CompogeMatch: Recuperação de modelos conceituais usando padrões de análise

Kuesley Fernandes do Nascimento
Universidade Federal de Santa Catarina – UFSC
Programa de Pós Graduação em Ciência da Computação – PPGCC
Florianópolis - SC, Brasil, 88.040-900
kuesley@inf.ufsc.br

e

Raul Sidnei Wazlawick
Universidade Federal de Santa Catarina – UFSC
Departamento de Informática e Estatística – INE
Florianópolis - SC, Brasil, 88.040-900
raul@inf.ufsc.br

Abstract

This paper presents the method called CompogeMatch that detects analysis patterns (APs) in conceptual models using ontology. The purpose of detecting such patterns is to help indexing conceptual models in order to make the search for software components or software in general more fast and effective. The identification of APs allows this method to infer concept equivalence or subsumption even when the lexical term used in a conceptual model is mapped into the ontology. This is the main advantage of this method among others used to retrieve conceptual models. A comparison between this method and keyword retrieval is also showed made at the end of the paper.

Keywords: Software Engineering, Conceptual Model, Ontology, Development Software Based Components, Analysis Pattterns.

Resumo

Este artigo apresenta o método CompogeMatch para detectar padrões de análise (PAs) em modelos conceituais usando ontologia. O objetivo de detectar os PAs é permitir a indexação dos modelos conceituais para realizar buscas mais rápidas e efetivas por componentes de software ou software em geral. A identificação de APs permite esse método inferir equivalência conceitual, ou generalização desde que o termo léxico utilizado em um modelo conceitual esteja mapeado na ontologia. Esta é a principal vantagem deste método entre os outros métodos utilizados para recuperação de modelos conceituais. Uma comparação entre o ComponeMatch e a busca por palavras chave é apresentada na seção de recuperação por PA no final deste artigo.

Palavras chaves: Engenharia de Software, Modelo Conceitual, Ontologia, Desenvolvimento Baseado em Componentes, Padrões de Análise.

1 Introdução

Realizar buscas por software em bibliotecas de componentes ou na Internet tem sido objeto de estudo, uma vez que isso pode contribuir para o compartilhamento, e, por conseqüência, o reuso de software [1] [2] [7] [11] [12]. Existem diversas técnicas de recuperação e medidas de similaridade de software. Um dos métodos que tem se mostrado útil é a utilização do modelo conceitual como fonte de consulta. O modelo conceitual de um software representa as informações do mundo real modeladas dentro do sistema [6]. Dessa forma, a busca pode ser realizada no nível conceitual das informações - entidades do mundo real, sem levar em consideração a tecnologia usada.

Para um mecanismo de busca recuperar informações dos sistemas a partir de seus modelos conceituais, utiliza-se, por exemplo, o nome dos conceitos como filtro. Isso é chamado de *busca por palavra chave.* Para isso, existem ferramentas de processamento de linguagem natural como GURU [12] e ROSA [7] que realizam essa recuperação por software. Outra técnica utilizada é o mapeamento de códigos fonte em gráficos conceituais apresentados por Mishne e Rijke (2006) para medir similaridade entre sistemas. Este trabalho apresenta o método CompogeMatch mostrando como realizar buscas por modelos conceituais (e por conseqüência componentes e sistemas), utilizando como índice de busca os padrões de análise (PAs). Para isso é utilizada uma ontologia para fazer o mapeamento entre os nomes dos conceitos e os PAs. O CompogeMatch foi escrito em Java (versão 5) e está disponível para verificação no endereço http://java.inf.ufsc.br/compoge/compogematch.htm. Para verificar seu funcionamento, basta submeter um modelo conceitual em formato XMI que o método apontará os padrões de análise encontrados no modelo submetido.

Este artigo está organizado da seguinte forma: na seção 2 será apresentado os PAs e o método CompogeMatch e como ele utiliza a ontologia e os PAs para fazer a busca nos modelos de software. Na seção 3 são apresentados os experimentos e resultados obtidos para a validação do método e a seção 4 apresenta as conclusões deste artigo.

2 Usando o CompogeMatch para Detectar Padrões de Análise

O modelo conceitual de um software contém as informações manipuladas pelo sistema. Se o sistema for desenvolvido para a área empresarial, por exemplo, é provável que termos como *cliente, produto, loja, estoque* estejam presentes no modelo conceitual. De qualquer forma, o levantamento de requisitos é que vai definir quais são os conceitos presentes no modelo conceitual. Analistas podem ou não criar o modelo conceitual orientado pelos PAs [6]. Se essa orientação não existir, os modelos produzidos podem conter diferenças morfológicas. Além disso, outro problema encontrado é a diferença nas identificações dos conceitos (sinônimos nos nomes dos conceitos). Esses diferenças tornam mais complexas as comparações entre modelos conceituais. A figura 1 mostra dois modelos conceituais para o mesmo domínio de negócio:

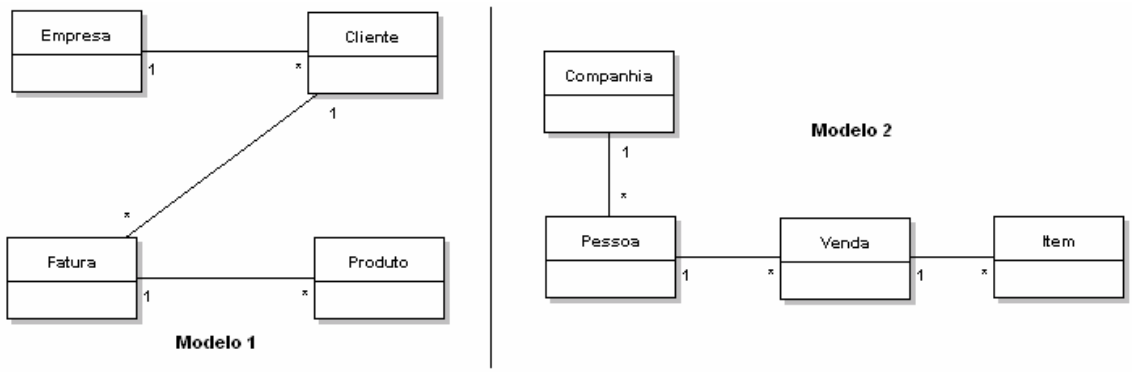


Figura 1: Modelos Conceituais Semelhantes

Note que para o mesmo domínio, neste caso, *venda para clientes*, ocorreu uma diferença entre os modelos conceituais. A recuperação desses modelos em um banco de dados através de uma palavra chave, seria ineficaz, a não ser que houvesse um mapeamento entre os conceitos equivalentes. Por exemplo, se a busca fosse feita utilizando a palavra chave *empresa* somente o *primeiro* modelo conceitual retornaria da busca. Enquanto que o ideal seria que os dois modelos retornassem na busca, pois tratam do mesmo domínio e os conceitos E*mpresa* e *Companhia* são equivalentes. A próxima seção apresenta três padrões de análise.

2.1 Padrões de Análise

Os padrões de análise foram apresentados, catalogados e classificados por Fowler (1997) em diversas áreas de negócios (Comércio, Estruturas, Suporte, etc.). Este artigo aborda apenas alguns padrões de análise utilizados na área comercial (*Party, Organization Hierarchies* e *Contract*). A figura 2 apresenta o padrão *Party*:

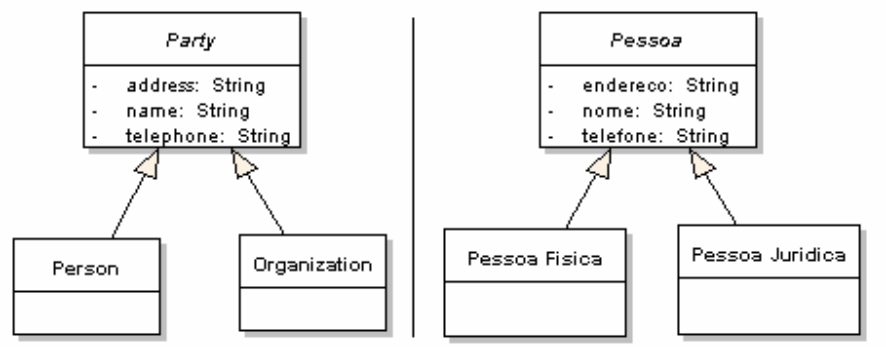


Figura 2: O padrão Party (Pessoa) original e em português

Aqui cabe ressaltar que o padrão *Party* (Pessoa) não é composto por três conceitos. Como a classe Party é abstrata o padrão deverá ser uma *Person (Pessoa Física) ou* uma *Organization* (*Pessoa Jurídica*), e *Party* é uma generalização desses dois conceitos.

A figura 3 mostra o padrão *Organization Hierarchies:*

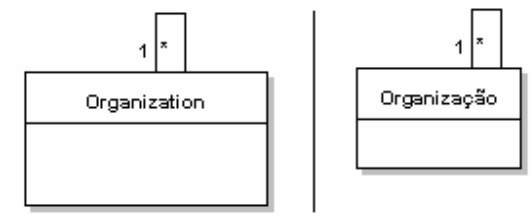


Figura 3: O padrão Organization Hierarchies (Hierarquia Organizacional) original e em português

Este padrão aparece com freqüência nos domínios de governo eletrônico ou ambientes coorporativos quando as organizações (empresas) necessitam representar suas unidades, divisões ou departamentos.

A figura 4 apresenta o padrão *Contract* (Contrato):

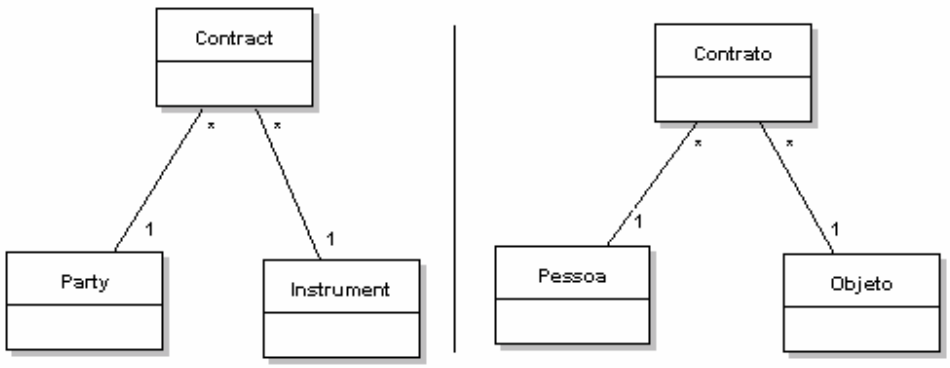


Figura 4: O padrão Contract original e em português

O padrão *Contract* aparece com freqüência em domínios comerciais por representar as informações de um contrato entre uma *Empresa* e uma *Pessoa* (*Party*). Para a detecção do padrão *Contract* é necessária a associação de três classes como mostrado na figura 4. Além disso, o padrão *Contract* exige a ocorrência concomitante do padrão Party.

A próxima seção mostra como o método CompogeMatch detecta esses padrões de análise em modelos conceituais.

2.2 Usando Ontologias para Detectar um Padrão

Como apresentado na seção 1, medir a similaridade por palavras chaves não resolve por exemplo, a identificação de sinônimos [4] [7] [11] [12] se não existir mapeamento prévio. Esse mapeamento semântico entre os termos sinônimos no método CompogeMatch é feito por uma ontologia. Por exemplo, se em modelos conceituais aparecerem os conceitos *Empresa*, *Companhia,* pode-se considerar que esses conceitos identificam as mesmas coisas no mundo real. Isso foi mostrado na figura 1. Neste caso, a ontologia teria que ter uma estrutura e relação parecida com a apresentada na figura 5 para possibilitar o relacionamento de equivalência desses termos.

Figura 5: Relação dos termos Empresa, Loja e Companhia na ontologia

A estrutura da ontologia acima suprimiu vários termos para simplificar a ilustração. Os conceitos *Empresa e Companhia* são subconceitos de *Pessoa Jurídica* que é subconceito de *Pessoa.* Dessa forma, esses três conceitos (*Empresa, Loja e Companhia*) são por transitividade subconceitos de *Pessoa*.

Um ponto que precisa ser ressaltado é que o critério utilizado para considerar que os termos *Empresa, Loja e Companhia* são equivalentes, é que a ontologia utilizada pelo método tem o objetivo de mapear os termos dos modelos conceituais com os padrões de análise, dessa forma, em outros contextos essa equivalência poderia não ser considerada adequada.

Para que o método possa inferir que os conceitos *Empresa e Companhia* são equivalentes basta verificar se eles têm um *super conceito* comum.

Além da correspondência de sinônimos dentro das estruturas e termos, a ontologia vai permitir identificar as relações *É UM* (generalização ou especialização). Com isso, é possível perguntar para a ontologia, por exemplo, se uma *Loja "é uma" Empresa* (verdadeiro) ou se uma *Loja "é uma" Pessoa Física* (falso).

Depois de tornar possível a detecção das associações e relações mecionadas anteriormente, a ontologia ainda deverá ser anotada com algumas propriedades para permitir a detecção dos padrões de análise. A figura 6 apresenta três propriedades que devem existir na ontologia para a detecção dos padrões *Party*, *Organization Hierarchies* e *Contract*:

Figura 6: Exemplo da Ontologia com a anotação das propriedades

As propriedades só precisam ser associadas com o conceito mais alto da hierarquia de termos da ontologia (super conceito). Neste casso *pessoa, lugar e objeto.* O método fará uma busca recursiva pelos conceitos para identificar se a propriedade ocorre em algum termo mais acima. Dessa forma, se for preciso saber se o conceito *Loja* tem a

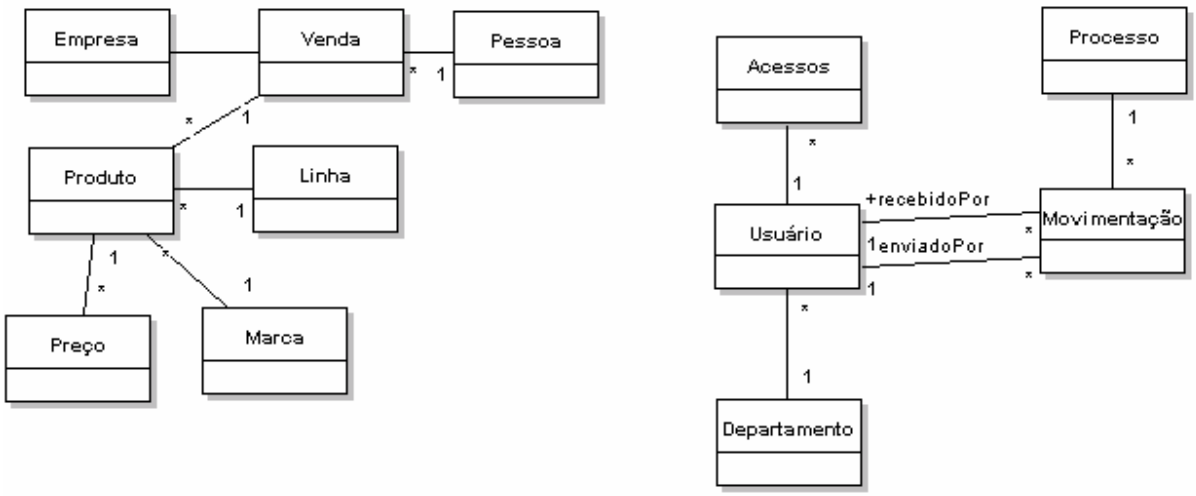
propriedade *ehPessoa* o método primeiro verifica o conceito *Loja*, se não ocorrer, sobe um nível até o conceito *Pessoa Jurídica* e faz a mesma pregunta. Se não encontrar a propriedade *ehPessoa* continua subindo até *Pessoa* quando vai encontrar a propriedade procurada *(ehPessoa)*. O pseudocódigo da figura 7 mostra como isso funciona:

```
funcao ConceitoTemPropriedade(NomeDoConceito: String; NomeDaPropriedade: Propriedade): Boolean
var
con: Conceito;
inicio
con := Ontologia.PegarConceitoPorNome(NomeDoConceito);
se NomeDoConceito = “Thing” entao
retorna false
senão
se c.TemPropriedade("ehPessoa") = false entao
retorna ConceitoTemPropriedade (c.NomeDoPai, NomeDaPropriedade);
senão
return true;
fim;

funcao PegarConceitoPorNome(NomeDoConceito: String): Conceito
var
con: Conceito;
inicio
para i todos os conceitos da ontologia faça
if ontologia[i].Nome == NomeDoConceito entao
return ontologia[i];
fim para;
return null;
fim;
```

Figura 7: Pseudo-código de como verificar se um conceito tem uma propriedade

Depois de apresentar as informações relevantes da ontologia que serão utilizadas no processo de detecção dos PAs, assim como as regras de detecção dos padrões, o próximo passo e submeter alguns modelos e verificar os resultados do método CompogeMatch. A figura 8 apresenta três modelos conceituais submetidos ao método:



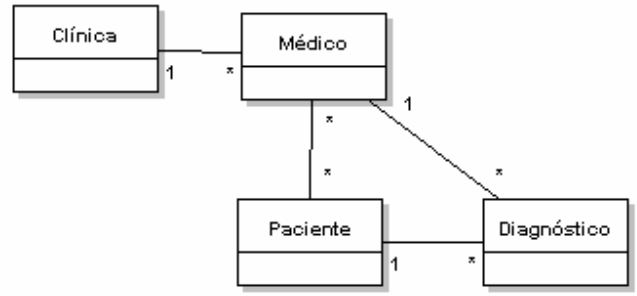


Figura 8: Modelos conceituais ejemplo para execução do método CompogeMatch

O resultado da detecção dos PAs nos modelos conceituais da figura 8 pode ser visualizado na tabela 1:

Tabela 1: Resultado do CompogeMatch em Modelos Conceituais

Conceitos	**Padrões encontrados**
(Empresa, Pessoa, Venda, Produto, Marca, Linha, Preco)	*Organization Hierarchies* [Empresa] *Party* [Pessoa] *Contract* [Pessoa, Venda, Produto]
(Departamento, Usuario, Processo, Movimentacao, Acessos)	*Organization Hierarchies* [Departamento] *Party* [Usuario] *Contract* [Usuario, Movimentacao e Processo]
(Clínica, Paciente, Médico, Diagnóstico)	*Organization Hierarchies* [Clínica] *Party* [Médico, Paciente]

A detecção de PAs em modelos conceituais através do uso de ontologia, torna possível o procedimento inverso, ou seja, a busca de modelos conceituais utilizando como índice de busca os PAs. Dessa forma, basta submeter um padrão de análise, por exemplo, *Party* que será possível a recuperação de modelos que contenham esse padrão.

A seção 3 apresenta alguns experimentos e resultados obtidos utilizando essa técnica de recuperação de modelos conceituais.

3 Recuperação de Modelos Conceituais através de PAs

Com a técnica de detecção de PAs em modelos conceituais, é possível realizar buscas utilizando como filtro os PAs. Esta seção apresenta alguns experimentos realizados com o CompogeMatch para recuperar modelos conceituais através do filtro de AP.

Como parâmetro de comparação foi utilizada uma *busca por palavras chaves* utilizando o método de *comparasion lexical* [11]. Os modelos conceituais utilizados nesses experimentos são software usados por órgãos governamentais e software open source. A tabela 2 apresenta a lista de sistemas que participaram desses experimentos:

Tabela 2: Software utilizado nos experimentos

Software	**Padrões encontrados**
Xplanner	Software Open Source para gerenciamento de projetos que pode ser encontrado no link http://www.xplanner.org
Mantis Bug Tracker	Software Open Source para registro de bug e controle de mudança em projetos de softtware. Link para o projeto: http://www.mantisbt.org/
JurisRBAC	Projeto Open Source para controle de atividades jurídicas para processos do juizado especial. Link para visualização do projeto: http://jurisrbac.codigolivre.org.br/
Protocolo	Software para gerenciamento de documentos eletrônicos. Link http://200.103.18.35:81/protocolo/
Material	Software para controle de estoque de material de expediente e almoxarifado utilizado pelo Governo do Estado de Rondônia.
Procon	Software para atendimento dos consumidores com queixas do procon do Estado de Rondônia. Link para o projeto:

	http://200.103.18.35/procon/
BananaPOS	Software Open Source para ponto de venda em linux. Link para o projeto: http://bananapos.com/pos/home.html

Os modelos conceituais dos sistemas da tabela 2 foram traduzidos para o português para ficar em compatibilidade com a língua da ontologia, pois o método CompogeMatch não faz a tradução automaticamente e a ontologia utilizada nestes experimentos estão em português. Os conceitos dos sistemas onde foram realizados os experimentos são apresentados na tabela 3:

Tabela 3: Conceitos dos sistemas para experimentos

Software	**Conceitos**
Xplanner	[Papel, Estória, Tarefa, Tempo de Execução, Pessoa, Permissão, Iteração, Projeto, Histórico, Atributo, Nota, Identificação, Arquivo, Exemplo de Dados, Integração]
Mantis	[Arquivo, Histórico, Notícias, Erro, Projeto, Patrocinador, Filtros, Configuração, Anotação, Categoria, Versão, Perfil]
JurisRBAC	[Advogados, Assuntos, Audiências, Documentos, Fases, Logs, Partes, Processos, Sentenças]
Protocolo	[Processos, Movimentação, Arquivo Morto, Departamento, Secretaria, Usuário, Perfil, Guia de Remessa, Guia de Distribuição]
Material	[Material, Grupo, Movimentação, Item de Movimentação, Validade, Estoque, Secretaria, Unidade]
Procon	[Consumidor, Fornecedor, Queixa, Carta, Audiência, Notificação]
BananaPOS	[Conta, Endereço, Área, Autoridade, Banco, Conta Corrente, Categoria, Classe, Coleção, Comissão, Companhia, Unidade de Negócio, Contato, Contrato, Pais, Curso, Moeda, Cliente, Entrega, Departamento, Grupo, Desconto, Discrepância, Documento, Empregado, Evento, Inventário, Etiqueta, Livro Razão, Venda, Item da Venda, Pagamento, Produto, Catálogo, Promoção, Reserva, Retorno]

As tabelas 4, 5 e 6 apresentam os resultados da detecção do método CompogeMatch pelo PAs *Party, Organization Hierarchies* e *Contract* respectivamente:

Tabela 4: Resultado da busca pelo padrão *Party*

Software	**Conceito**
Xplanner	Pessoa
Mantis	Patrocinador
JurisRBAC	Advogados
Protocolo	Usuário, Departamento. Secretaria
Material	Secretaria, Unidade
Procon	Consumidor, Fornecedor
BananaPOS	Cliente, Departamento, Empregado

Note que na tabela 4 o padrão *Party* foi encontrado em sete modelos conceituais. No software BananaPOS os conceitos *Cliente, Departamento e Empregado* correspondem ao PA *Party.*

Tabela 5: Resultado da busca pelo padrão *Organization Hierarchies*

Software	**Conceito**
Protocolo	Departamento Secretaria
Material	Secretaria Unidade
BananaPOS	Departamento Unidade de Negócio

Somente em três modelos conceituais (Protocolo, Material e BananaPOS) foram encontradas ocorrências do padrão *Organization Hierarchies*.

Tabela 6: Resultado da busca pelo padrão *Contract*

Software	**Conceito**
Protocolo	[Usuário, Movimentação, Processo]
BananaPOS	[Cliente, Venda, Produto]

O PA *Contract* foi encontrado em apenas dois modelos conceituais (Protocolo e BananaPOS).

A partir dessa detecção foi possível realizar a recuperação considerando o filtro um PA. A recuperação foi realizada da seguinte forma: *(i)* foram submetidos *palavras chave* ao algoritmo que verificou a correspondência lexica; *(ii)* a partir da palavra chave o método CompogeMatch encontrou o PA correspondente; *(iii)* recuperar os modelos conceituais que contenham o PA correspondente a palavra chave submetida.

A tabela 7 apresenta os resultados dessas buscar dos métodos *lexical comparison level* e *CompogeMatch* realizada nos sistemas da tabela 2:

Tabela 7: Resultado das buscas dos métodos
Lexical Comparison Level e CompogeMatch

Filtro	**Lexical Comparison Level**	**CompogeMatch**
Pessoa	Xplanner[Pessoa]	Xplanner[Pessoa] Mantis [Patrocinador] JurisRBAC[Advogados] Protocolo[Usuário] Procon[Consumidor, Fornecedor] BananaPOS[Cliente, Empregado]
Secretaria	Protocolo[Secretaria] Material[Secretaria]	Protocolo[Departamento, Secretaria] Material[Secretaria, Unidade] BananaPOS[Departamento, Unidade de Negócio]

Note que o resultado do método CompogeMatch foi melhor do que o método de comparação lexical, uma vez que o método CompogeMatch realizou a recuperação em duas etapas. A primeira é encontrar o PA a partir da palavra chave, ou seja, dada a palavra chave *Pessoa* qual é o padrão de análise correspondente? Neste caso o padrão seria *Party*. A segunda etapa é recuperar todos os modelos conceituais dos sistemas que contenha o padrão *Party*. Enquanto que a comparação lexical comparou somente a correspondência sintática das palavras.

4 Conclusão

Este trabalho apresentou o método CompogeMatch e como ele recupera modelos conceituais de software utilizando ontologias e padrões de análise. Utilizou-se como filtro de busca os padrões de análise [6]. Dessa forma, as buscas podem ser realizadas de duas formas: a primeira foi a submissão de um padrão de análise para recuperar os modelos que contenham esse padrão; a segunda foi a submissão de uma palavra chave, e a partir dessa palavra chave, deve ser identificado qual padrão de análise está correspondente, e, depois disso os modelos poderiam ser recuperados por este PA.

Além disso, a detecção por padrões de análise e a adoção de uma ontologia para fazer o mapeamento prévio dos termos, evita a falta na recuperação de modelos semelhantes. Por exemplo, se o modelo A tem o conceito *Empregado* e o B tem o conceito *Funcionario* a ontologia indica a correspondência semântica entre esses termos, aparecendo os dois modelos (A e B) em uma busca desse domínio.

Este artigo apresentou duas ferramentas na área de recuperação de software por modelos conceituais. O uso de ontologias para mapear os conceitos dos modelos conceituais em padrões de análise e o uso de PAs como filtro de recuperação de sistemas.

5 Referências

[1] Amin R., Cinnéide M. Ó. e Veale T. (2004) "LASER: A Lexical Approach to Analogy in Software Reuse", 26TH Int. Conf. on Software Engineering, Scotland.

[2] Braga R., Mattoso M. e Werner C. (2001) "The Use of Mediation and Ontology Technologies for Software Component Information Retrieval", R. Braga, M. Mattoso, C. Werner, ACM Symposium on Software Reusability, SSR2001, Toronto, Canada, pp.19-28.

[3] Buchli F. e Nierstrasz O., (2003) “Detecting Software Patterns using Formal Concept Analysis”, Diploma thesis, University of Bern.

[4] Chandresekaran B. Josephson J.R. e Benjamins V.R. (1999) “What are ontologies, and why do we need them?”, IEEE Intelligent Systems, pp. 20-26.

[5] Fernandez E.B. e Yuan X (2000) “Semantic analysis patterns” 19th Int. Conf. on Conceptual Modeling ER2000, pp. 183-195.

[6] Fowler M., Analysis Patterns (1997) Reusable Object Models, Addison Wesley. Gamma, E., Helm, R., Johnson, R.; Vlissides, J. (1995). Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley.

[7] Girardi M.R. and Ibrahim B., Using English to retrieve software. The Journal of Systems and Software, 30(3):249-270, September 1995.

[8] Guarino, N. (1997) “Understanding, building, and using ontologies: a commentary to using explicit ontologies”. Iin KBS development. International Journal of Human and Computer Studies, v. 46, p. 293-310.

[9] H. S. Hamza (2004), “Improving Analysis Patterns Reuse: An Ontological Approach”, OOPSLA.

[10] Maarek. Y. S., Berry D. M. and Kaiser G. E., Guru: Information retrieval for reuse. In Landmark Contribuitions in Software Reuse and Reverse Engineering. Prentice Hall, 1994.

[11] Maedche A. e S. Staab (2002), “Mensuring Similarity between Ontologies”, EKAW, Spain.

[12] Mishne G., Rijke e M., “Source Code Retrieval using Conceptual Similarity”, RIAO 2004, Pittsburgh.