Definição de Escopo em Linhas de Produto de Software: uma abordagem semi-automática por meio de anotação linguística

Andressa Ianzen, Andreia Malucelli, Sheila Reinehr
Programa de Pós-Graduação em Informática (PPGIa)
Polytechnic School
Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba - PR, Brasil
aianzen@ppgia.pucpr.br, malu@ppgia.pucpr.br, sheila.reinehr@pucpr.br

***Abstract*—To identify the reuse assets that should be developed in a software product line, it is necessary to perform an activity called product line scoping, which aims at identify and define products, features and areas of the field that should be part of the product line, as well as common and variables features. When creating the product line from existent legacy systems, the scoping activity must use the knowledge about the products of these software systems. The approaches for scoping are dependent of the knowledge about the product's domain. Therefore, it is necessary to have the domain expert participation, however this professional often do not have too much free time. We present a proposal to analyze documents and to identify features of a product line semi-automatically, using techniques of linguistic annotation in order to minimize the time spent by domain experts.**

Keywords-product line scoping, software product lines, product family engineering

Resumo—Para a identificação dos ativos reutilizáveis que devem ser desenvolvidos em uma linha de produtos de software (LPS) é essencial a realização da atividade de definição de escopo da LPS, cujo objetivo é identificar e delimitar produtos, funcionalidades e áreas do domínio que devem fazer parte da LPS, assim como funcionalidades comuns e variáveis. Ao criar uma LPS a partir de sistemas de software existentes é necessário utilizar o conhecimento sobre estes sistemas durante a atividade de definição de escopo. As abordagens para realizar a definição de escopo são dependentes do conhecimento do domínio destes sistemas, havendo a necessidade da participação ativa de especialistas neste domínio, que normalmente não possuem muito tempo livre. Assim, este artigo apresenta uma proposta para analisar documentos e identificar funcionalidades de uma LPS de maneira semi-automática, utilizando técnicas de anotação linguística, com o objetivo de minimizar o tempo gasto pelos especialistas de domínio.

Palavras-chave-definição de escopo, linhas de produto de software, engenharia de família de produtos

I. INTRODUÇÃO

Construir software com qualidade e desempenho é o foco da Engenharia de Software, que segundo [1], desenvolve Sistemas de Informação (SIs) de alta qualidade com custos adequados e se relaciona com todos os aspectos da produção de software.

Segundo [2], para obter produtos de alta qualidade e economicamente viáveis é necessário um conjunto sistemático de ferramentas, processos e técnicas, sendo o reuso uma das técnicas mais relevantes nesse conjunto. Ainda, segundo os autores, o enfoque da Linha de Produto de Software (LPS) surge como uma proposta de construção sistemática de software.

De acordo com [3], a LPS permite que a organização ganhe vantagem competitiva, pois gera o aumento da qualidade, redução dos custos, redução do tempo de entrega e minimização dos riscos dos produtos.

A abordagem de LPS pode ser dividida em duas atividades, segundo [4]: a engenharia de domínio e a engenharia de aplicação. Na engenharia de domínio são identificados e desenvolvidos os ativos que serão reutilizados pelos produtos da LPS. Na engenharia de aplicação, os ativos desenvolvidos são selecionados para a criação de determinados produtos.

As abordagens para que uma organização inicie o uso das LPS podem ser classificadas em pró-ativas, reativas e extrativas [5]:

- Na abordagem do tipo pró-ativa, a base comum de ativos (*assets*) reutilizáveis é desenvolvida primeiro e os produtos são desenvolvidos utilizando os ativos.
- Na abordagem reativa, a linha é iniciada com um ou alguns poucos produtos, incrementando-a na medida em que novos produtos são necessários.
- A abordagem extrativa se utiliza de um ou mais produtos de software existentes para criar a base de ativos comuns.

Para a identificação dos ativos reutilizáveis que devem ser desenvolvidos, é essencial realizar a atividade de definição de escopo. Esta atividade é geralmente o primeiro passo a ser realizado ao iniciar a construção da LPS e tem como objetivo

mapear o escopo da LPS, identificando e delimitando os produtos, funcionalidades e áreas do domínio que devem fazer parte da LPS, assim como suas funcionalidades comuns e variáveis[6].

A correta definição de escopo implicará no sucesso da LPS, ou seja, se o escopo definido for muito grande, os ativos de software desenvolvidos terão que acomodar muitas variações e ficarão muito complexos para serem úteis; se for muito pequeno, os ativos de software podem não ser desenvolvidos de forma genérica o suficiente para acomodar crescimento futuro: a oportunidade de incluir novos produtos na LPS será rejeitada por estar fora do escopo, ou, se aceita, resultará em retrabalho; e se o escopo definido abranger os produtos errados, a LPS não encontrará um segmento de mercado para atender [7].

Em [4] os autores classificam esta atividade em três categorias principais: *product portfolio planning, domain potential analysis* e *asset scoping*. Na primeira categoria, o objetivo principal é definir os produtos que devem fazer parte da LPS e identificar suas principais funcionalidades; na segunda, o objetivo é avaliar onde os investimentos de reuso devem se focar; na terceira, o objetivo é definir os componentes que serão construídos para reuso. A abordagem proposta neste artigo tem o objetivo de apoiar abordagens de definição de escopo da categoria *product portfolio planning* (também chamada *product portfolio scoping*).

II. ABORDAGENS PARA DEFINIÇÃO DE ESCOPO

Existem diversas abordagens na literatura para realizar ou apoiar a atividade de definição de escopo, e para identificar as abordagens mais recentes foi realizada uma pesquisa em Maio de 2012. A pesquisa foi realizada nas bases científicas Scirus, IEEE Xplore, ACM (*Association for Computing Machinery*), Scopus e ScienceDirect, utilizando as palavras-chave "*product line scoping*" e "*product line planning*". Foram pesquisados trabalhos a partir do ano de 2006. A tabela 1 apresenta o número de artigos encontrados nas respectivas bases científicas.

TABLE I. QUANTIDADE DE ARTIGOS ENCONTRADOS EM BASES CIENTÍFICAS

	Scirus	**IEEE Xplore**	**ACM**	**Scopus**	**Science Direct**
"product line scoping"	42	43	52	12	7
"product line planning"	28	20	4	9	6
TOTAL	70	63	56	21	13

A pesquisa retornou 223 documentos, sendo excluídos os documentos com títulos duplicados e aqueles que não eram artigos científicos, resultando em 151 artigos científicos. Os resumos destes artigos foram lidos, sendo excluídos os artigos que não tinham como foco a atividade de definição de escopo. Em caso de dúvidas na análise do resumo, o artigo completo foi analisado. Ao final desta análise restaram 45 artigos, dos quais oito não estavam disponíveis na íntegra, sendo possível realizar a leitura completa e análise de 37 artigos. Após a leitura completa, apenas 14 artigos apresentavam abordagens para definição de escopo, melhorias ou apoio nesse processo, os quais estão descritos a seguir.

Em [8] é proposta uma abordagem para identificar as pessoas que conhecem os produtos atuais da organização, por meio da análise de log de alterações em código fonte. As atividades descritas neste trabalho são definidas como recuperação de *ownership architecture*, e foram incorporadas como um dos componentes da abordagem PuLSE (*Product Line Software Engineering*). O principal benefício citado para a fase de definição de escopo é a identificação das pessoas que conhecem os produtos atuais da organização e suas implementações, possibilitando sua participação nos workshops e entrevistas realizados nessa fase.

Uma abordagem colaborativa para realizar a definição de escopo é proposta em [9]. O foco do trabalho é a migração de sistemas existentes para uma abordagem de LPS. Para a engenharia de colaboração foi utilizado o padrão ThinkLets e para a definição de escopo foi utilizado o PulSE-Eco (*Product Line Software Engineering – Economic Scoping*). Também foi utilizada a técnica EasyWinWin na negociação de requisitos. Nesta abordagem é necessária a presença dos *stakeholders* para a realização da fase *Product Line Mapping* do Pulse, que é realizado com um ou mais workshops ou entrevistas com os especialistas no domínio e no escopo.

Uma abordagem colaborativa que aplica princípios das metodologias ágeis é proposta em [10]. Nesta abordagem a necessidade da participação dos *stakeholders* em todas as atividades do processo é enfatizada.

Em [11] a prática ágil *planning game* foi adaptada para ser usada com LPS. Os resultados mostraram que a técnica é eficiente para estabelecer um ciclo de *feedback* da engenharia de aplicação para a engenharia da família. A técnica se mostrou complementar ao processo contínuo de definição de escopo. A definição de escopo, neste trabalho, foca principalmente em alterações ou em novos requisitos de clientes externos. O *planning game* em linhas de produto endereça os problemas dos engenheiros de aplicação como clientes internos da engenharia de família, reutilizando os componentes da linha de produtos.

Em [12] foram identificadas na literatura várias abordagens para definição de escopo e foram consideradas como relevantes as abordagens que apresentavam uma relação clara com linhas de produto e alguma maturidade e documentação suficiente para que fossem entendidas e aplicadas. O intervalo de datas de publicação utilizado para a pesquisa não foi informado, porém as referências apresentam trabalhos entre os anos 2000 e 2008. De todas as abordagens encontradas, 16 abordagens foram consideradas relevantes. Estas abordagens foram classificadas de acordo com alguns fatores como: objetivo, entradas e saídas, variabilidade, maturidade, se existem publicações recentes sobre a abordagem, dentre outros. Algumas conclusões citadas em [12] são listadas a seguir:

- as abordagens de definição de escopo podem ser utilizadas para vários objetivos, em basicamente três cenários distintos: definição de escopo na engenharia de LPS, durante a engenharia de requisitos ou para a evolução da LPS;

- os principais artefatos de entrada para a atividade de definição de escopo são descrições dos produtos, do domínio e requisitos;
- a forma de apresentação do resultado da definição de escopo é diversificada, e ainda não está claro como utilizá-la nas fases seguintes;
- não foi identificado um domínio específico de sistemas onde esta atividade fosse mais ou menos apropriada;
- a atividade de definição de escopo envolve todos os *stakeholders* de uma organização.

Identificar comunalidades entre diferentes domínios é o foco do trabalho de [13], porém a identificação também é dependente de pessoas. A abordagem utiliza os resultados da análise de domínio e outros modelos para fazer com que as comunalidades sejam visíveis para os desenvolvedores, por meio da construção de diagramas. Neste trabalho é afirmado que o problema mais difícil é encontrar comunalidades essenciais entre sistemas e guardá-las de maneira sistemática.

A abordagem proposta em [14], usa preferências dos clientes sobre as características dos produtos para gerar múltiplos portfólios de produtos, cada um contendo uma variação de produto por segmento de cliente. O método é para sistemas que já existem e estão evoluindo para uma estrutura de linhas de produto, e também leva em consideração o impacto da evolução na estrutura atual do sistema, analisando sua estrutura para sugerir a variação dos produtos.

A abordagem CAVE (*Commonality And Variability Extraction*) proposta por [15] é uma abordagem manual que utiliza documentação de usuário como fonte para o processo de definição de escopo. Um consultor de linhas de produto seleciona os documentos do sistema que serão utilizados, avalia suas semelhanças e em seguida aplica os padrões definidos para buscar os elementos selecionados (funcionalidade, domínio, elemento opcional, etc.). Após a aplicação dos padrões é possível gerar uma matriz com os elementos encontrados relacionados com os produtos nos quais foram identificados. Essa matriz então é avaliada por um especialista de domínio.

O trabalho apresentado em [16] compara e analisa as abordagens tradicionais para a realização da atividade de definição de escopo para encontrar seus componentes essenciais e desenvolvê-los em uma abordagem única. Os autores afirmam que não existe uma única abordagem que abrange as três categorias existentes, e que não é fácil identificar os pontos comuns, diferentes, fortes e fracos de cada uma delas.

Uma abordagem adicional às abordagens de definição de escopo é apresentada em [17], com o objetivo de auxiliar organizações a identificar quais funcionalidades são mais importantes tendo como objetivo aumentar lucro ou diminuir despesas. A abordagem identifica entidades relevantes que influenciam na rentabilidade da linha e seus relacionamentos, em seguida formula o problema de otimização matematicamente.

Dois artigos encontrados buscam identificar funcionalidades utilizando o código fonte. Em [18] é proposta uma abordagem de engenharia reversa para extrair informações de variabilidade do código fonte de produtos de software similares. O foco do trabalho é auxiliar organizações que falharam na tentativa de construir uma linha de produtos de software de maneira pró-ativa e acabaram construindo um único produto inicial e depois vários clones deste para atender às novas demandas. Esta abordagem assume que os códigos que serão analisados são clones um dos outros, e portanto possuem uma estrutura de código muito similar.

Já em [19] a abordagem proposta utiliza engenharia reversa para criar um diagrama de classes simplificado do código fonte dos produtos e a partir disso decompõe o diagrama em partes menores, chamadas "primitivas de construção", utilizadas para comparar os produtos na segunda etapa e identificar funcionalidades candidatas. Se o nome das classes e métodos que fazem a mesma coisa tiverem nomes diferentes entre os produtos, eles precisam ser uniformizados para que a abordagem possa ser utilizada.

Em [20] é apresentada uma abordagem para analisar variabilidades candidatas a partir do histórico de versão de produtos, considerando que suas variabilidades constam no histórico de alterações. Já em [21] o objetivo é auxiliar na transição de descrição de produtos para modelo de funcionalidades (feature model) de forma semi-automática. As descrições dos produtos devem estar organizadas em tabelas, onde cada linha representa um produto.

Com relação aos artigos que buscavam auxiliar na identificação dos produtos e funcionalidades da linha, identificamos que na maioria das abordagens apresentadas existe a necessidade da presença de um especialista, *stakeholder* ou cliente:

- em [9] é proposta uma abordagem colaborativa, que precisa da presença de especialistas de domínio;
- em [10] a proposta colaborativa é aplicada em metodologias ágeis, e necessita da presença dos *stakeholders* que conheçam o domínio;
- uma das conclusões da pesquisa apresentada em [12] é que após a análise dos papéis envolvidos e requeridos durante a aplicação das abordagens encontradas, se conclui que a atividade de definição de escopo é uma atividade que envolve vários *stakeholders* das organizações;
- em [13], as comunalidades entre diferentes domínios são identificadas para posterior análise pelos desenvolvedores;
- a abordagem proposta em [14], usa preferências dos clientes sobre as características do produtos;
- a abordagem CAVE, proposta por [15], procura diminuir a necessidade da presença dos especialistas, analisando documentação de usuário manualmente por meio de padrões de busca pré-estabelecidos.
- em [18], [19] e [20] as abordagens apoiam os especialistas na identificação das funcionalidades

porém são limitadas pelo contexto em que se aplicam, pois em [18] os códigos analisados precisam ser muito semelhantes (clones), em [19] os elementos como classes e métodos precisam ter os mesmos nomes e em [20] o histórico de alterações precisa existir e ter informações relevantes.

A abordagem CAVE diminui o tempo necessário da presença de um especialista de domínio, porém, possui atividades manuais, que demandam tempo e que estão sujeitas a vários erros por serem executadas por humanos. Um dos trabalhos futuros citados em [15] é a automação dos padrões para prover análise automática de documentos e identificação de artefatos para a linha de produtos, pois a tarefa de aplicar manualmente padrões em documentos grandes é uma tarefa tediosa.

A proposta apresentada neste trabalho busca semi-automatizar a análise de documentos e a identificação de funcionalidades de uma linha de produtos a ser criada a partir de sistemas de software existentes (abordagem extrativa), utilizando técnicas de anotação linguística, visando minimizar a presença constante dos especialistas de domínio durante a fase de definição de escopo.

III. PROPOSTA

A. *Concepção da Proposta*

Para realizar a concepção inicial da proposta, foram identificadas duas etapas principais:

- identificar quais artefatos dos sistemas de software existentes seriam analisados (documentos de caso de uso, código fonte e etc.);
- identificar a maneira de automatizar a busca por padrões nos artefatos selecionados.

Na etapa 1, os artefatos selecionados para serem utilizados nesta proposta foram documentos textuais que contenham informações relacionadas às funcionalidades dos sistemas existentes. Para utilização durante a construção e avaliação da proposta, foi definida a utilização de manuais de usuário, pois estes normalmente descrevem os produtos de maneira orientada a característica [22].

Na etapa 2, foram realizadas simulações com a ferramenta PorOnto[23] que foi desenvolvida para construir ontologias de forma semi-automática utilizando como artefatos de entrada arquivos em formato PDF (*Portable Document Format*). Esta ferramenta foi utilizada porque implementou busca de padrões em textos escritos em língua portuguesa, e é gratuita. Para as simulações realizadas na ferramenta, foram utilizados manuais de celular encontrados gratuitamente na *web*.

A ferramenta PorOnto permite a busca e visualização da frequência de termos simples ou compostos encontrados nos arquivos fornecidos como entrada. Como resultado das simulações realizadas, verificou-se que vários termos compostos que foram identificados nos manuais poderiam ser considerados funcionalidades.

Para identificação destes termos, a ferramenta PorOnto utiliza a ferramenta de anotação linguística TreeTagger [24]. No trabalho apresentado em [23], esta ferramenta foi selecionada após a comparação entre as ferramentas gratuitas de anotação linguística disponíveis na época, portanto uma nova busca foi realizada para definir a ferramenta que seria utilizada nesta proposta.

A única ferramenta mais recente encontrada de anotação linguística para língua portuguesa, gratuita e disponível integralmente para uso foi o LX-Parser [25]. Portanto, foram realizados testes para que fosse possível escolher qual das duas ferramentas seria utilizada na proposta.

Os testes foram realizados com a implementação de um algoritmo que, a partir da leitura de manuais em formato texto (TXT), aplicava a anotação linguística e buscava então termos compostos em cada linha do arquivo anotado, utilizando as anotações que indicavam verbos e substantivos, em qualquer ordem e em qualquer número.

Em um teste realizado com apenas um manual, foram identificados 2260 termos compostos, dentre estes 799 foram encontrados apenas pelo LX-Parser, 777 apenas pelo TreeTagger e 684 foram encontradas pelos dois parsers.

Os resultados foram analisados e testes específicos foram realizados buscando identificar a qualidade da anotação realizada pelas ferramentas, já que a busca pelos termos após a anotação era realizada da mesma maneira. Durante os testes foi identificado que a anotação realizada pelo TreeTagger estava mais correta do que a realizada pelo LX-Parser. O LX-Parser foi desenvolvido e treinado com português europeu, talvez por este motivo seus resultados foram inferiores. Portanto, a ferramenta TreeTagger foi selecionada para utilização nesta proposta.

Após a análise dos resultados dos testes, foi identificada a necessidade de refinar o algoritmo para:

- definir padrões de busca para cada linha anotada pela ferramenta, incluindo o uso de adjetivos;
- retirar caracteres especiais que atrapalham a realização da anotação linguística pela ferramenta;
- ignorar resultados que contenham verbos auxiliares, pois estes não correspondem a funcionalidades;
- identificar a raiz (*stemm*) das palavras anotadas para que funcionalidades escritas em tempos verbais diferentes possam ser comparadas corretamente; e
- identificar verbos sinônimos.

Além dos itens listados acima, relacionados diretamente ao algoritmo, também foi verificado durante os testes a necessidade de "limpar" os arquivos texto utilizados para o processamento. Este processo implica diretamente na qualidade final do processamento, pois dependendo do formato do manual em PDF (imagens entre o texto), sua versão em formato TXT, gerado automaticamente, pode conter problemas que atrapalham a busca pelos padrões no algoritmo, como por exemplo, frases incompletas e palavras separadas por espaço

em branco. Por este motivo o arquivo em formato TXT deve ser pré-processado da seguinte maneira:

- deve conter uma frase por linha (sem necessidade de incluir pontuação);
- não deve possuir palavras separadas incorretamente por espaços em branco ou hifenização;

Para aumentar a velocidade de execução do algoritmo, alguns capítulos dos manuais, que não contêm funcionalidades, podem ser retirados, como por exemplo: apresentações da marca/fabricante, índice, introdução, cuidados gerais (segurança), especificações técnicas, endereços, garantia, acessórios.

B. *Proposta*

O método proposto foi desenvolvido para plataforma web utilizando a linguagem Java, versão 6, através do IDE (*Integrated Development Environment*) Eclipse. O algoritmo desenvolvido é composto por cinco etapas, apresentadas na Fig.1:

1) *Etapa 1 – Pré-processamento*

O usuário informa o diretório onde se encontram os manuais dos produtos que serão utilizados para criação da LPS. Se os manuais estiverem em formato PDF, são transformados para o formato TXT coma biblioteca PDFBox [26] e salvos no mesmo diretório.

2) *Etapa 2 - TreeTagger*

Os manuais transformados em arquivos no formato TXT são processados pelo TreeTagger. Para isto, cada linha lida do arquivo é transformada em uma lista de palavras separadas por espaço em branco (tokenização), utilizando uma biblioteca do LXParser, e são retirados caracteres especiais que prejudicam a anotação linguística realizada, como o parênteses, colchetes, chaves e caracteres como "#@?!". Estes caracteres foram identificados através dos testes realizados durante a seleção da ferramenta de anotação linguística que seria utilizada na proposta.

Os arquivos são então processados pelo TreeTagger. Para cada arquivo processado, outro é gerado e salvo em uma nova

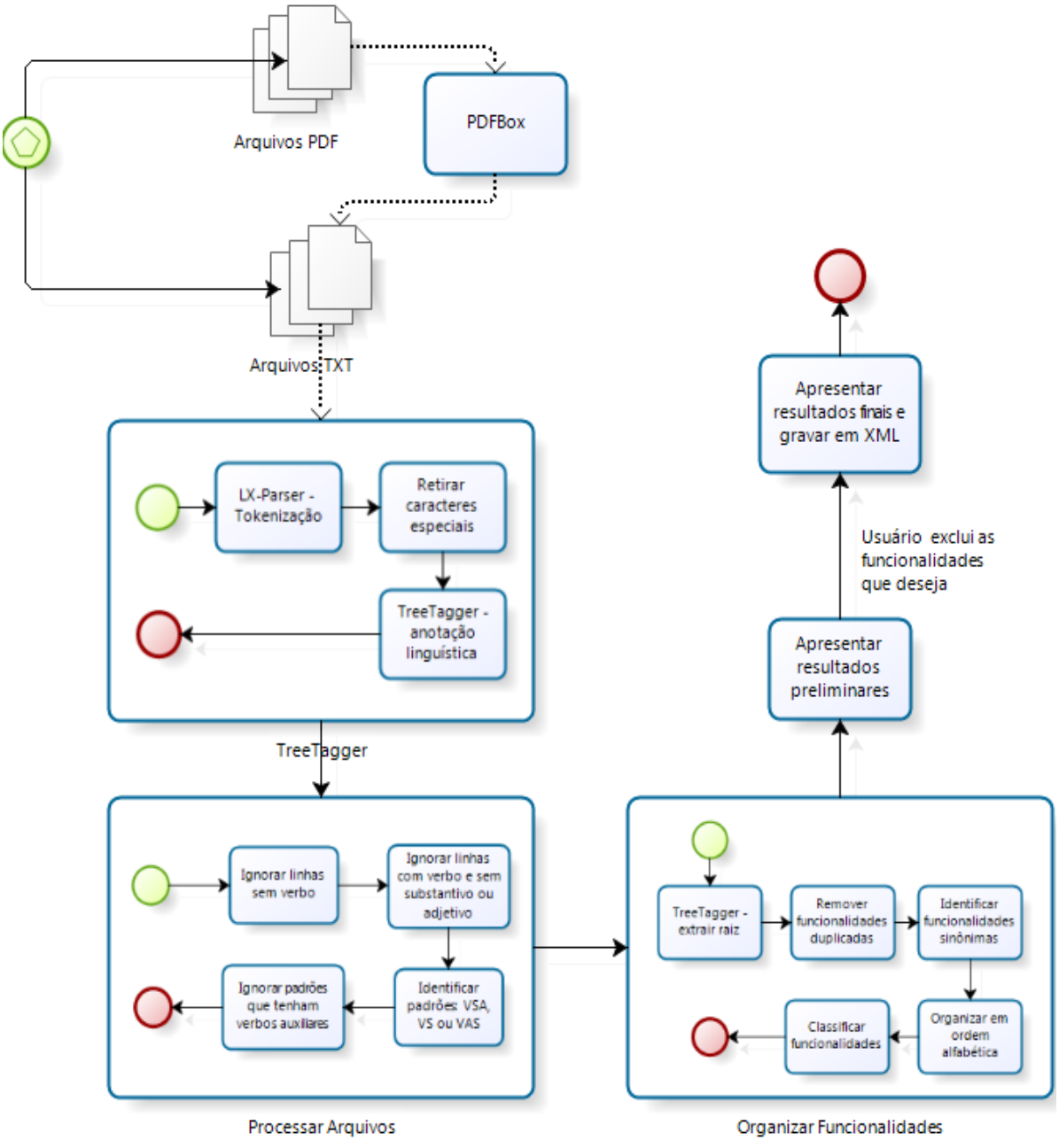

Figure 1. Etapas da abordagem proposta

pasta, contendo as anotações linguísticas geradas pela ferramenta. O TreeTagger realiza a anotação linguística utilizando as tags especificadas na tabela 2.

3) Etapa 3 – Processar Arquivos

Os arquivos que contêm as anotações linguísticas são lidos, linha a linha, em busca de funcionalidades. As linhas que não possuem verbo, ou as que possuem verbo, porém não possuem substantivo ou adjetivo, são ignoradas. As que restam, são processadas com a finalidade de identificar sequências de palavras anotadas da seguinte forma: VERBO + SUBSTANTIVO + ADJETIVO, VERBO + ADJETIVO + SUBSTANTIVO, VERBO + SUBSTANTIVO + SUBSTANTIVO OU VERBO + SUBSTANTIVO. Uma linha do arquivo pode conter nenhuma, uma ou mais sequências válidas. Ou seja, na linha "Para enviar mensagem de texto use o telefone", seriam encontradas duas sequências válidas: "enviar mensagem texto" e "use telefone".

As sequências válidas identificadas são analisadas novamente, para verificar quais delas estão relacionadas à verbos auxiliares, sendo então descartadas. A lista de verbos auxiliares utilizada foi retirada do site [27].

TABLE II. TAGS DO TREE TAGGER

Tag	Descrição
ADJ	Adjetivo
ADV	Advérbio
DET	Determinante
CARD	Número cardinal / ordinal
NOM	Nome Comum / Próprio
P	Pronome
PREP	Preposição
V	Verbo
I	Interjeição
VIRG	Separadores dentro da oração
SENT	Separadores de orações

4) Etapa 4 – Organizar Funcionalidades

As funcionalidades identificadas são processadas novamente pelo TreeTagger, desta vez com outra finalidade: identificar a raiz de cada palavra, para que seja possível a comparação de funcionalidades que estejam relacionadas ao mesmo substantivo, porém este aparece de formas diferentes, como por exemplo: "enviar mensagem" e "enviar mensagens".

Este processamento também possibilita a avaliação dos verbos sinônimos. As funcionalidades são organizadas de acordo com o substantivo contido, ao mesmo tempo em que as funcionalidades repetidas são removidas e as funcionalidades sinônimas são identificadas. Os verbos sinônimos utilizados nesta etapa foram retirados do dicionário de sinônimos OpenThesaurusPT [28]. O nome dos arquivos processados que continham as funcionalidades são armazenados para que as funcionalidades possam ser classificadas dentro da família. Por fim, as funcionalidades são organizadas em ordem alfabética.

5) Etapa 5 – Apresentar Resultados

As funcionalidades são apresentadas ao usuário, com suas respectivas funcionalidades sinônimas, quando aplicável, e com sua classificação identificada, que foi realizada com base na definição de [4] que afirma que a variabilidade pode ser separada em três tipos principais: comunalidades, o que é comum para todos os produtos; variabilidades, comum apenas a alguns produtos da família; específico de produto, normalmente não é necessário para o negócio. Este último tipo pode não ser integrado no conjunto de ativos da família.

As funcionalidades, neste trabalho, foram classificadas em comuns, variáveis ou opcionais. As funcionalidades comuns são as que foram encontradas em todos os produtos, as variáveis foram encontradas em mais de um produto, porém não em todos, e as opcionais foram encontradas apenas em um produto. Elas são identificadas, pelas cores verde (comuns), amarela (variáveis) e vermelha (opcionais).

O usuário avalia as funcionalidades identificadas e tem a possibilidade de excluir as que desejar e de sinalizar funcionalidades sinônimas. As funcionalidades são então apresentadas novamente ao usuário, com um mecanismo de "filtragem" conforme apresentado na Fig.2, para facilitar sua visualização, e são gravadas em um arquivo de formato XML (*eXtensible Markup Language*) para proporcionar sua

FUNCIONALIDADES IDENTIFICADAS

Substantivo	Funcionalidade raiz (Funcionalidade original)	Classificação	Produto
CONTATO	apagar contato (apagar contatos)	COMUM	LG KB775 LG Optimus 2X P990 Samsung Corby
CONTATO	chamar contato (chamar contato)	VARIÁVEL	LG Optimus 2X P990
MENSAGEM	enviar mensagem (enviar mensagem)	OPCIONAL	LG Optimus 2X P990 Samsung Corby

Figure 2. Lista das funcionalidades identificadas e classificadas

reutilização e distribuição.

IV. AVALIAÇÃO DA PROPOSTA

Para avaliar a proposta foram processadas três páginas de um manual de celular retirado da *web*, relacionadas às funcionalidades para Contatos. O processamento foi manual e também automático, com uso do algoritmo proposto, para ser possível realizar uma comparação.

O processamento manual identificou 41 funcionalidades em aproximadamente 8 minutos. As mesmas páginas foram processadas pelo algoritmo proposto e apresentadas ao usuário após 9 segundos de processamento, sem considerar o tempo para converter o arquivo em TXT. Para este experimento o arquivo foi pré-processado manualmente para conter apenas o capítulo de Contatos e também foi "limpo" de acordo com os itens informados anteriormente na seção III:

- deve conter uma frase por linha (sem necessidade de incluir pontuação);
- não deve possuir palavras separadas incorretamente por espaços em branco ou hifenização;

O algoritmo encontrou 106 funcionalidades, divididas entre 42 substantivos. As funcionalidades foram todas apresentadas ao usuário, que demorou 04 minutos e 30 segundos para excluir 45 funcionalidades que não era relevantes, restando 61 funcionalidades relevantes.

Entre estas 61 funcionalidades relevantes, 26 também haviam sido encontradas na busca manual e 35 foram localizadas apenas pelo algoritmo e consideradas relevantes pelo usuário.

Das 41 funcionalidades encontradas na busca manual, apenas 14 não foram localizadas pelo algoritmo. Estas funcionalidades estão listadas na tabela 3, assim como a frase do arquivo onde cada funcionalidade deveria ter sido encontrada, com suas respectivas anotações linguísticas.

TABLE III. FUNCIONALIDADES NÃO ENCONTRADAS NO ALGORITMO

Funcionalidade	Frase com anotaçãolinguística
1 - Chamar contato	(P a)(V partir)(PRP+DET da)(NOM lista)(V toque)(DET o)(NOM contato)(PR que)(V deseja)(V chamar).
2 - Enviar cartão de visita do contato via mensagem multimídia	(NOM escolha)(V enviar)(CONJ como)(DET uma)(NOM mensagem)(PRP de)(NOM texto)(NOM mensagem)(ADJ multimídia)(V usando)(DET o)(F email)(CONJ ou)(V via)(NOM bluetooth).
3 - Enviar cartão de visita do contato via email	(NOM escolha)(V enviar)(CONJ como)(DET uma)(NOM mensagem)(PRP de)(NOM texto)(NOM mensagem)(ADJ multimídia)(V usando)(DET o)(F email)(CONJ ou)(V via)(NOM bluetooth).
4 - Procurar contato	(V procurar)(PRP por)(QUOTE -)(V permite)(V buscar)(V digitando)(DET o)(NOM número)(CONJ ou)(NOM grupo). No entanto, outra funcionalidade encontrada se refere à busca de contatos pelo grupo ou pelo número do telefone: "digitar número grupo (digitando número grupo)".
5 - Configurar lista de contatos para gravar no telefone	(NOM escolha)(P se)(V quer)(V ver)(DET os)(NOM contatos)(ADJ salvos)(PRP+DET no)(NOM telefone)(CONJ e)(PRP+DET no)(ADV sim)(ADV somente)(PRP+DET no)(NOM telefone)(CONJ ou)(ADV somente)(PRP+DET no)(ADV sim). No entanto, outras funcionalidades encontradas se referem à configuração da lista de contatos: "adaptar configuração contato (adaptar configurações contatos), alterar configuração contato (alterando configurações contato), configurar listar contato (configurar lista contatos)".
6- Configurar lista de contatos para gravar no cartão SIM	
7 - Configurar lista de contatos para gravar no telefone e no cartão SIM	
8 - Configurar lista de contatos para mostrar o primeiro nome	(P você)(ADV também)(V pode)(V selecionar)(V mostrar)(DET o)(CARD primeiro)(CONJ ou)(DET o)(ADJ último)(NOM nome)(PRP+DET do)(NOM contato). No entanto, outra funcionalidade encontrada se refere à esse tipo de configuração: "mostrar último nome (mostrar último nome)".
9 - Copiar contatos entre o cartão SIM e o telefone	(V copiar)(QUOTE -)(V copia)(ADJ seus)(NOM contatos)(PRP+DET do)(ADV sim)(PRP para)(DET o)(ADJ seu)(NOM dispositivo)(CONJ ou)(PRP+DET do)(ADJ seu)(NOM dispositivo)(PRP para)(DET o)(ADV sim).
10 - Mover contatos entre o cartão SIM e o telefone	(V mover)(QUOTE -)(P este)(V trabalha)(PRP+DET da)(ADJ mesma)(NOM forma)(PR que)(V copiar)(CONJ mas)(DET o)(NOM contato)(V será)(ADJ salvo)(ADV somente)(PRP+DET na)(NOM localidade)(PRP para)(DET o)(PR qual)(V for)(V movido)(PRP por)(NOM exemplo)(P se)(P você)(V mover)(DET o)(NOM contato)(PRP+DET do)(ADV sim)(PRP para)(DET o)(NOM telefone)(P ele)(V será)(V apagado)(PRP+DET da)(NOM memória)(PRP+DET do)(X sm).
11 - Enviar contatos via bluetooth	(V enviar)(ADJ todos)(DET os)(NOM contatos)(PRP por)(NOM bluetooth).
12 - Fazer cópia de segurança dos contatos para o telefone	(V permite)(V fazer)(DET uma)(NOM cópia)(PRP de)(ADJ todos)(DET os)(ADJ seus)(NOM contatos)(PRP para)(DET o)(NOM telefone)(CONJ ou)(PRP para)(DET o)(ADV sim). No entanto, outra funcionalidade encontrara se refere à esta: "fazer cópia todo (fazer cópia todos)".
13 - Fazer cópia de segurança dos contatos para o cartão SIM	
14 - Apagar todos os contatos	(V apagar)(NOM contatos)(QUOTE -)(V apagar)(ADJ todos)(DET os)(ADJ seus)(NOM contatos).

Analisando as 14 funcionalidades não encontradas no processamento automático identifica-se que, apesar de algumas não terem sido encontradas exatamente com o texto esperado, outras semelhantes foram identificadas. Isso foi identificado em 7 funcionalidades (50%): funcionalidades de número 4, 5, 6, 7, 8, 12 e 13 na tabela 3. Todas as 14 funcionalidades não foram identificadas no processamento automático pelo mesmo motivo, o estilo de escrita das frases não se encaixou em nenhum dos padrões procurados.

O desempenho da abordagem proposta foi analisado com as medidas de *precision* e *recall* [29] que são definidas pelas fórmulas (1) e (2):

$$Precision = tp / (tp + fp) \quad (1)$$

$$Recall = tp / (tp + fn) \quad (2)$$

Nas fórmulas, tp significa *true positive* (funcionalidades recuperadas relevantes), fp significa *false positive*

(funcionalidades recuperadas não-relevantes), e fn significa *false negative* (funcionalidades não recuperadas relevantes).

O *precision* é a fração das funcionalidades recuperadas que são relevantes, o *recall* é a fração das funcionalidades relevantes que foram recuperadas [29].

Para calcular as medidas serão considerados os valores tp = 61 (funcionalidades relevantes consideradas pelo usuário), fp = 45 (fucionalidades recuperadas e excluídas pelo usuário), fn = 14 (funcionalidades relavantes encontradas pelo usuário, porém não encontradas pelo algoritmo).

Os resultados das medidas são *precision* = 0,57 e *recall* = 0,81, ou seja, dos resultados recuperados 81% são relevantes para o critério pesquisado e 57% são precisos, isto é, segundo um avaliador humano, a resposta seria considerada correta para o critério pesquisado.

A partir destas medidas também pode-se extrair a medida *F-measure*, que é uma média harmônica entre as medidas *precision* e *recall*, onde quanto mais próximo de 1 for o resultado, melhor. A fórmula da medida *F-measure* é definida na fórmula (3), onde *r = recall* e *p = precision*:

$$F\text{-}measure = 2 * r * p / r + p. \quad (3)$$

O resultado da medida *F-measure* é de 0,67.

Para melhorar as medidas *precision*, *recall* e *F-measure*, é possível alterar o algoritmo para levar em consideração, na identificação das funcionalidades, apenas aquelas que tenham uma palavra específica considerada como substantivo ou adjetivo pelo TreeTagger.

Fazendo uma simulação com os resultados do processamento manual e automático realizado neste artigo, considerando para o processamento automático apenas as funcionalidades que apresentaram como substantivo ou como adjetivo a palavra Contato, teríamos os seguintes valores: tp = 26, fp = 10 e fn = 14 (idem anterior), resultando em *precision* = 0,72; *recall* = 0,65; *e F-Measure* = 0,68.

O *recall* diminuiu porque o número de funcionalidades relevantes identificadas foi reduzido em comparação ao cálculo anterior, no entanto o número de funcionalidades relevantes que não foram identificadas permaneceu o mesmo.

V. CONCLUSÕES

Os resultados apresentados demonstram que é possível semi-automatizar a análise de documentos na busca por funcionalidades de produtos que serão utilizadas posteriormente para a criação de uma LPS.

Todas as abordagens de definição de escopo identificadas necessitam da presença do especialista do domínio no qual será criada a LPS, porém apenas o trabalho proposto em [15] procurou minimizar o tempo gasto por este profissional.

No trabalho proposto em [15], a análise dos documentos pode ser realizada por outros profissionais e não necessariamente especialistas no domínio em questão, porém o especialista precisa validar esta análise executada, e como ela é totalmente manual, está sujeita a erros humanos e retrabalho.

A abordagem proposta neste artigo mostrou que é possível minimizar ainda mais o tempo gasto pelo especialista, pois demonstrou ser possível recuperar as funcionalidades a partir de documentos, como por exemplo, manuais de usuário, diminuindo o tempo gasto na análise de documentos.

A qualidade da recuperação das funcionalidades está diretamente relacionada à qualidade dos arquivos que serão processados (limpeza) e do estilo de escrita para que os padrões sejam identificados. Considerando que o processamento realizado em apenas 3 páginas retornou 106 funcionalidades e o manual processado possui 65 páginas (já desconsiderando os capítulos como introdução, índices e etc. conforme citado na seção III deste artigo), pode-se concluir que é necessária a realização de ajustes no algoritmo para melhores resultados, pois o número de funcionalidades identificadas no manual como um todo, considerando a média encontrada em 3 páginas, seria de mais de 2200 funcionalidades.

Dentre os possíveis ajustes é possível identificar a necessidade de revisão dos padrões de busca e a avaliação dos benefícios de incluir no algoritmo uma lista de "*stop features"*, que seria populada cada vez que o usuário exclui funcionalidades irrelevantes durante o processo já existente. Desta forma, as funcionalidades que estiverem na lista de "*stop features*" seriam ignoradas e não seriam apresentadas ao usuário, diminuindo ainda mais o tempo de análise dos resultados e possibilitando uma espécie de treinamento do algoritmo para determinado domínio, melhorando os resultados finais. As principais contribuições esperadas da abordagem proposta são:

- auxiliar organizações que desejam migrar para a abordagem de LPS a iniciar o mapeamento de seus produtos, e visualizá-los como famílias de um mesmo domínio que compartilham componentes vistos como funcionalidades em comum. A partir desta visualização é possível iniciar o planejamento da arquitetura da linha, e identificar o reuso proporcionado pela linha; e
- proporcionar uma maneira única de apresentar a família de produtos de uma forma visível a humanos e máquinas, possibilitando processamentos automáticos.

REFERÊNCIAS

[1] I. Sommerville. "Engenharia de Software", 8ª edição. São Paulo: Pearson Addison-Wesley, 2007. 552 p.

[2] I. M. Gimenes andG. H. Travassos, "O enfoque de Linha de Produto para Desenvolvimento de Software." In: XXI JAI - Livro Texto ed.Florianopolis : Sociedade Brasileira de Computação, 2002.

[3] P. Clements and L. Northrop, "Software Product Lines: Practices and Patterns". Boston: Addison-Wesley, 2002, 563 p.

[4] F. Linden,K. Schimid and E. Rommes,"Software Product Lines in Action". Springer, 2007.

[5] V. Alves, N. Niu, C. Alves andG. Valença,"Requirements Engineering for Software Product Lines: A Systematic Literature Review."Information and Software Technology, v. 52, agosto 2010.

[6] I. John and M. Eisenbarth, "A Decade of Scoping: A Survey." In: Proceedings of the 13th International Software Product Line Conference, 1., 2009, Airport Marriott, San Francisco, CA, USA. Anais… Pittsburgh, 2009, p. 31-40.

[7] SEI – SOFTWARE ENGINEERING INSTITUTE. "A Framework for Software Product Line Practice", Version 5.0 - Scoping. Disponível em:

<http://www.sei.cmu.edu/productlines/frame_report/productLS.htm>. Acesso em 23 mar. 2011.

[8] D. Ganesan, D. Muthig, J. Knodel andD. Rose, "Discovering Organizational Aspects from the Source Code History Log during the Product Line Planning Phase – A Case Study." IEEE International Working Conference on Reverse Engineering (WCRE 2006), Villa dei Papi: 2006, p. 211 -220.

[9] M. A. Noor, P. Grünbacher and R. O. Briggs, "A collaborative approach for Product Line Scoping : a case study in collaboration engineering." In: Proceedings of the 25th IASTED International Multi-Conference. Anais… Innsbruck, Austria: 2007, p. 216 - 223.

[10] M. Noor, R. Rabiser, and P. Grunbacher,"Agile product line planning: A collaborative approach and a case study." The Journal of Systems and Software, vol. 81, Jun. 2008, p. 868-882.

[11] R. Carbon, J. Knodel, D. Muthig and G. Meier," Providing Feedback from Application to Family Engineering - The Product Line Planning Game at the Testo AG." In: 12th International Software Product Line Conference. Anais… Limerick, Ireland: Ieee, 2008, p. 180-189.

[12] I. John, and M. Eisenbarth,"A Decade of Scoping: A Survey." In: Proceedings of the 13th International Software Product Line Conference, 1., 2009, Airport Marriott, San Francisco, CA, USA. Anais… Pittsburgh, 2009, p. 31-40.

[13] Y. Liu, K. Nguyen, M; Witten and K. Reed, "Cross Product Line Reuse in Component-based Software Engineering." In: 2010 International Conference on Computer Application and System Modeling (ICCASM 2010). Anais… Taiyuan, China: 2010, p. 427-434.

[14] M. I. Ullah, G. Ruhe and V. Garousi,"Decision support for moving from a single product to a product portfolio in evolving software systems." The Journal of Systems and Software, vol. 83, Dec. 2010, p. 2496-2512.

[15] I. John, "Using Documentation for Product Line Scoping." IEEE Software, vol. 27, 2010, p. 42 - 47.

[16] J. Lee, S. Kang and D. H. Lee, "A Comparison of Software Product Line Scoping Approaches," International Journal of Software Engineering and Knowledge Engineering, Vol. 20, Issue 5, pp. 637-663, World Scientific, October 2010.

[17] J. Muller, "Value-Based Portfolio Optimization for Software Product Lines", Software Product Line Conference (SPLC), 2011 15th International, pp.15-24, 22-26 Aug. 2011.

[18] S. Duszynski, "A scalable goal-oriented approach to software variability recovery", In: Software Product Lines - 15th International Conference, SPLC 2011, Munich, Germany, August 22-26, 2011.

[19] T. Ziadi, L. Frias, M.A.A. da Silva and M. Ziane, "Feature Identification from the Source Code of Product Variants", Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on, pp.417-422.

[20] K. Yoshimura, F. Narisawa, K. Hashimoto, and T. Kikuno, "A Method to Analyze Variability Based on Product Release History: Case Study of Automotive System", in Proc. SPLC (2), 2008, pp.249-256.

[21] M. Acher et al., "On extracting feature models from product descriptions." In Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems (VaMoS '12). ACM, New York, NY, USA, pp. 45-54.

[22] I. John, J. Knodel, T. Lehner and D. Muthig, "A Practical Guide to Product Line Scoping." In: Software Product Lines: Proceedings of the 10th International Software Product Line Conference (SPLC 2006). Anais… Baltimore, Maryland, August 21-24, 2006.

[23] F. M. Zahra andA. Malucelli, "Poronto : ferramenta para construção semiautomática de ontologias em português."Dissertação (Mestrado) - Pontifícia Universidade Católica do Paraná, Curitiba, 2009. 94 f.

[24] H. Schmid, "Probabilistic Part-of-Speech Tagging Using Decision Trees." In: Proceedings of the International Conference on New Methods in Language Processing. Anais… Manchester, UK, 1994.

[25] J. Silva, A. Branco, S. Castro and R. Reis, "Out-of-the-Box Robust Parsing of Portuguese." In Proceedings of the 9th International Conference on the Computational Processing of Portuguese (PROPOR'10), pp. 75–85.

[26] The Apache Software Foundation, "Apache PDFBox – Java PDF Library". Disponível em: <http://pdfbox.apache.org/>. Acesso em 01. Mar. 2012.

[27] Priberam Informárica. "Dicionário Priberam da Línrua Portuguesa. " Disponível em: <http://www.priberam.pt/dlpo/Conjugar.aspx?pal=ser>. Acesso em 01.mar.2012.

[28] OpenThesaurusPT. "Dicionário de Sinônimos para a língua portuguesa." Dispomível em: <openthesaurus.caixamagica.pt>. Acesso em 01. mar. 2012.

[29] C. D. Manning, P. Raghavan and H. Schütze,"An Introduction to Information Retrieval". Cambridge University Press, 2009.