale e item
sales_commissions_updated	1	sim - sale
service_POS_node	1	sim - POS

System_displays_state	1	não
System_records_sale	1	sim - sale

Lista de *n* gramas – *process sale.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
easily_display_entered_items	2	sim - item (plural)
item_identifier_for_removal	2	sim - item
tax_calculation_system_service	2	não
coupons_for_accounting_reasons	1	não
credit_payment_signature_input_mechanism	1	sim - payment
Customer_credit_payment_signature	1	sim – Customer e payment
Invalid_item_ID_System	1	sim - item
rebate_receipts_item_rebate	1	sim - item
System_records_cash_payment	1	sim - payment

As próximas tabelas apresentam os termos organizados em tabelas de uni gramas, bi gramas, tri gramas e *n* gramas, de acordo com a extração dos termos de cada detalhamento de caso de uso que descreve o sistema *Time Card*.

Lista de uni gramas – *Change Password.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Password	4	Não
System	4	Não
User	4	Sim
Employees	3	Não
Error	3	Não
Able	2	Não
Due	2	Não
Users	2	sim – plural
Access	1	Não
Administrative	1	Não
Computer	1	Não
Failure	1	Não
Home	1	Não
New	1	Não
Not	1	Não
Notified	1	Não

Lista de bi gramas – *Change Password.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
logged_system	2	Não
again_indefinitely	1	Não
been_changed	1	Não
client_firewall	1	sim - client
New_passwords	1	Não
notified_error	1	Não

O processamento da descrição detalhada do caso de uso chamado *Change Password* identifica além dos uni gramas e bi gramas listados nas tabelas a seguir, uma única ocorrência de um termo formado de quatro palavras (n grama): *Change_Password_use_case.* As tabelas seguintes apresentam os termos identificados para os demais detalhamentos de casos de uso.

Lista de uni gramas – *Create Charge Code.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Administrator	8	não
System	5	não
Failure	4	não
Project	4	sim
Client	2	sim
Data	2	não
Due	2	não
Employees	2	não
New	2	não
Unable	2	não
User	2	sim
Administrative	1	não
Change	1	não
Code	1	não
Name	1	não
Same	1	não
View	1	não

Lista de bi gramas – *Create Charge Code.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
charge_code	5	sim
Charge_codes	4	sim - plural
activities_organized	2	não
logged_user	2	sim - user
logged_system	1	não
notified_conflict	1	não
notified_error	1	não

Lista de tri gramas e *n* gramas – *Create Charge Code.*

Able_bill_project_administrator	**2**	**sim - project**
activity_charge_code_for_project	2	sim – charge code e project
New_charge_code	1	sim – charge code

Lista de uni gramas – *Create Employee.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Employee	6	não
System	5	não
Administrator	4	não
User	4	sim
Administrative	3	não
Due	2	não
State	2	não
Unable	2	não
View	2	não
Addition	1	não
Change	1	não
Data	1	não
Email	1	não
New	1	não
Previous	1	não

Lista de bi gramas – *Create Employee.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Administrative_user	3	sim - user

communication_error	2	não
logged_user	2	sim - user
user_error	2	sim - user
new_employee	1	não
notified_conflict	1	não
sent_employee	1	não

O processamento do detalhamento de caso de uso denominado *create employee* identificou ainda duas ocorrências do termo tri grama *"view_existing_employees"* e duas ocorrências do *n* grama *"Create_Employee_use_case"*.

Lista de uni gramas – *Export Time Entries.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Unable	4	não
Entries	2	sim - plural
Range	2	não
Subset	2	não
System	2	não
Administrative	1	não
Data	1	não
Due	1	não
Error	1	não
User	1	sim

Lista de bi gramas – *Export Time Entries.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
administrative_user	7	sim
logged_user	2	sim - user
exported_file	1	não
system_error	1	não
target_file	1	não
time_entries	1	sim – entry (plural)

Lista de uni gramas – *Login.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
User	5	sim
Employees	3	não
Able	2	não
Access	1	não
Administrative	1	não
Computer	1	não
Employee	1	não
Failure	1	não
Home	1	não
System	1	não

Lista de bi gramas e tri gramas – *Login.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
again_indefinitely	1	não
Authenticated_administrator	1	não
client_firewall	1	sim - client
logged_by_system	1	não
Login_use_case	1	não

Lista de uni gramas – *Record Time.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
Employee	10	não
Administrator	6	não
Timecard	4	sim
Complete	2	não
Homes	2	não
Hours	2	não
Number	2	não
Accessible	1	não
Client	1	sim
Project	1	sim
Submission	1	não
Unusual	1	não

Lista de bi gramas – *Record Time.*

Termos Candidatos	Número de ocorrências	Conceito Ontologia Inicial
employee_timecard	3	sim – time card
client_sites	2	sim - client
employee_list	2	não
existing_entry	2	sim - entry
logged_activity	2	não
Administrative_users	1	sim – user (plural)
charge_number	1	não
existing_data	1	não
hours_employee	1	não
positive_decimal	1	não

O detalhamento de caso de uso denominado *record time* extraiu o termo tri grama *"day_time_period"* e 4 ocorrências do *n* grama *"Record_Time_use_case"*.

APÊNDICE C – *SURVEY*

Para avaliação das tuplas extraídas dos detalhamentos de casos de uso dos sistemas *ProxGer* [LAR07] e *Time Card* [ARR01] foi aplicado uma *survey* para o preenchimento de 10 participantes. No documento da *survey*, foi apresentada uma seção com uma breve explicação e a ontologia inicial dos dois sistemas. Após, foi solicitado a marcação de um "X" para cada tupla considerada relevante de acordo com a percepção de cada membro participante. Participaram da *survey* alunos da pós-graduação dos cursos de mestrado e doutorado da PUC-RS e dois mestres formados convidados, com algum conhecimento em modelagem de sistemas e ontologias. As tabelas estão definidas com os índices conforme exemplo: Para a sentença "3. Customer Wants purchase and fast service with minimal effort." São geradas as 4 tuplas *3-a.(Customer,Wants,purchase), 3-b.(Customer,Wants,service), 3-c.(service,can_be,fast)* e *3-d.(effort,can_be,minimal)*. Abaixo, seguem as tabelas com as sentenças e tuplas ordenadas e as respostas de cada participante.

Sentenças pré-processadas – extraídas do caso de uso *process sale.*

Índice	Sentença Pré-processada
1	Cashier wants accurate.
2	Salesperson wants sales commissions updated.
3	Customer Wants purchase and fast service with minimal effort.
4	Customer Wants easily visible display of entered items and prices.
5	Customer Wants proof of purchase to support returns.
6	Company Wants to accurately record transactions and satisfy customer interests.
7	Company Wants some fault tolerance to allow sales capture even if server components.
8	Company Wants automatic and fast update of accounting and inventory.
9	Manager Wants to be able to quickly perform override operations.
10	Government Tax Agencies Want to collect tax from every sale.
11	Payment Authorization Service Wants to receive digital authorization requests in the correct format and protocol.
12	Payment Authorization Service Wants to accurately account for payables to the store.
13	Cashier is identified and authenticated.
14	Sale is saved.
15	Tax is correctly calculated.
16	Accounting and Inventory are updated.
17	Commissions recorded.
18	Receipt is generated.
19	Payment authorization approvals are recorded.
20	Cashier starts a new sale.

21	Cashier enters item identifier.
22	System records sale line item and presents item description.
23	System presents price.
24	System running total.
25	Price calculated from a set of price rules.
26	System presents total with taxes calculated.
27	Cashier tells Customer the total.
28	Cashier asks for payment.
29	Customer pays and System handles payment.
30	System presents receipt.
31	Customer leaves with receipt and goods.

Tuplas identificadas a partir das sentenças pré-processadas.

índice tuplas	Tupla(s) da Ontologia
1	(Cashier,wants,accurate)
2	(Salesperson,wants,sales_commissions_updated)
3-a	(Customer,Wants,purchase)
3-b	(Customer,Wants,service)
3-c	(service,can_be,fast)
3-d	(effort,can_be,minimal)
4-a	(Customer,Wants,easily_display_entered_items)
4-b	(Customer,Wants,prices)
4-c	(easily_display_entered_items,can_be,visible)
5-a	(Customer,Wants,proof)
5-b	(proof,support,returns)
6-a	(Company,Wants_accurately_record,transactions)
6-b	(Company,satisfy,customer_interests)
7-a	(Company,Wants,fault_tolerance)
7-b	(fault_tolerance,allow,sales)
7-c	(sales,capture,even_server_components)
8-a	(Company,Wants,automatic_update)
8-b	(Company,Wants,fast_update)
8-c	(update,has_type,accounting)
8-d	(update,has_type,inventory)
9-a	(Manager,Wants_to_be,able)
9-b	(able,quickly_perform,override_operations)
10	(Government_Tax_Agencies,Want_to_collect,tax_sale)
11-a	(Payment_Authorization_Service,Wants_to_receive,authorization_requests)
11-b	(Payment_Authorization_Service,Wants_to_receive,protocol)
11-c	(Payment_Authorization_Service,Wants_to_receive,format)
11-d	(authorization_requests,can_be,digital)
11-e	(format,can_be,correct)
11-f	(protocol,can_be,correct)
12	(Payment_Authorization_Service,Wants_to_accurately_account_ for,payables)
13-a	(Cashier,is,identified)
13-b	(Cashier,is,authenticated)
14	(Sale,is,saved)
15	(Tax,is,correctly_calculated)
16-a	(Accounting,are,updated)
16-b	(Inventory,are,updated)

17	(Commissions,do,record)
18	(Receipt,is,generated)
19	(Payment_authorization_approvals,are,recorded)
20-a	(Cashier,starts,sale)
20-b	(sale,can_be,new)
21	(Cashier,enters,item_identifier)
22-a	(System,records,sale_line_item)
22-b	(System,presents,item_description)
23	(System,presents,price)
24	(System,running,total)
25	(Price,calculated_from,set_rules)
26-a	(System,presents,total)
26-b	(total,can_be,taxes_calculated)
27	(Cashier,tells,Customer_total)
28	(Cashier,asks_for,payment)
29-a	(Customer,do,pays)
29-b	(System,handles,payment)
30	(System,presents,receipt)
31-a	(Customer,leaves_with,receipt)
31-b	(Customer,leaves_with,goods)

Respostas dos participantes.

Índice tuplas	**participante 01**	**participante 02**	**participante 03**	**participante 04**	**participante 05**	**participante 06**	**participante 07**	**participante 08**	**participante 09**	**participante 10**
1			X				X			
2	X		X	X	X	X	X	X		X
3-a	X	X	X	X	X	X	X	X		X
3-b	X	X		X		X	X	X	X	
3-c					X	X				X
3-d					X	X				X
4-a				X	X	X	X	X		
4-b	X		X	X		X			X	
4-c				X		X				X
5-a	X	X		X		X	X			
5-b		X	X							X
6-a	X			X	X	X	X	X	X	X
6-b	X		X			X	X			
7-a	X		X	X	X	X	X	X		X
7-b				X		X	X			
7-c						X				
8-a		X	X	X	X	X	X	X	X	X
8-b	X				X	X	X	X	X	X
8-c	X			X	X	X	X		X	
8-d	X	X		X	X	X	X		X	
9-a			X							
9-b				X			X			
10	X	X	X	X	X	X	X	X		X
11-a	X	X	X	X		X	X	X		X
11-b		X		X		X	X			
11-c		X		X						
11-d			X		X		X	X	X	X
11-e					X		X	X		

11-f					X		X	X		
12				X	X	X	X	X		
13-a		X		X	X	X	X	X		
13-b		X	X	X	X	X	X	X		
14		X	X	X		X	X	X		
15		X	X	X		X	X	X		
16-a			X	X		X	X	X		
16-b			X	X		X	X	X		
17			X	X		X	X	X		
18		X	X	X		X	X	X		
19				X		X	X	X		
20-a	X		X	X	X	X	X	X	X	X
20-b					X			X		
21	X		X	X	X	X	X	X	X	X
22-a	X	X	X	X	X	X	X	X	X	X
22-b	X			X	X	X		X	X	X
23	X			X	X	X	X	X	X	X
24			X			X	X	X	X	
25		X		X	X	X	X	X	X	X
26-a	X		X			X	X	X	X	X
26-b										
27			X		X		X	X		X
28	X		X	X	X		X	X	X	X
29-a	X	X		X	X	X	X	X	X	X
29-b	X		X		X	X		X	X	
30	X	X	X	X	X	X	X	X	X	X
31-a	X				X		X	X	X	
31-b	X		X		X		X	X	X	

Sentenças pré-processadas – caso de uso *change password.*

Índice	Sentença Pré-processada
1	The Change Password use case allowXs employees and administrative users to change password.
2	The user must have logged in to the system.
3	Employees must be able to log in from any computer.
4	Employees including home.
5	This access may be from behind a client firewall.
6	The user is notified that or password has been changed.
7	The failure is logged by the system.
8	The user is allowed to try again indefinitely.
9	New passwords do not match.
10	System is unable to store new password due to a system or communications error.
11	The user is notified of the error.

Tuplas identificadas a partir das sentenças pré-processadas.

índice tuplas	Tupla(s) da Ontologia
1-a	(Change_Password_use_case,allows,employees)

1-b	(users,change,password)
1-c	(users,can_be,administrative)
2	(user,must_have,logged_system)
3-a	(Employees,must_be,able)
3-b	(able,log,computer)
4	(Employees,including,home)
5	(access,may_be_from,client_firewall)
6-a	(user,is,notified)
6-b	(password,has,been_changed)
7	(failure,is,logged_system)
8	(user,is_allowed_to_try,again_indefinitely)
9	(New_passwords,do,not)
10-a	(System,is_unable_to_store,password)
10-b	(System,communications,error)
10-c	(System,is_unable_to_store,error)
10-d	(password,can_be,new)
10-e	(system,can_be,due)
10-f	(error,can_be,due)
11	(user,is,notified_error)

Respostas dos participantes.

índice tuplas	participante 01	participante 02	participante 03	participante 04	participante 05	participante 06	participante 07	participante 08	participante 09	participante 10
1-a								x		
1-b	x	x	x	x	x	x	x	x	x	x
1-c	x			x	x	x		x	x	x
2			x	x		x	x	x		
3-a			x	x			x			
3-b				x						
4				x			x	x		
5	x		x			x	x	x	x	
6-a				x			x	x		
6-b		x	x	x			x	x		
7			x	x			x	x		
8			x	x	x		x	x		
9							x			

10-a			X	X		X	X	X	X	
10-b	X			X			X			
10-c				X	X			X		
10-d				X	X			X		
10-e				X						
10-f				X						
11			X		X		X	X		

Sentenças pré-processadas – caso de uso *create charge code.*

índice	Sentença Pré-processada
1	The user must be logged in as an administrative user.
2	The administrator sees a view of existing charge codes for a selected project.
3	The new charge code appears in the view and may be used by employees.
4	Administrator adds a new client and project.
5	The administrator sees a view of existing clients.
6	The administrator enters the name of a new client.
7	Employees will not be able to bill to the project until the administrator adds a charge code.
8	The administrator sees a view of all existing charge codes.
9	Charge codes are activities organized by client and project.
10	The administrator attempts to add a charge code has the same activity as another charge code for the project.
11	The administrator is notified of the conflict.
12	No change to existing data.
13	System is unable to store data due to system or communications failure.
14	The system is unable to store the new charge code.
15	The user is notified of the error.
16	The failure is logged by the system.

Tuplas identificadas a partir das sentenças pré-processadas.

índice tuplas	Tupla(s) da Ontologia
1-a	(user,must_be,logged_user)
1-b	(logged_user,can_be,administrative)
2	(administrator,sees,view_existing_charge_codes_selected_project)
3-a	(new_charge_code,appears,view)
3-b	(code,may_be_used,employees)
4-a	(Administrator,adds,client)
4-b	(Administrator,adds,Project)

4-c	(client,can_be,new)
5	(administrator,sees,view_existing_clients)
6	(administrator,enters,name)
7-a	(Employees,will_not_be,able_bill_project_administrator)
7-b	(able_bill_project_administrator,adds,charge_code)
8	(administrator,sees,view_existing_charge_codes)
9-a	(Charge_codes,are,activities_organized)
9-b	(Charge_codes,are,Project)
10-a	(administrator,attempts_to_add,charge_code)
10-b	(charge_code,has,activity_charge_code_for_project)
10-c	(activity_charge_code_for_project,can_be,same)
11	(administrator,is,notified_conflict)
12	(change,existing,data)
13-a	(System,is_unable_to_store,data)
13-b	(System,communications,failure)
13-c	(System,is_unable_to_store,failure)
13-d	(system,can_be,due)
13-e	(failure,can_be,due)
14-a	(system,is,unable)
14-b	(unable,store,charge_code)
14-c	(charge_code,can_be,new)
15	(user,is,notified_error)
16	(failure,is,logged_system)

Respostas dos participantes.

índice tuplas	**participante 01**	**participante 02**	**participante 03**	**participante 04**	**participante 05**	**participante 06**	**participante 07**	**participante 08**	**participante 09**	**participante 10**
1-a	X	X	X							
1-b	X				X		X	X	X	
2				X		X	X	X	X	
3-a				X		X	X	X		
3-b	X		X	X	X	X	X	X	X	
4-a	X	X	X	X	X	X	X	X	X	X
4-b	X	X	X	X	X	X	X	X	X	X
4-c					X			X		
5			X			X	X	X		
6	X		X	X	X	X	X	X	X	X
7-a							X	X		
7-b							X			
8	X		X	X		X	X	X	X	

9-a		X	X	X	X	X		X		
9-b	X	X		X						
10-a	X		X	X	X		X	X		
10-b			X							
10-c								X		
11			X	X			X	X		
12				X			X			
13-a			X	X	X	X	X	X	X	
13-b				X			X	X		
13-c				X				X		
13-d										
13-e							X			
14-a				X						
14-b				X			X			
14-c					X			X		
15		X	X	X			X			
16			X	X			X			

Sentenças pré-processadas – caso de uso *create employee.*

índice	Sentença Pré-processada
1	The Create Employee use case allows the administrative user to add an employee to the system.
2	The Create Employee use case allows the administrative user to add an employee to the system.
3	The user must be logged in as an administrative user.
4	The administrative user adds a new employee.
5	The administrator sees a view of all existing employees by name.
6	The administrator adds an employee.
7	The new employee appears in the view.
8	An email is sent to the employee.
9	The administrator sees a view of all existing employees by name.
10	Administrator is notified of the conflict.
11	No change to existing data.
12	System is unable to add the employee due to a system or communication error.
13	The system is unable to complete the addition.
14	The system informs the administrative user of the error.
15	The view reverts to the previous state.

Tuplas identificadas a partir das sentenças pré-processadas.

índice tuplas	Tupla(s) da Ontologia
1-a	(Create_Employee_use_case,allows,user)
1-b	(user,add,employee)
1-c	(user,can_be,administrative)

2-a	(Create_Employee_use_case,allows,administrative_user)
2-b	(administrative_user,add,employee)
3-a	(user,must_be,logged_user)
3-b	(logged_user,can_be,administrative)
4-a	(administrative_user,adds,employee)
4-b	(employee,can_be,new)
5	(administrator,sees,view_existing_employees)
6	(administrator,adds,employee)
7	(new_employee,appears,view)
8	(email,is,sent_employee)
9	(administrator,sees,view_existing_employees)
10	(Administrator,is,notified_conflict)
11	(change,existing,data)
12-a	(System,is_unable_to_add,employee)
12-b	(System,is_unable_to_add,communication_error)
12-c	(system,can_be,due)
12-d	(communication_error,can_be,due)
13-a	(system,is,unable)
13-b	(unable,complete,addition)
14-c	(system,informs,user_error)
14-d	(user_error,can_be,administrative)
15-a	(view,reverts,state)
15-b	(state,can_be,previous)

Respostas dos participantes.

índice tuplas	**participante 01**	**participante 02**	**participante 03**	**participante 04**	**participante 05**	**participante 06**	**participante 07**	**participante 08**	**participante 09**	**participante 10**
1-a				X						
1-b		X	X	X				X	X	
1-c	X				X		X	X	X	
2-a				X			X	X		
2-b	X		X	X	X	X		X	X	X
3-a		X	X					X	X	
3-b		X			X		X	X	X	
4-a	X	X	X	X	X	X		X	X	X
4-b					X		X	X		
5	X		X	X		X	X	X	X	
6	X	X	X	X	X	X	X	X	X	X
7			X	X		X	X	X		
8			X				X	X		
9	X		X	X		X	X	X	X	
10				X		X	X			

11				X			X			
12-a			X	X		X	X	X	X	
12-b				X			X			
12-c										
12-d										
13-a			X	X						
13-b							X			
14-c	X			X	X		X	X		
14-d				X				X		
15-a			X	X		X	X		X	
15-b				X	X		X	X		

Sentenças pré-processadas – caso de uso *export time entries.*

índice	Sentença Pré-processada
1	The user must be logged in as an administrative user.
2	The administrative user exports the time entries.
3	The administrative user selects a range of dates.
4	The administrative user selects a subset of clients or all.
5	The administrative user selects a subset of employees or all.
6	The administrative user selects a target file.
7	The data is exported to the file as XML.
8	System is unable to export the entries due to a system error.
9	The administrative user selects a range of dates.
10	The system is unable to export the entries.
11	The administrative user is notified of the error.

Tuplas identificadas a partir das sentenças pré-processadas.

índice tuplas	Tupla(s) da Ontologia
1-a	(user,must_be,logged_user)
1-b	(logged_user,can_be,administrative)
2	(administrative_user,exports,time_entries)
3	(administrative_user,selects,range)
4	(administrative_user,selects,subset)
5	(administrative_user,selects,subset)
6	(administrative_user,selects,target_file)
7	(data,is,exported_file)
8-a	(System,is,unable)
8-b	(unable,export,entries)
8-c	(system_error,can_be,due)
9	(administrative_user,selects,range)

10	(system,is,unable)
11-a	(unable,export,entries)
11-b	(administrative_user,is_notified,error)

Respostas dos participantes.

índice tuplas	participante 01	participante 02	participante 03	participante 04	participante 05	participante 06	participante 07	participante 08	participante 09	participante 10
1-a		X		X					X	
1-b		X		X			X	X	X	
2	X	X	X	X	X	X	X	X	X	X
3	X		X					X	X	
4	X		X	X						
5	X		X							
6	X		X	X		X		X	X	X
7		X	X	X			X	X		
8-a			X	X						
8-b							X			
8-c										
9			X	X				X	X	
10			X	X						
11-a							X			
11-b			X			X	X	X	X	

Sentenças pré-processadas – caso de uso *login.*

índice	Sentença Pré-processada
1	The Login use case allows employees and the administrative user to access the system.
2	Employees must be able to log in from any computer.
3	Employees including home.
4	This access may be from behind a client firewall.
5	The user is authenticated as either an administrator or an employee.
6	The failure is logged by the system.
7	The user is allowed to try again indefinitely.

Tuplas identificadas a partir das sentenças pré-processadas.

índice tuplas	Tupla(s) da Ontologia
1-a	(Login_use_case,allows,employees)
1-b	(user,access,system)
1-c	(user,can_be,administrative)
2-a	(Employees,must_be,able)
2-b	(able,log,computer)
3	(Employees,including,home)
4	(access,may_be_from,client_firewall)
5-a	(user,is,authenticated_administrator)
5-b	(user,is,employee)
6	(failure,is,logged_by_system)
7	(user,is_allowed_to_try,again_indefinitely)

Respostas dos participantes.

índice tuplas	participante 01	participante 02	participante 03	participante 04	participante 05	participante 06	participante 07	participante 08	participante 09	participante 10
1-a		X		X						
1-b			X	X	X	X		X	X	X
1-c					X	X	X	X	X	
2-a				X			X			
2-b				X						
3								X		
4				X		X	X	X	X	
5-a			X	X			X	X	X	
5-b	X	X	X	X			X	X	X	
6			X	X			X	X		
7							X	X		

Sentenças pré-processadas – caso de uso *record time.*

índice	Sentença Pré-processada
1	The Record Time use case allows any employee to track or own hours.
2	Administrative users can record hours for any employee.
3	The Record Time use case must be accessible from client sites and the employees homes.
4	The employee selects a charge number from all available charge numbers.

5	employee organized by client and project.
6	The employee selects a day from the time period.
7	The employee enters the hours worked as a positive decimal number.
8	Employee edits existing data
9	The employee selects an existing entry.
10	Employee submits timecard as complete.
11	The employee elects to submit the timecard.
12	Administrator edits an employee timecard.
13	The administrator selects an employee from a list.
14	The administrator selects an existing entry.
15	Administrator submits an employee timecard as complete.
16	The administrator selects an employee from a list.
17	The administrator elects to submit the timecard.
18	The submission is logged as an unusual activity.

Tuplas identificadas a partir das sentenças pré-processadas.

índice tuplas	Tupla(s) da Ontologia
1	(Record_Time_use_case,allows,employee)
2	(Administrative_users,can_record,hours_employee)
3-a	(Record_Time_use_case,must_be,client_sites)
3-b	(Record_Time_use_case,employees,homes)
3-c	(Record_Time_use_case,must_be,homes)
3-d	(client_sites,can_be,accessible)
4	(employee,selects,charge_number)
5-a	(employee,organized,client)
5-b	(employee,organized,project)
6	(employee,selects,day_time_period)
7-a	(employee,enters,hours)
7-b	(hours,worked,number)
7-c	(number,can_be,positive_decimal)
8	(Employee,edits,existing_data)
9	(employee,selects,existing_entry)
10-a	(Employee,submits,timecard)
10-b	(timecard,can_be,complete)
11	(employee,elects_to_submit,timecard)
12	(Administrator,edits,employee_timecard)
13	(administrator,selects,employee_list)
14	(administrator,selects,existing_entry)

15-a	(Administrator,submits,employee_timecard)
15-b	(employee_timecard,can_be,complete)
16	(administrator,selects,employee_list)
17	(administrator,elects_to_submit,timecard)
18-a	(submission,is,logged_activity)
18-b	(logged_activity,can_be,unusual)

Respostas dos participantes.

índice tuplas	participante 01	participante 02	participante 03	participante 04	participante 05	participante 06	participante 07	participante 08	participante 09	participante 10
1		X	X	X			X			
2		X	X	X	X		X	X	X	
3-a										
3-b										
3-c										
3-d							X	X		
4	X		X			X	X	X	X	
5-a			X			X	X			
5-b			X			X	X			
6	X		X	X		X	X	X	X	
7-a	X	X	X			X	X	X	X	X
7-b							X			
7-c						X			X	
8		X	X	X	X	X	X	X	X	X
9	X	X	X		X	X	X	X	X	X
10-a	X	X	X	X		X	X	X	X	X
10-b				X				X		
11	X		X			X	X	X	X	
12	X		X	X	X	X	X	X	X	X
13	X						X			
14	X		X	X		X	X	X	X	
15-a	X		X		X	X	X	X	X	
15-b	X							X		
16	X			X			X			
17	X		X			X	X	X	X	
18-a			X	X			X			
18-b				X				X		

ANEXO A – DESCRIÇÕES DE CASOS DE USO

O detalhamento de caso de uso denominado *process sale*, descreve o sistema *ProxGer*, e está descrito logo abaixo.

Detalhamento de caso de uso extraído de Larman [LAR07].

ID	**UC1**
Name	Process Sale
Actors	Cashier
Stakeholders	Cashier: wants accurate, fast entry, and no payment errors, as cash drawer short ages are deducted from his/her salary. Salesperson: wants sales commissions updated. Customer: Wants purchase and fast service with minimal effort. Wants easily visible display of entered items and prices. Wants proof of purchase to support returns. Company: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server components (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory. Manager: Wants to be able to quickly perform override operations, and easily debug Cashier problems. Government Tax Agencies: Want to collect tax from every sale. May be multiple agencies, such as national, state, and county. Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.
Preconditions	Cashier is identified and authenticated.
Postconditions	Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated. Payment authorization approvals are recorded.
Main Flow	1. Customer arrives at POS checkout with goods and/or services to purchase. 2. Cashier starts a new sale. 3. Cashier enters item identifier. 4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules. Cashier repeats steps 3-4 until indicates done.

	5. System presents total with taxes calculated. 6. Cashier tells Customer the total, and asks for payment. 7. Customer pays and System handles payment. 8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory). 9. System presents receipt. 10. Customer leaves with receipt and goods (if any).
Extensions	*a. At any time, Manager requests an override operation: 1. System enters Manager-authorized mode. 2. Manager or Cashier performs one Manager-mode operation. e.g., cash balance change, resume a suspended sale on another register, void a sale, etc. 3. System reverts to Cashier-authorized mode. *b. At any time, System fails: To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario. 1. Cashier restarts System, logs in, and requests recovery of prior state. 2. System reconstructs prior state. 2a. System detects anomalies preventing recovery: 1. System signals error to the Cashier, records the error, and enters a clean state. 2. Cashier starts a new sale. 1a. Customer or Manager indicate to resume a suspended sale. 1. Cashier performs resume operation, and enters the ID to retrieve the sale. 2. System displays the state of the resumed sale, with subtotal. 2a. Sale not found. 1. System signals error to the Cashier. 2. Cashier probably starts new sale and re-enters all items. 3. Cashier continues with sale (probably entering more items or handling payment). 2-4a. Customer tells Cashier they have a tax-exempt status (e.g., seniors, native peoples). 1. Cashier verifies, and then enters tax-exempt status code. 2. System records status (which it will use during tax calculations) 3a. Invalid item ID (not found in system):

	1. System signals error and rejects entry. 2. Cashier responds to the error: 2a. There is a human-readable item ID (e.g., a numeric UPC): 1. Cashier manually enters the item ID. 2. System displays description and price. 2a. Invalid item ID: System signals error. Cashier tries alternate method. 2b. There is no item ID, but there is a price on the tag: 1. Cashier asks Manager to perform an override operation. 2. Managers performs override. 3. Cashier indicates manual price entry, enters price, and requests standard taxation for this amount (because there is no product information, the tax engine can't otherwise deduce how to tax it). 2c. Cashier performs Find Product Help to obtain true item ID and price. 2d. Otherwise, Cashier asks an employee for the true item ID or price, and does either manual ID or manual price entry (see above). 3b. There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers): 1. Cashier can enter item category identifier and the quantity. 3c. Item requires manual category and price entry (such as flowers or cards with a price on them): 1. Cashier enters special manual category code, plus the price. 3-6a: Customer asks Cashier to remove (i.e., void) an item from the purchase: This is only legal if the item value is less than the void limit for Cashiers, otherwise a Manager override is needed. 1. Cashier enters item identifier for removal from sale. 2. System removes item and displays updated running total. 2a. Item price exceeds void limit for Cashiers: 1. System signals error, and suggests Manager override. 2. Cashier requests Manager override, gets it, and repeats operation. 3-6b. Customer tells Cashier to cancel sale: 1. Cashier cancels sale on System. 3-6c. Cashier suspends the sale: 1. System records sale so that it is available for retrieval on any POS register.

	2. System presents a "suspend receipt" that includes the line items, and a sale ID used to retrieve and resume the sale. 4a. The system supplied item price is not wanted (e.g., Customer complained about something and is offered a lower price): 1. Cashier requests approval from Manager. 2. Manager performs override operation. 3. Cashier enters manual override price. 4. System presents new price. 5a. System detects failure to communicate with external tax calculation system service: 1. System restarts the service on the POS node, and continues. 1a. System detects that the service does not restart. 1. System signals error. 2. Cashier may manually calculate and enter the tax, or cancel the sale. 5b. Customer says they are eligible for a discount (e.g., employee, preferred customer): 1. Cashier signals discount request. 2. Cashier enters Customer identification. 3. System presents discount total, based on discount rules. 5c. Customer says they have credit in their account, to apply to the sale: 1. Cashier signals credit request. 2. Cashier enters Customer identification. 3. Systems applies credit up to price=0, and reduces remaining credit. 6a. Customer says they intended to pay by cash but don't have enough cash: 1. Cashier asks for alternate payment method. 1a. Customer tells Cashier to cancel sale. Cashier cancels sale on System. 7a. Paying by cash: 1. Cashier enters the cash amount tendered. 2. System presents the balance due, and releases the cash drawer. 3. Cashier deposits cash tendered and returns balance in cash to Customer. 4. System records the cash payment. 7b. Paying by credit: 1. Customer enters their credit account information. 2. System displays their payment for verification.

	3. Cashier confirms. 3a. Cashier cancels payment step: 1. System reverts to "item entry" mode. 4. System sends payment authorization request to an external Payment Authorization Service System, and requests payment approval. 4a. System detects failure to collaborate with external system: 1. System signals error to Cashier. 2. Cashier asks Customer for alternate payment. 5. System receives payment approval, signals approval to Cashier, and releases cash drawer (to insert signed credit payment receipt). 5a. System receives payment denial: 1. System signals denial to Cashier. 2. Cashier asks Customer for alternate payment. 5b. Timeout waiting for response. 1. System signals timeout to Cashier. 2. Cashier may try again, or ask Customer for alternate payment. 6. System records the credit payment, which includes the payment approval. 7. System presents credit payment signature input mechanism. 8. Cashier asks Customer for a credit payment signature. Customer enters signature. 9. If signature on paper receipt, Cashier places receipt in cash drawer and closes it. 7c. Paying by check. 7d. Paying by debit. 7e. Cashier cancels payment step: 1. System reverts to "item entry" mode. 7f. Customer presents coupons: 1. Before handling payment, Cashier records each coupon and System reduces price as appropriate. System records the used coupons for accounting reasons. 1a. Coupon entered is not for any purchased item: 1. System signals error to Cashier. 9a. There are product rebates: 1. System presents the rebate forms and rebate receipts for each item with a rebate. 9b. Customer requests gift receipt (no prices visible):

	1. Cashier requests gift receipt and System presents it. 9c. Printer out of paper. 1. If System can detect the fault, will signal the problem. 2. Cashier replaces paper. 3. Cashier requests another receipt.

Os seis detalhamentos de casos de uso abaixo descrevem o sistema *Time Card*, conforme apresentam as tabelas abaixo.

Detalhamento de caso de uso *Change Password* [ARR01].

Name	**Change Password**
Description	The Change Password use case allows employees and administrative users to change their password.
Preconditions	1. The user must have logged in to the system.
Deployment constraints	1. Employees must be able to log in from any computer, including home, client sites, and on the road. This access may be from behind a client's firewall.
Main Flow	Employee changes his or her password. 1. The user enters his or her current password and new password twice. 2. The user is notified that his or her password has been changed.
Alternate flow	Invalid current password. 1. The user enters his or her current password and new password twice. 2. The user is notified that his or her attempt failed. 3. The failure is logged by the system. 4. The user is allowed to try again indefinitely.
Alternate flow	New passwords do not match. 1. The user enters his or her current password and new password twice. 2. The user is notified that his or her attempt failed. 3. The user is allowed to try again indefinitely.
Exception flow	System is unable to store new password due to a system or communications error. 1. The user enters his or her current password and new password twice. 2. The user is notified of the error, complete with any available details.

	3. The failure is logged by the system.

Detalhamento de caso de uso *Create Charge Code* [ARR01].

Name	**Create Charge Code**
Description	The Create Charge Code use case allows the administrative user to add a new charge code to the system so that employees can bill to the charge code. Since each charge code is specific to a client and a project, the administrative user may need to add a client or project first.
Preconditions	1. The user must be logged in as an administrative user.
Main Flow	Add a charge code to an existing project. 1. The administrator sees a view of existing charge codes for a selected project. Charge codes are activities organized by client and project. 2. The administrator selects from a list of common activities or enters a new activity to create a new charge code for the selected project. 3. The new charge code appears in the view and may be used by employees.
Alternate flow	Administrator adds a new client and project. 1. The administrator sees a view of existing clients. 2. The administrator enters the name of a new client. 3. The administrator selects the new client and enters the name and description for a new project. 4. Employees will not be able to bill to the project until the administrator adds a charge code.
Alternate flow	Duplicate data; input charge code already exists at the specified level. 1. The administrator sees a view of all existing charge codes. Charge codes are activities organized by client and project. 2. The administrator attempts to add a charge code that has the same activity as another charge code for the project. Once the list of common activities that does not include duplicates is created, this will be less likely to happen. 3. The administrator is notified of the conflict. No change to existing data.
Exception flow	System is unable to store data due to system or communications failure. 1. The administrator sees a view of existing charge codes for a selected project.

	Charge codes are activities organized by client and project. 2. The administrator selects from a list of common activities or enters a new activity to create a new charge code for the selected project. 3. The system is unable to store the new charge code. 4. The user is notified of the error, complete with any available details. 5. The failure is logged by the system.

Detalhamento de caso uso *Create Employee* [ARR01].

Name	**Create Employee**
Description	The Create Employee use case allows the administrative user to add an employee to the system, so that the employee may enter his or her hours into the time-tracking system.
Preconditions	1. The user must be logged in as an administrative user.
Main Flow	The administrative user adds a new employee. 1. The administrator sees a view of all existing employees by name. 2. The administrator adds an employee, with a name and password. 3. The new employee appears in the view. 4. An email is sent to the employee, instructing him or her to log in and change his or her password. 5. The employee can log in.
Alternate flow	Duplicate data; employee already exists. 1. The administrator sees a view of all existing employees by name. 2. The administrator adds an employee, with a name and password. 3. Administrator is notified of the conflict. No change to existing data.
Exception flow	System is unable to add the employee due to a system or communication error. 1. The administrative user sees a view of all existing employees by name. 2. The administrative user adds an employee, with a name and a password. The system is unable to complete the addition, due to a system or communication error. 3. The system informs the administrative user of the error, complete with available details. The view reverts to the previous state. 4. If possible, an error is added to a log.

Detalhamento de caso de uso *Export Time Entries* [ARR01].

Name	**Export Time Entries**
Description	The Export Time Entries use case allows the administrative user to save specified time-tracking data to a formatted file.
Preconditions	1. The user must be logged in as an administrative user.
Main Flow	The administrative user exports the time entries. 1. The administrative user selects a range of dates. 2. The administrative user selects a subset of clients or all. 3. The administrative user selects a subset of employees or all. 4. The administrative user selects a target file. 5. The data is exported to the file as XML. The administrator is notified when the process is complete.
Exception flow	System is unable to export the entries due to a system error. 1. The administrative user selects a range of dates. 2. The administrative user selects a subset of clients or all. 3. The administrative user selects a subset of employees or all. 4. The administrative user selects a target file. 5. The system is unable to export the entries. The administrative user is notified of the error. 6. If possible, the error is recorded to a log.

Detalhamento de caso de uso *Login* [ARR01].

Name	**Login**
Description	The Login use case allows employees and the administrative user to access the system.
Deployment constraints	1. Employees must be able to log in from any computer, including home, client sites, and on the road. This access may be from behind a client's firewall.
Main Flow	The administrative user or employee's username and password are valid. 1. The administrator or employee supplies a username and password. 2. The user is authenticated as either an administrator or an employee. This is not a choice during the login; it is determined by the username.

Alternate flow	First Login. 1. The administrator or employee supplies a username and password. 2. The user is authenticated as either an administrator or an employee. This is not a choke during the login; it is determined by the username. 3. The user is instructed to change his or her password. 4. Include the Change Password use case at this point.
Alternate flow	Invalid authentication information. 1. The administrator or employee supplies a username and password. 2. The user is notified that he or she has entered incorrect login information. 3. The failure is logged by the system. 4. The user is allowed to try again indefinitely.

Detalhamento de caso de uso *Record Time* [ARR01].

Name	**Record Time**
Description	The Record Time use case allows any employee to track his or her own hours. Administrative users can record hours for any employee.
Preconditions	1. The user must be logged in.
Deployment constraints	The Record Time use case must be accessible from client sites and the employees'homes. In the case of client sites, they will often be behind the client's firewall.
Main Flow	An employee records his or her own time. 1. The employee sees any previously entered data for the current time period. 2. The employee selects a charge number from all available charge numbers, organized by client and project. 3. The employee selects a day from the time period. 4. The employee enters the hours worked as a positive decimal number. 5. The new hours are added to the view and are seen in any subsequent views.
Alternate flow	Employee edits existing data. 1. The employee sees previously entered data for the current rime period. 2. The employee selects an existing entry. 3. The employee changes the charge number and/or the hours worked.

	4. The new information is updated in the view and is seen in any subsequent views.
Alternate flow	Employee submits timecard as complete. 1. The employee sees any previously entered data for the current time period. 2. The employee elects to submit the timecard. 3. The employee is asked to confirm his or her choice and warned that he or she will not be able to edit his or her entries. 4. The timecard is submitted; it is no longer available for editing.
Alternate flow	Administrator edits an employee's timecard. 1. The administrator selects an employee from a list. 2. The administrator sees previously entered data for the current time period. 3. The administrator selects an existing entry. 4. The administrator changes the charge number and/or the hours worked. 5. The update is logged as an unusual activity. 6. The new information is updated in the view and is seen in any subsequent views.
Alternate flow	Administrator submits an employee's timecard as complete. 1. The administrator selects an employee from a list. 2. The administrator sees any previously entered data for the current time period. 3. The administrator elects to submit the timecard. 4. The administrator is asked to confirm his or her choice and warned that he or she will not be able to edit his or her entries. 5. The submission is logged as an unusual activity. 6. The timecard is submitted; it is no longer available for editing.