

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO FELHBERG

**INTEROPERABILIDADE SEMÂNTICA DE APLICAÇÕES
COMERCIALMENTE DISPONÍVEIS USANDO ONTOLOGIAS
DE DOMÍNIO**

VITÓRIA
2007

GUSTAVO FELHBERG

**INTEROPERABILIDADE SEMÂNTICA DE APLICAÇÕES
COMERCIALMENTE DISPONÍVEIS USANDO ONTOLOGIAS
DE DOMÍNIO**

Monografia apresentada à Universidade Federal do Espírito Santo como requisito parcial para obtenção do título de Bacharel em Ciência da Computação, na área de concentração de Engenharia de Software, sob orientação do Professor Dr. Giancarlo Guizzardi.

VITÓRIA
2007

GUSTAVO FELHBERG

**INTEROPERABILIDADE SEMÂNTICA DE APLICAÇÕES
COMERCIALMENTE DISPONÍVEIS USANDO ONTOLOGIAS
DE DOMÍNIO**

Aprovada em 22 de Outubro de 2007

COMISSÃO EXAMINADORA

Prof. Giancarlo Guizzardi, Ph.D.

Orientador

Prof.^a Renata Silva Souza Guizzardi, Ph.D.

Prof. João Paulo Andrade Almeida, Ph.D.

Agradecimentos

Agradeço acima de tudo a Deus, por ter tornado tudo isso possível.

Aos meus pais, Marlene e Lourival, pelo apoio incondicional
em todos os momentos dessa caminhada até aqui. Sem
seus incentivos, não poderia estar aqui.

Aos meus grandes novos amigos (que já
nem são tão novos assim) achados nessa selva
chamada UFES, em especial Bruno e Aline, pelo apoio,
momentos de desespero nos finais de período, as longas
conversas discutindo os problemas da vida. Essa
amizade vai durar pra vida toda.

Aos mais novos amigos ainda, em especial Andréia e Thassy,
pelos ótimos momentos de gargalhadas nessa reta final.

Ao meu grande orientador Giancarlo, que com suas constantes
novas idéias, deixaram esse projeto muito interessante de ser realizado.

Ao pessoal da CVRD, em especial a João Hermínio e
Carlos Magno, pelos puxões de orelha, pelas longas conversas
e principalmente pelos ensinamentos que não vou levar apenas
para a minha vida profissional, mas também para
minha vida pessoal.

Por último e não menos importante, a Dan, que mesmo tendo
entrado a pouco tempo em minha vida, me deu forças nessa reta final,
me incentivando a cada minuto para conseguir entregar
essa monografia a tempo. E olha que quase não deu tempo!!!

RESUMO

Organizações de um modo geral estão cada vez mais preocupadas em oferecer produtos e serviços com maior qualidade em prazos cada vez menores. Para isso, sistemas de informação estão sendo usados para atender as mais diversas áreas dessas organizações. Contudo, essas aplicações são geralmente desenvolvidas por fabricantes diferentes, logo não estão preparadas para trocar informações entre outras aplicações, que atendem áreas diferentes, desenvolvidas por fabricantes diferentes. Dessa forma, é necessário que seja desenvolvida alguma abordagem que proveja comunicação entre as informações manipuladas por esses sistemas.

Este trabalho tem como objetivo apresentar uma abordagem de integração semântica entre ferramentas de software comercialmente disponíveis (*Commercial off-the-shelf - COTS*) de forma que suas informações manipuladas sejam compartilhadas, evitando assim, informações repetidas e inconsistentes. Essa abordagem está baseada em uma fundamentação semântica das ferramentas de software a partir do uso de ontologias de referência.

Palavras-chave: Sistemas de Informação – integração semântica - ontologias

LISTA DE FIGURAS

Figura 1 - Exemplo de Diagrama de Casos de Uso	17
Figura 2 - Tecnologias de Integração Semântica.....	20
Figura 3 - Morfologia do CD Player	31
Figura 4 - Morfologia do CD Player com exemplos de elementos morfológicos	32
Figura 5 - Alguns elementos morfológicos do CD Player.....	33
Figura 6 - Exemplos de navegação entre elementos morfológicos	34
Figura 7 - Parte do Mapa Morfológico do CD Player com sua interface principal.....	36
Figura 8 - Mapa Morfológico do CD Player	38
Figura 9 - Extração de conceitos a partir do mapa morfológico	40
Figura 10 - Exemplo de generalização	41
Figura 11 - Parte do modelo conceitual do CD Player com relações de agregação	42
Figura 12 - Exemplo de relacionamento de associação no modelo conceitual	43
Figura 13 - Modelo Conceitual do CD Player.....	44
Figura 14 - Diagrama de Casos de Uso com as fronteiras de escavação da ferramenta Microsoft Project	48
Figura 15 - Mapa Morfológico completo da ferramenta Microsoft Project 2003	56
Figura 16 – Mapa Morfológico do Microsoft Project - Parte 1 de 7	57
Figura 17 – Mapa Morfológico do Microsoft Project - Parte 2 de 7	58
Figura 18 – Mapa Morfológico do Microsoft Project - Parte 3 de 7	59
Figura 19 – Mapa Morfológico do Microsoft Project - Parte 4 de 7	60
Figura 20 – Mapa Morfológico do Microsoft Project - Parte 5 de 7	61
Figura 21 – Mapa Morfológico do Microsoft Project - Parte 6 de 7	62
Figura 22 – Mapa Morfológico do Microsoft Project - Parte 7 de 7	63
Figura 23 - Modelo Conceitual da ferramenta Microsoft Project 2003	66
Figura 24 - Ontologia de Processos de Software e suas sub-ontologias	68
Figura 25 - Sub-ontologia de Atividades.....	69
Figura 26 - Parte da sub-ontologia de Recurso.....	69
Figura 27 - Parte da sub-ontologia de Procedimento	70
Figura 28 - Parte da ontologia de Processos de Software.....	71
Figura 29 - Ontologia de Riscos de Software.....	75
Figura 30 - Mapeamento entre as ontologias de Riscos e Processos de Software com o Modelo Conceitual do Microsoft Project	79
Figura 31 – Ontologia de Gerência de Configuração de Software.....	85

Figura 32 – Modelo Conceitual da ferramenta de gerência de configuração CVS	89
Figura 33 – Mapeamento entre a Ontologia de Gerência de Configuração de Software e o Modelo Conceitual do CVS.....	90
Figura 34 – Mapeamento entre as Ontologias de Referência de Processo de Software e Gerência de Configuração de Software	93
Figura 35 – Mapeamento entre os Modelos Conceituais do Microsoft Project e CVS.....	96

LISTA DE TABELAS

Tabela 1 – Conceitos e Relacionamentos da ontologia de Processo de Software	74
Tabela 2 – Conceitos e Relacionamentos da ontologia de Riscos de Software	77
Tabela 3 – Mapeamento entre conceitos do Modelo Conceitual e Ontologia de Riscos de Software.....	80
Tabela 4 – Mapeamento entre conceitos do Modelo Conceitual e Ontologia de Processos de Software.....	81
Tabela 5 – Dicionário de conceitos e relacionamentos da ontologia de GCS	88
Tabela 6 – Mapeamento conceitual entre o modelo conceitual do CVS e da ontologia de GCS	92
Tabela 7 – Mapeamento entre conceitos das ontologias de Processo de Software e GCS.....	94
Tabela 8 – Mapeamento entre os modelos conceituais do Microsoft Project e o CVS.....	97

SUMÁRIO

Capítulo 1 - Introdução.....	10
1.1 Objetivos.....	10
1.2 Metodologia.....	11
1.3 Estrutura do Trabalho	11
Capítulo 2 - Pressupostos Teóricos	13
2.1 Ontologias.....	13
2.1.1 Ontologias em Engenharia de Software	14
2.2 Processos de Software	15
2.3 Casos de Uso	16
2.4 Integração de Ferramentas.....	17
2.4.1 Abordagens.....	18
Capítulo 3 - Abordagem Proposta	25
3.1 Extração de Modelos Conceituais	25
3.1.1 Bases de dados relacionais	26
3.1.2 Esquemas XML para OWL	27
3.1.3 Extração de Modelos Conceituais a partir da Interface Gráfica.....	29
3.1.3.1 Escavação de Ontologias	29
3.1.3.1.1 Criação do Mapa Morfológico	30
3.1.3.1.2 Construção do Modelo Conceitual	39
3.2 Abordagem	44
Capítulo 4 - Estudo de Caso	46
4.1 Gerência de Projetos.....	46
4.2 Casos de Uso	47
4.3 Escavação da ferramenta Microsoft Project	54
4.3.1 Mapa Morfológico	54
4.3.2 Modelo Conceitual	64
4.4 Ontologia de Processos.....	67
4.4.1 Sub-ontologia de Atividade	68
4.4.2 Sub-ontologia de Recurso.....	69
4.4.3 Sub-ontologia de Procedimento.....	70
4.4.4 Ontologia de Processos de Software	70
4.5 Ontologia de Riscos.....	74

4.6 Mapeamento entre um modelo conceitual e uma ontologia	77
4.6.1 Mapeamento entre a Ontologia de Processos de Software, Ontologia de Riscos e o Modelo Conceitual do Microsoft Project	78
Capítulo 5 - Exemplo de integração de ferramentas através de integração dos modelos conceituais	83
5.1 Ontologia de Gerência de Configuração de Software	84
5.2 Modelo Conceitual do CVS.....	88
5.3 Mapeamento entre Ontologia de GCS e Modelo Conceitual do CVS	89
5.4 Mapeamento entre Ontologias de Referência.....	92
5.5 Mapeamento entre Modelos Conceituais	94
Capítulo 6 - Considerações Finais	98
6.1 Perspectivas Futuras	99
Referências	100

Capítulo 1

Introdução

Empresas de desenvolvimento de software estão cada vez mais preocupadas em entregar produtos e serviços com um alto grau de qualidade e com prazos cada vez menores. Para isso, ferramentas de software passaram a ser usadas para apoiar o processo de desenvolvimento. Contudo, as ferramentas atualmente disponíveis no mercado normalmente apóiam somente partes específicas do processo, sendo necessário o uso de mais de uma ferramenta para abordá-lo como um todo.

Dessa forma, com vários softwares atuando em conjunto, é necessária a comunicação entre estes a fim de trocarem informações relativas ao processo. Assim, os dados relacionados a uma determinada etapa do processo que foram manipulados automaticamente por uma ferramenta podem ser usados por outra que atende uma etapa diferente. No entanto, visto que essas ferramentas normalmente são produzidas por fabricantes diferentes, essas não disponibilizam meios de comunicação entre si. Além dessa falha de comunicação, as ferramentas tratam os conceitos de forma diferente, com terminologias diferentes, mesmo se tratando de conceitos com o mesmo significado.

Para que possa ser estabelecida comunicação, é necessário que sejam identificados os conceitos semanticamente idênticos manipulados por diferentes softwares. Dessa forma a troca de informações pode ser feita mediante essas fronteiras conceituais.

1.1 Objetivos

Este trabalho possui três objetivos principais. O primeiro objetivo consiste em explicitar a semântica subjacente a modelos conceituais de ferramentas de software. Em outras palavras, este consiste em mapear um modelo conceitual de uma ferramenta em uma ontologia de referência para que essa possa servir como base semântica para o modelo que até então possuía apenas elementos sintáticos.

O segundo objetivo consiste em fazer uma avaliação sobre uma ferramenta de software a fim de saber se esta ferramenta pode ser usada de forma adequada no apoio a um processo de desenvolvimento. Este objetivo depende diretamente do primeiro, porque para

atingir este, o primeiro deve ser atingido. A explicitação semântica de um modelo conceitual é feito através do mapeamento entre o modelo e uma ontologia. A partir desse mapeamento, a ferramenta será avaliada se apta para apoiar um processo de desenvolvimento. O primeiro e segundo objetivo será apresentado a partir de um estudo de caso mostrando a extração do modelo conceitual de uma ferramenta e explicitando semanticamente o seu modelo.

O terceiro objetivo consiste em apresentar uma integração semântica entre ferramentas que apóiam um processo de desenvolvimento. Isso será feito a partir da integração entre os modelos conceituais das ferramentas apresentadas no exemplo de integração.

1.2 Metodologia

A Metodologia utilizada no desenvolvimento desse trabalho foi composta das seguintes atividades: revisão literária, definição do escopo do trabalho, extração do modelo conceitual usado no estudo de caso apresentado, estudo das ontologias e modelos conceituais envolvidos e apresentados nesse trabalho, desenvolvimento do mapeamento entre esses artefatos.

Inicialmente foi feita uma pesquisa bibliográfica abordando o assunto de ambientes de desenvolvimento de software, integração de ferramentas - apontando dificuldades encontradas e abordagens usadas, ontologias aplicadas à computação e métodos de extração de modelos conceituais.

Em seguida, foi definido o escopo do trabalho, já apresentado nesse capítulo introdutório. Depois, usando a abordagem encontrada em (HSI, 2005), foi feita a extração do modelo conceitual da ferramenta Microsoft Project. Para esse trabalho, serão usadas as ontologias de Processos de Software, definida em (FALBO, 1998) e (BERTOLLO, 2006), ontologia de Riscos de Software, definida em (FALBO, 2004) e a ontologia de Gerência de Configuração de Software, definida em (NUNES, 2005) e em (ARANTES, 2007). Essas ontologias serão brevemente apresentadas no decorrer desse trabalho.

A partir do estudo das ontologias e modelos conceituais é apresentado um exemplo de integração semântica entre duas ferramentas que apóiam um processo de desenvolvimento de software a partir do mapeamento de suas respectivas ontologias e modelos conceituais.

1.3 Estrutura do Trabalho

Este trabalho está organizado em seis capítulos. Além do presente capítulo, que constitui a introdução, há, ainda, mais cinco, descritos a seguir.

O Capítulo 2 – Pressupostos Teóricos – descreve algumas introduções teóricas necessárias para o entendimento dos conceitos usados no decorrer deste trabalho e faz uma discussão sobre o estado da arte da integração de ferramentas.

O Capítulo 3 – Abordagem Proposta – discute qual a abordagem usada nesse trabalho para apresentar a integração semântica entre as ferramentas. Apresenta também abordagens usadas atualmente para a extração de modelos conceituais de ferramentas, focando na abordagem usada no estudo de caso apresentado nesse trabalho.

O Capítulo 4 – Estudo de Caso – apresenta um estudo de caso que mostra como um modelo conceitual de uma ferramenta pode ser explicitado semanticamente através de seu mapeamento em uma ontologia de referência. Através deste mapeamento, é apresentado também como uma ferramenta pode ser avaliada como apta a apoiar um processo de desenvolvimento de forma adequada.

O Capítulo 5 – Exemplo de integração de ferramentas através de integração dos modelos conceituais - apresenta um exemplo de integração semântica entre duas ferramentas que apoiam o processo de desenvolvimento de software a partir do mapeamento de seus modelos conceituais através do mapeamento das ontologias de referência para os quais estas foram mapeadas.

Por fim, o Capítulo 6 – Considerações Finais – apresenta as conclusões obtidas com o desenvolvimento deste trabalho e são apresentadas algumas possibilidades de trabalhos futuros.

Capítulo 2

Pressupostos Teóricos

Neste capítulo será feita uma introdução teórica dos principais conceitos discutidos neste trabalho. Essa conceituação é necessária para garantir o entendimento das técnicas, abordagens e exemplos discutidos a seguir.

Este capítulo está dividido da seguinte forma: A seção 2.1 faz uma introdução ao conceito de ontologias e discute um pouco sua função neste trabalho. A seção 2.2 descreve brevemente o que são processos de software e qual a importância em seu uso em organizações atualmente. A seção 2.3 faz uma breve introdução sobre Casos de Uso, discorrendo sobre suas aplicações no decorrer de um processo de desenvolvimento de software. Por fim, na seção 2.4 é discutido o estado da arte de integração de ferramentas de software.

2.1 Ontologias

Nas ultimas décadas há um crescente interesse no campo de ontologias em ciências da computação e informação. Nos últimos anos, esse interesse tem se expandido consideravelmente no contexto de pesquisa da Web Semântica e MDA, devido a ontologias fazerem parte direta nessas iniciativas.

Etimologicamente, *ont-* vem do verbo grego *einai* (ser) e a palavra em latim Ontologia (*ont- + logia*) pode ser traduzida como o estudo da existência (GUIZZARDI, 2007).

Os primeiros estudos relacionados à representação de conhecimento que se assemelham aos que estudamos hoje datam dos tempos da Grécia antiga, realizados principalmente por Aristóteles (384-322 A.C.), nos campos da lógica, ciências naturais e filosofia metafísica (RUSSEL, 1995).

O termo “ontologia” em ciências da computação e informação apareceu pela primeira vez em 1967, em um trabalho sobre fundamentos de modelagem de dados por S. H. Mealy, em uma passagem onde ele distingue três reinos no campo de processamento de dados: (i) “o mundo real”, (ii) “idéias sobre a existência de algo na cabeça dos homens” e (iii) “símbolos no papel ou outra mídia de armazenamento”. Mealy conclui essa passagem

argumentando sobre a existência das coisas no mundo independentemente de suas (possíveis) múltiplas representações e alega que “Isso é uma questão da ontologia ou uma questão do que existe.” (MEALY, 1967). De uma maneira independente, ainda outro campo da ciência da computação, o de Inteligência Artificial (IA) começou a usar o que ficou conhecido com *ontologias de domínio*. Desde que esse termo foi usado pela primeira vez em IA por (HAYES, 1978) e desde o desenvolvimento de sua ontologia de líquidos, um grande número de ontologias de domínio foram desenvolvidas nas mais diversas áreas. Nos últimos anos, uma explosão de trabalhos relacionados a ontologias tem acontecido em ciência da computação, motivado principalmente pelo crescente interesse em Web Semântica (GUIZZARDI, 2007).

Ontologias são bases de conhecimento de algum domínio específico de interesse. Estas servem, dentre inúmeras outras finalidades, para manter um entendimento único das informações de algum domínio em questão. Ontologias consistem em conceitos e relações – o vocabulário – e suas definições, propriedades e restrições descritas na forma de axiomas (FALBO, 1998). Por meio de ontologias, é possível conseguir uma uniformidade de vocabulário, de forma a evitar ambigüidades e inconsistências. Mais precisamente, não é o vocabulário que especifica uma ontologia, mas as conceituações que os termos do vocabulário pretendem capturar (CHANDRASEKARAN *et al.*, 1999).

Manter um vocabulário padrão é essencial para as atividades relacionadas ao processo de desenvolvimento de software. Essa padronização harmoniza o entendimento de todos os recursos envolvidos, facilitando a troca de informações entre pessoas diferentes, com visões diferentes do processo.

Atualmente existem muitas linguagens usadas para representar ontologias. Dentre as principais, podemos citar: Cálculo dos Predicados, KIF, Ontolingua, UML, EER, LINGO, ORM, CML, DAML+OIL, F-Logic e OWL.

2.1.1 Ontologias em Engenharia de Software

Dentre os mais diversos campos da Engenharia de Software, o mais relacionado com Ontologias é o de Engenharia de Domínio. O propósito da Engenharia de Domínio é identificar, modelar, construir, catalogar e disseminar um conjunto de artefatos de software que podem ser aplicados ao software existente e futuro em um domínio de aplicação particular. A meta principal é estabelecer mecanismos que permitem engenheiros de software a compartilhar esses componentes – para reusa-los – durante o trabalho em sistemas novos ou já existentes (PRESSMAN, 2000).

Dentre as atividades da Engenharia de Domínio, as principais são: análise de domínio, especificação de infra-estrutura e implementação de infra-estrutura. O termo Análise de Domínio apareceu pela primeira vez na literatura em (NEIGHBORS, 1981) com a seguinte definição: “Análise de Domínio é uma tentativa de identificar objetos, operações e relações que especialistas de domínio entendem como importante em um dado domínio”. O principal produto dessa atividade é o Modelo de Domínio, que descreve os objetos e seus relacionamentos em um determinado domínio de discussão. Segundo (ARANGO, 1989), esse modelo deve servir como: (1) uma fonte unificada de referência quando aparecem ambigüidades na análise dos problemas ou durante a implementação dos componentes reusáveis; (2) um repositório de conhecimento compartilhado para ensino e comunicação; e (3) uma especificação para o desenvolvedor de componentes reusáveis.

Como se pode notar, ontologias estão intimamente relacionadas com modelos de domínio. Ambas provêm uma conceituação a cerca do domínio em questão, buscando oferecer um entendimento uniforme dos conceitos e relacionamentos envolvidos. Uma ontologia pode fornecer um entendimento comum entre desenvolvedores e pode ser usada como um modelo de domínio (FALBO *et. al.*, 2002). Em (GUIZZARDI, 2000), o autor discute em detalhes as correspondências entre os processos de Engenharia de Domínio e Engenharia de Ontologias.

2.2 Processos de Software

Cada vez mais as empresas de software vislumbram a necessidade de definir um processo de desenvolvimento para tentar garantir a qualidade de seus produtos. Apesar de não ser garantia uma boa qualidade do produto final, um processo de software bem definido pode aumentar a probabilidade de um produto ser gerado com qualidade.

Contudo, a criação de um processo de software não é uma simples atividade. Existem muitas referências que indicam as melhores práticas a serem seguidas, mas cada organização deve definir seus processos levando em consideração não só essas informações, mas também suas próprias características (BERTOLLO *et al.*, 2006).

Um processo de software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software (FUGGETTA, 2000). Um processo de software bem definido deve indicar as atividades a serem executados, os recursos (humanos, de hardware e de software) requeridos, os artefatos consumidos e produzidos e os

procedimentos a serem adotados (métodos, técnicas, modelos de documentos, entre outros) (FALBO, 1998) (GRUHN, 2002).

Processos de desenvolvimento de software podem ser definidos a partir de uma ontologia (ou conjunto de ontologias) referentes ao domínio de Engenharia de Software. Uma ontologia de referência desse nível busca compreender todo o conhecimento necessário para a elaboração de um processo de software.

A partir de um processo de software definido, ferramentas específicas de domínio podem ser identificadas para apoiar partes específicas do processo. Dessa forma, podem ser buscadas no mercado várias ferramentas que abordam o domínio em questão. Contudo, uma ferramenta que está proposta a atender determinado domínio não garante que o está abordando de forma correta. Para isso, esta deve manipular seus conceitos de forma consistente com os abordados pela ontologia de referência que originou o processo de desenvolvimento. Caso seja identificado que essa ferramenta lida com conceitos que não estão de acordo com a ontologia de referência, essa não está apta a apoiar de maneira ideal esse processo de desenvolvimento.

Não será considerada neste trabalho a definição de um processo de software baseado em ontologias de referência. Neste trabalho será apresentado como um estudo de caso uma ferramenta que é amplamente usada no apoio de processos de software em diversas organizações. Será apresentada uma verificação se essa ferramenta manipula os conceitos de forma consistente com os propostos por duas ontologias de domínio ao qual essa ferramenta está propensa a abordar. Dessa forma, poderá ser avaliado se essa ferramenta pode ser usada no apoio a um processo de desenvolvimento de software.

2.3 Casos de Uso

Segundo (PRESSMAN, 2001), casos de uso são conjuntos de cenários que identificam um processo que indica como o sistema a ser construído será usado. Tipicamente, casos de uso são usados para capturar os requisitos do sistema, isto é, o que o sistema está proposto a fazer (OMG, 2005).

Casos de uso normalmente são apresentados em diagramas indicando que um ou mais atores podem executar determinadas ações relativas às funcionalidades do sistema. Os atores são os usuários do sistema, sejam eles humanos ou outros sistemas, ou seja, qualquer entidade que pode interagir com o sistema desenvolvido.

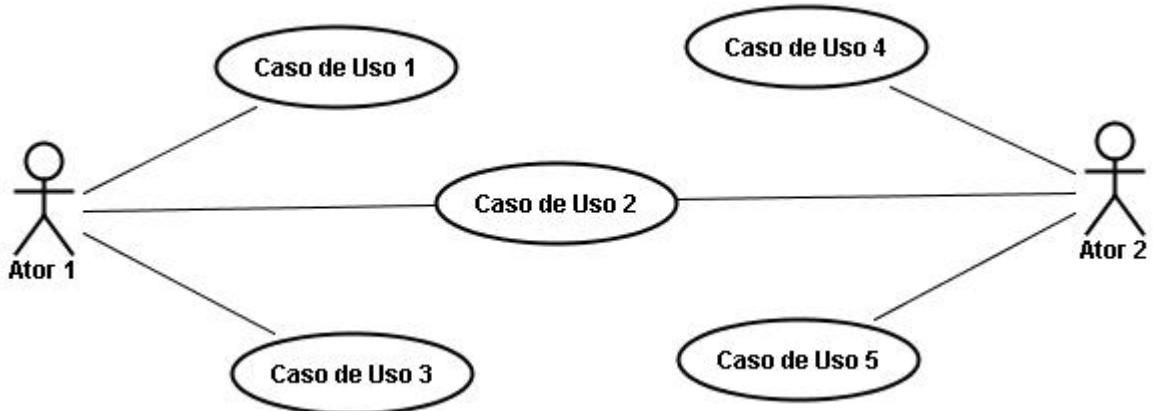


Figura 1 - Exemplo de Diagrama de Casos de Uso

A Figura 1 apresenta um modelo de um diagrama de classes com dois atores e cinco casos de uso. Os bonecos representam os atores e os círculos representam os casos de uso referentes ao sistema. As setas entre os atores e os casos de uso indicam quais casos de uso um determinado ator pode executar. Como pode ser visto, um caso de uso pode ser executado por mais de um ator. Isso significa que a funcionalidade do sistema referente a esse caso de uso pode ser acessada por mais de um usuário.

Diagramas de casos de uso são artefatos que normalmente são criados durante a fase de análise em um processo de desenvolvimento. Esses ajudam a verificar se todas as funcionalidades inicialmente desejadas estão sendo abordadas durante o desenvolvimento.

2.4 Integração de Ferramentas

A cada dia que passa mais empresas estão adotando Sistemas de Informação (SI) para apoiar seus negócios. Isso é necessário, pois as empresas precisam recuperar suas informações de forma rápida para se manterem competitivas no mercado. Empresas de software já têm como fator obrigatório o uso de SI, visto o dinamismo que as informações são trocadas.

Os envolvidos em um processo de desenvolvimento de software precisam de acesso a grande parte ou todas as informações relacionadas às etapas do ciclo de vida do projeto. O grande problema é que normalmente essas informações estão distribuídas em sistemas de informação heterogêneos, levando as informações a ficarem também heterogêneas e distribuídas (WACHE, 2001). Em empresas de desenvolvimento de software são usados vários SI, divididos em áreas específicas do processo. Dessa forma, cada sistema ou conjunto de sistemas detêm as principais informações relativas a uma parte do processo. Com isso, as

pessoas interessadas no desenvolvimento devem recuperar as informações em diferentes sistemas, o que demanda um tempo maior e com possibilidade de informações conflitantes.

Visando acabar com esse isolamento de informações, vê-se cada vez mais necessária a integração de ferramentas. Dessa forma, os sistemas envolvidos no processo de desenvolvimento de software podem se comunicar diretamente, compartilhando informações relativas ao processo, facilitando a recuperação de conhecimento.

Segundo (BROWN et al., 1992), a integração de ferramentas deve prover:

- Sinergia entre as ferramentas para alcançar produtividade;
- Visibilidade do processo de desenvolvimento para manter o controle;
- Comunicação não ambígua entre as ferramentas para alcançar qualidade; e
- Consistência entre interfaces para promover eficiência e para prover compatibilidade.

Apesar de ter um impacto significativo no negócio da empresa, ambientes integrados ainda têm seu potencial limitado por dificuldades envolvidas ao processo de integração (CHEN, 1992).

2.4.1 Abordagens

A seguir, várias abordagens que constituem o estado da arte de integração de ferramentas são apresentadas.

2.4.1.1 Integração de ferramentas baseada em modelos (*Model-based tool integration*)

Esse tipo de integração se baseia no paradigma de desenvolvimento de software baseado em modelos (*Model-Based Software Development*), propagado pela *Object Management Group* (OMG) com sua Arquitetura Orientada a Modelos (do inglês, *Model-Driven Architecture - MDA*). Esse padrão adota modelos ao invés de código puro como o centro do processo de desenvolvimento, sendo usado para diferentes tipos de atividades de desenvolvimento. Essa abordagem permite que as ferramentas de modelagem sejam integradas com base na intercessão dos metamodelos das linguagens de modelagem suportadas por essas ferramentas (metamodelos das ferramentas), abrindo caminho para outra geração de integração de ferramentas baseada em (meta-) modelos. Dessa forma, pode ser explorado um mais alto nível de abstração, uma maior riqueza de visualização e o poder de

expressividade muito maior se comparado com um código de alguma linguagem de programação. Para isso, o *MDA* provê um conjunto de padrões inter-relacionados, englobando uma linguagem para definição de metamodelos (*Meta Object Facility – MOF*) e linguagens baseadas em MOF para especificação de restrições (*Object Constraint Language – OCL*), troca de metadados (*XML Metadata Interchange – XMI*) e transformação de modelos (*Query/View/Transformation - QVT*) (KAPSAMMER, 2006).

As linguagens de transformação de modelos têm recebido bastante atenção no contexto de integração de ferramentas baseada em modelos. Sua principal função é transformar Modelos Independentes de Plataforma (*PIM*, do inglês, *Platform Independent Model*) para Modelos Específicos de Plataforma (*PSM*, do inglês, *Platform Specific Model*).

Existem *frameworks* e plataformas que permitem integração de ferramentas baseadas as linguagens de transformação de modelos baseadas em QVT. Como exemplo, WOTIF (*Web-based open tool integration framework*) (ISIS, 2007) que usa um mecanismo de transformação de grafos que realiza diferentes cenários de integração de ferramentas (por exemplo, integração direta de ferramentas e integração via um metamodelo comum), mas requer que toda ferramenta a ser integrada suporte determinada API para instalação de plugins (KAPSAMMER, 2006). Outros exemplos são GeneralStore (REICHMANN, 2004) e o MDDI (Model-driven Development Integration Project for Eclipse) (ECLIPSE, 2007).

2.4.1.1 Integração Semântica e Ontologias

Outro fator determinante na integração de ferramentas baseadas em modelos é a interoperabilidade semântica entre os modelos. Um problema encontrado é que modelos por si só não oferecem semântica aos conceitos abordados, o que dificulta o progresso da integração. Vários estudos já foram e estão sendo realizados nesse aspecto, com atual destaque no uso de ontologias. Em comparação com outras técnicas, a integração baseada em ontologias conta com um alto poder de expressividade de algumas de suas linguagens e com técnicas apropriadas de raciocínio (KAPSAMMER, 2006).

Existem algumas técnicas de integração semântica que são diretamente aplicadas em integração de ferramentas baseadas em modelos. Dentre elas, podemos citar Elevação de Metamodelos para Ontologias (*Lifting Metamodels to Ontologies*), Integração de Ontologias de Ferramentas (*Integrating Tool Ontologies*), Identificação de Mapeamentos (*Mapping*

Discovery) e Representação e Raciocínio em Mapeamentos (*Representation of and Reasoning with Mappings*). Estas técnicas serão discutidas a seguir na Figura 2 (KAPSAMMER, 2006).

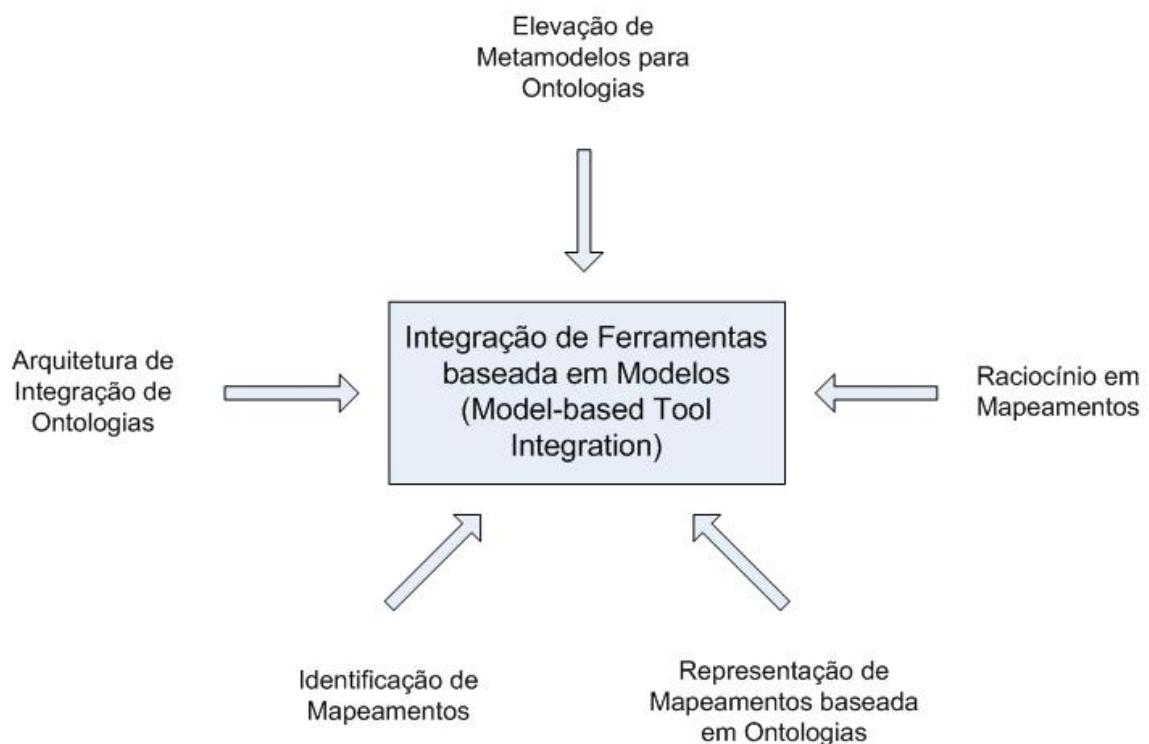


Figura 2 - Tecnologias de Integração Semântica

2.4.1.1.1 Elevação de Metamodelos para Ontologias

Essa técnica consiste em derivar as ontologias a partir dos metamodelos das ferramentas. Dentre os principais exemplos práticos, são destacados (KAPSAMMER, 2006):

- **OntoLIFT:** É uma forma de criação de ontologias de forma semi-automática a partir de esquemas de bancos de dados usando padrões sintáticos da mesma forma que é aplicado no mapeamento de esquemas de bases de dados para modelos de Entidade e Relacionamento (ER) (KAPSAMMER, 2006).
- **Ferdinand et al.:** Abordagem desenvolvida por (FERDINAND, et al., 2004), que propõe um mecanismo automático para representar os conceitos do domínio da ferramenta em linguagem OWL (Web Ontology Language) a partir do esquema XML provido pela ferramenta.

2.4.1.1.2. Integração de Ontologias de Ferramentas

A integração de ferramentas baseada em modelos realiza integração de metamodelos a partir da semântica extraída das ontologias referentes aos domínios das ferramentas. Dessa forma, essas ontologias também precisam ser integradas. Para isso, devem ser confrontadas diferentes formas de heterogeneidade, estabelecer certa arquitetura de integração de ontologias e prover mecanismos apropriados de descoberta de mapeamento entre ontologias, representação e raciocínio (NOY, 2004).

De acordo com (KAPSAMMER, 2006), no nível arquitetural de integração de ontologias, podem ser distinguidas três alternativas:

1. Mapeamento direto entre ontologias;
2. Mapeamento indireto via uma ontologia comum e distribuída, chamada de ontologia de alto nível;
3. Mapeamento baseado em uma biblioteca de ontologias já mapeadas.

Para a integração baseada em modelos, é usada uma abordagem híbrida, envolvendo todas as três arquiteturas.

2.4.1.1.3. Identificação de Mapeamentos entre Ontologias

Conceitos comuns entre as ontologias devem ser mapeados a fim de uma integração consistente. Para isso, técnicas de mapeamento de conceitos devem achar correspondências entre ontologias. Isso pode ser feito de forma totalmente manual ou usando técnicas com base heurística ou de aprendizado de máquina que usa várias características de ontologias, como sua estrutura, definições de conceitos e instâncias de classes para achar mapeamentos (NOY, 2004).

2.4.1.1.4. Representação e Raciocínio em Mapeamentos

Uma vez definidos os mapeamentos entre as ontologias, faz-se necessário suas representações de forma a serem usados para aplicar algum tipo de raciocínio lógico, a fim de tirar proveito dos relacionamentos entre as ontologias.

Segundo (NOY, 2004), as principais formas de representação de mapeamentos entre ontologias são: definir axiomas de ligação em lógica de primeira ordem para representar transformações, classes relacionadas e propriedades de duas ontologias; e usar *views* para descrever mapeamentos de uma ontologia global (ontologia de alto nível) para ontologias locais; e representação de mapeamentos como instâncias de ontologias de mapeamentos. As ontologias de mapeamento provêm diferentes formas de ligar conceitos da ontologia de origem com a de destino, regras de transformação para especificar como os valores devem ser modificados e condições e efeitos dessas regras (KAPPEL, 2005).

Uma vez feitos estes mapeamentos, informações podem ser inferidas a partir destas ontologias relacionadas através de um processo de raciocínio automático.

2.4.1.2 ModelCVS – Uma estrutura semântica para integração de ferramentas baseada em modelos

Essa abordagem provê uma infra-estrutura semântica para integração de ferramentas baseadas em modelos. O núcleo dessa infra-estrutura é um sistema baseado em uma aplicação de gerência de configuração como o CVS (*Concurrent Versioning System*), provendo assim uma arquitetura aberta, porém consistente para integração (KAPPEL, et al., 2005). Essa é uma arquitetura escalável, visando diminuir o esforço repetitivo para integração de uma nova ferramenta, permitindo assim, aproveitar ao máximo o reuso de conhecimento relacionado à integração.

O ModelCVS permite transformação transparente de modelos entre linguagens de modelagem de ferramentas diferentes baseadas em MOF, assim como permite um mecanismo de gerência de configuração para os registrar as várias atualizações que os modelos podem sofrer. Isso provê um desenvolvimento concorrente permitindo armazenar e versionar artefatos de software que clientes podem acessar via mecanismos de *check-in* e *check-out*, similares com servidores de controle de versão. ModelCVS foca diferentes cenários de integração, como tradução de metamodelos e modularização. Tecnologias semânticas em termos de ontologias são usadas em conjunto com uma base de conhecimento para armazenar informações relevantes à integração de ferramentas, passíveis ao entendimento de máquina, permitindo assim a diminuição do esforço repetido e automatizar parcialmente o processo de integração (KAPSAMMER, 2005).

A integração é tratada tanto no nível sintático quanto no nível semântico. O nível sintático lida com os metamodelos que definem as estruturas e os tipos de dados dos modelos,

ao passo que o nível semântico usa ontologias que descrevem a semântica dos conceitos de modelagem. O nível semântico de integração e o reuso de conhecimento de integração serão especialmente apoiados através de um componente dedicado a infra-estrutura semântica que facilita a elevação de metamodelos ao nível semântico, encontrando mapeamentos entre os metamodelos a serem integrados e encontrando conflitos semânticos na junção de modelos.

Segundo (KAPPEL, et al., 2005), esse projeto tem como metas principais de pesquisa (1) criar uma nova linguagem de integração escalável de ferramentas baseada em modelos, (2) criar tecnologias inovadoras para integração de metamodelos baseados em ontologias e (3) criar uma base de conhecimento aberta para integração de ferramentas.

2.4.1.2.1 Nova linguagem de integração escalável de ferramentas baseada em modelos

Esse tipo de integração inclui a ligação entre metamodelos. Essas ligações definem as transformações de modelos facilitando a tradução transparente entre eles. A grande dificuldade na ligação entre modelos é devida a heterogeneidade entre os conceitos envolvidos nos metamodelos. Para isso, o projeto ModelCVS propõe a criação de uma linguagem especificamente desenvolvida para ligação de metamodelos. Serão identificados padrões arquiteturais de integração de modelos que asseguram abertura, escalabilidade e evolução de uma solução de integração de ferramentas (KAPPEL, 2005).

2.4.1.2.2 Tecnologias inovadoras para integração de metamodelos baseados em ontologias

Uma forma de melhorar a qualidade no apoio a automatização de integração de metamodelos é se basear em ontologias como referência. Muitos trabalhos vêm sendo desenvolvidos na área de integração de ontologias. A diferença essencial entre metamodelos e ontologias é que metamodelos definem os conceitos de uma linguagem de modelagem em termos de sua sintaxe, enquanto que as ontologias focam na semântica de seus conceitos, não levando somente em consideração a forma sintática.

Uma técnica usada é a de elevação de metamodelos, que visa permitir transição entre o nível sintático de um metamodelo para o nível semântico de uma ontologia. Esse processo tem como objetivo efetuar um mapeamento entre os níveis de metamodelo e de ontologia de forma quer ambos os níveis possam ser usados sinergicamente (KAPPEL, 2005).

2.4.1.2.3 Base de conhecimento aberta para integração de ferramentas

Um requisito fundamental para essa abordagem é prover reusabilidade para o processo de definir semântica para a ontologia da ferramenta. Esse processo de reusabilidade tem como objetivo criar uma base de conhecimento de integração de ferramentas para prover conceitos a serem reutilizados para a definição das ontologias das ferramentas.

Segundo (KAPPEL, 2005), os passos principais da construção dessa base são:

1. Identificação de conceitos genéricos e reusáveis e desenvolvimento de uma estrutura para organizar o conteúdo da base de conhecimento para oferecer reusabilidade;
2. Preencher a base de integração com exemplos de inferência para aumentar a capacidade de reuso e mapeamento de conceitos. Para esse projeto, esses exemplos seriam os resultados obtidos pelos cenários de integração propostos nessa pesquisa.
3. Definir um conhecimento sólido de conflitos de mesclagem semântica, ou melhor, problemas encontrados durante a junção de dois modelos referentes à mesma representação, porém com algumas diferenças entre si, para melhorar a habilidade de gerência de configuração de modelos, identificação automática e consequentemente, resolução desses conflitos;
4. Estabelecimento de uma plataforma pública podendo ser acessada pela internet, permitindo colaboração externa para a base de conhecimento para aumentar a reusabilidade.

Capítulo 3

Abordagem Proposta

Nesse capítulo será apresentada a abordagem proposta para atingir os objetivos deste trabalho, definidos no capítulo introdutório.

Na seção 3.1 será feita uma introdução sobre a extração de modelos conceituais a partir de alguns componentes de uma aplicação. Em suas subseções serão apresentadas brevemente duas técnicas de extração de modelos, a saber, uma a partir de bancos de dados relacionais e outra a partir de arquivos XML. Em seguida, na subseção 3.1.3 será apresentada com mais detalhes a técnica de extração de modelos conceituais a partir da interface gráfica da aplicação, que será a técnica usada no caso de uso e exemplo de integração de ferramentas presentes nos capítulos 4 e 5, respectivamente.

Na seção 3.2 será apresentada a abordagem propriamente dita adotada para atingir os objetivos deste trabalho.

3.1 Extração de Modelos Conceituais

Com o objetivo de representar o conhecimento de uma forma mais estruturada, de mais fácil leitura, permitindo o compartilhamento de informações relativas ao domínio avaliado, modelos conceituais passaram a ser usados. Modelos Conceituais são representações que descrevem as principais características de um domínio de conhecimento (WEBER, 1997). Esses modelos permitem que o conhecimento referente a determinado domínio possa ser apresentado de forma clara e concisa. Existem técnicas que apóiam a criação de modelos conceituais a partir de domínios bem definidos de aplicação. Essas técnicas se baseiam na extração de informações a partir de componentes de aplicações já desenvolvidas e disponíveis comercialmente.

Um erro cometido por alguns autores é afirmar que os artefatos extraídos a partir de algum tipo de componente do sistema que representam informações, sejam bancos de dados relacionais, sejam arquivos de troca de informação (como XML, por exemplo) ou pela própria interface gráfica sejam classificados como ontologias. Ontologias possuem um conceito muito mais amplo, tendo um poder de expressividade muito maior do que aqueles modelos extraídos a partir das abordagens encontradas na literatura. Todas as abordagens pesquisadas nessa

seção tratam os resultados finais da extração como ontologias, mas na realidade deveriam ser referenciados como modelos conceituais. Ontologias podem ser tratadas como modelos conceituais, porém o inverso não é válido. Os modelos obtidos são extraídos a partir das ferramentas, restringindo muito o domínio, podendo ser aplicado, em muitos casos, apenas no domínio específico da ferramenta, não podendo ser reutilizado para outras aplicações. Ontologias têm o propósito de representar os conceitos e relacionamentos que são de consenso em uma comunidade em um determinado domínio e existem casos onde uma ferramenta não expressa os conceitos desse senso comum corretamente. Dessa forma, o modelo fica preso aos conceitos propostos pela ferramenta, podendo fugir aos princípios ontológicos do domínio. A partir disso, para as abordagens descritas nessa seção, os artefatos produzidos serão tratados como modelos conceituais ao invés de ontologias.

Uma dificuldade encontrada na aplicação das técnicas que serão apresentadas é que a pessoa que usá-las deve, além de conhecer as técnicas, deve possuir um bom conhecimento sobre o domínio referente ao modelo conceitual a ser extraído. Isso porque essas técnicas propõem a extração de semântica a partir de artefatos que só capturam aspectos sintáticos das informações. Sem um conhecimento do domínio, não é possível saber com precisão qual a semântica relacionada com os itens sintáticos presentes nos componentes da aplicação. Dessa forma, ao analisar os conceitos extraídos, o responsável pela criação do modelo conceitual deve usar o seu conhecimento do domínio para identificar qual o significado e como esse conceito pode ser aplicado no modelo.

Nas sessões seguintes são abordadas algumas formas de extração de modelos conceituais usadas atualmente. A seção 3.3.1 trata a extração de modelos conceituais a partir de esquemas de bancos de dados relacionais. A seção 3.3.2 foca na criação de códigos na linguagem de ontologias da web (OWL) a partir de esquemas XML das ferramentas. Por fim, a seção 3.3.3 abre a discussão de extração de modelos conceituais a partir da interface gráfica da ferramenta, apresentando a técnica usada para a extração de modelos usada nesse trabalho.

3.1.1 Bases de dados relacionais

Bancos de dados relacionais são partes cada vez mais cruciais em sistemas de informação usados atualmente. A grande maioria dos sistemas que manipulam uma quantidade considerável de informação utiliza sistemas de bancos de dados relacionais como mecanismo de armazenamento.

Dados são armazenados em esquemas lógicos. Esses esquemas provêm algumas descrições sobre o universo de discurso aos quais os sistemas de informação operam. Dessa

forma, pode-se pensar na utilidade de extrair conceitos sobre um universo de discurso a partir dessas descrições. Em outras palavras, a princípio, modelos relacionais podem ser usados como ponto de partida para se levantar conceitos que comporão um modelo conceitual de um dado domínio.

No entanto, esse processo não provê a extração de dados muito específicos, não sendo possível a obtenção de um modelo conceitual muito complexo, visto que esses esquemas relacionais trazem muitas informações conceitualmente irrelevantes para o universo de discurso. Outro ponto a ser considerado é que os modelos de dados relacionais não são suficientes para descrições mais completas. Essa falha nos mecanismos de construção leva ao que é chamada de sobre-especificação, que é a introdução de entidades extras para capturar alguns fatos (VOLZ et al., 2002) que não são diretamente mapeados a partir dos esquemas relacionais.

Segundo (VOLZ et al., 2002), com o projeto OntoLIFT, o processo de obtenção da semântica de base de dados relacional se baseia na tradução das relações e atributos para conceitos e regras de um modelo conceitual. Essa tradução é realizada via uma série de regras de tradução descritas em (VOLZ et al., 2002).

A tradução cria todas as possíveis entidades ontológicas, detecta sobre-especificações, ou seja, relações auxiliares no esquema original e caso necessário, remove informações redundantes. A tradução é composta de duas fases. A primeira fase os conceitos são estabelecidos. Na segunda fase, as regras dos conceitos são criadas a partir da base de dados (VOLZ et al., 2002).

3.1.2 Esquemas XML para OWL

XML (*EXtensible Markup Language*) é uma meta linguagem que tem como objetivo criar linguagens de marcação (VOLZ et al., 2002). Originalmente ela foi desenvolvida para adequar diferenças entre publicações eletrônicas de grande escala. Hoje em dia é mais usada para a troca de uma grande variedade de dados na internet. Apesar disso, mesmo com essa nova forma de aplicação, as linguagens de esquema XML ainda contêm muitas características que são irrelevantes se tratando de troca de dados e estão concentrados em aspectos estruturais de documento ao invés de aspectos estruturais de dados. Segundo (W3C, 2004), XML provê uma superfície sintática para documentos estruturados, mas não impõe nenhuma restrição semântica no significado desses documentos. No entanto, visto sua forma estruturada de apresentação de informações, modelos conceituais podem ser construídos a partir de documentos com esquema XML. Para isso, o responsável pela criação do modelo conceitual

deve usar seu conhecimento relativo ao domínio para modelar de forma consistente essa conceitualização. (FERDINAND *et al.*, 2004) foca a extração de informações semânticas de esquemas de documentos e propõe um mecanismo para erguer um modelo conceitual em formato OWL a partir de um arquivo com esquema XML.

A abordagem é dividida em duas partes: primeiro documentos XML são traduzidos para grafos RDF e em seguida Esquemas XML são erguidos (*lifted*), originando os metamodelos em OWL.

A primeira parte se concentra no mapeamento de XML para RDF. XML é uma linguagem que define uma sintaxe genérica para armazenar e compartilhar informações de documentos se baseando em uma estrutura de árvores. Embora RDF tenha uma sintaxe baseada em XML, ambas servem a diferentes propósitos e tem sido desenvolvidas separadamente, o que as levam a diferentes fundamentos de modelagem (FERDINAND et al, 2004). Uma das principais diferenças é que RDF faz distinção entre recursos e propriedades (como exemplo carro e carro azul, respectivamente), enquanto que XML não o faz, tratando todos os conceitos como elementos.

Para suprir a falha entre as duas formas de representação de dados, foi proposto em (FERDINAND et al, 2004) um processo para erguer modelos de dados em RDF a partir de documentos XML. Esse mapeamento não necessita que qualquer alteração seja feita na estrutura dos documentos XML ou RDF, mantendo assim, compatibilidade com os documentos já existentes. Apesar das diferenças estruturais, essas não chegam a ser um obstáculo, visto que árvores são especializações de grafos.

Essa abordagem sugere que seja feita uma varredura em um arquivo XML analisando cada um de seus elementos e que para cada um deles, seja definido um recurso ou propriedade derivado em um arquivo RDF. Para isso, assim como nas outras abordagens, é necessário que o responsável por esta extração possua conhecimento suficiente para poder avaliar como um recurso XML possa ser derivado em um recurso ou propriedade em RDF.

A segunda parte da abordagem consiste em fazer o mapeamento entre um esquema XML (*XML Schema*) em um modelo conceitual em linguagem OWL. Esquemas XML e OWL tratam problemas diferentes: esquemas XML provêm meios para expressar e restringir a sintaxe e estrutura de documentos XML. Por outro lado, OWL se preocupa em modelar os relacionamentos semânticos de um dado domínio. No entanto, existe uma relação fundamental entre as duas linguagens, pois as duas se baseiam no paradigma orientado a objetos. Esquemas XML possuem a noção de hierarquia de classes e generalização e OWL é baseado na noção de Frames. Embora ambas contemplam níveis diferentes de abstração, as

linguagens têm como objetivo a obtenção de um vocabulário e estruturas comuns para apoiar a troca de informação (FERDINAND et al, 2004).

O mapeamento entre esquemas XML e OWL segue a mesma base do procedimento descrito acima. A diferença é que o mapeamento em esquemas XML é mais complexo, devido a um número maior de componentes a serem verificados.

3.1.3 Extração de Modelos Conceituais a partir da Interface Gráfica

Modelos Conceituais são representações usadas para descrever as principais informações relativas a um universo de conhecimento. Dessa forma, o uso desses modelos facilita o entendimento mais geral de um determinado domínio estudado. Infelizmente, um número significativo de sistemas desenvolvidos até hoje não possui algum tipo de documentação que apresenta esses modelos. Contudo, esse tipo de informação pode ser extraído dos artefatos mais comuns produzidos em um processo de desenvolvimento de software, a saber, Documento de Especificação de Requisitos, Documento de Análise, códigos fonte, etc., pois esses documentos concentram as principais informações relativas aos sistemas. No entanto, a maioria das empresas desenvolvedoras de software não disponibiliza esse tipo de documentação, dificultando a criação de um modelo conceitual.

Essa seção aborda a extração de modelos conceituais de sistemas que não oferecem modelos conceituais explicitamente representados durante seu processo de desenvolvimento.

3.1.3.1 Escavação de Ontologias

Nessa subseção será apresentada a abordagem proposta por (HSI, 2005) para construção de Modelos Conceituais a partir da interface gráfica de uma ferramenta. Essa abordagem, nomeada de Escavação de Ontologias (do inglês *Ontological Excavation*), consiste em fazer uma engenharia reversa de caixa preta para identificar os conceitos do domínio e relacionamentos a partir das características da aplicação (HSI, 2005).

Essa estratégia é usada quando o usuário não tem acesso a nenhuma documentação, parte do código fonte ou outro artefato que dê indícios de como o sistema foi desenvolvido. Dessa forma, os conceitos extraídos devem ser obtidos direta e exclusivamente da interface com o usuário. A metáfora “caixa preta” é feita justamente por essa falta de recursos que provêm informações relativas ao sistema escavado. De forma inversa, a metáfora “caixa branca” seria aplicada caso os conceitos para montar o modelo conceitual fossem extraídos a

partir de documentações do sistema, de um esquema de banco de dados, de um esquema XML da aplicação, de um metamodelo ou a partir do código fonte, por exemplo.

A informação é escavada a partir do que é chamado de morfologia do sistema. (HSI, 2005) define *Morfologia* de uma aplicação computacional como a apresentação externa de aplicação, consistindo nos elementos que acessíveis e perceptíveis pelo usuário. Em outras palavras, morfologia é uma representação da interface que o sistema faz com o usuário. Segundo (HSI, 2005), essa abordagem de escavação de modelos conceituais se aplica a extração de informações a partir dos seguintes tipos de interface: (1) interface por linha de comando, (2) interface gráfica e (3) algum dispositivo físico, cujos elementos morfológicos (elementos que fazem parte da morfologia) são os instrumentos físicos como *displays*, botões, etc.

Na descrição dessa abordagem será apresentada com o objetivo de ilustração a escavação do modelo conceitual da ferramenta de áudio *CD Player*, ferramenta integrante do pacote de aplicações básicas do sistema operacional Microsoft Windows, versões 95 e 98. Essa escavação foi usada como exemplo para elucidação da técnica em (HSI, 2005). No restante desse trabalho a descrição do processo de escavação de modelos conceituais será apresentado tomando como base a extração de informações a partir da interface gráfica de uma aplicação, visto que essa foi a abordagem utilizada nesse trabalho. As abordagens de escavação usando interface de linha de comando ou interface feita a partir de algum dispositivo físico não serão tratadas nesse trabalho.

O processo de escavação de ontologias se divide basicamente em duas etapas principais: (1) criação do mapa morfológico e (2) criação do modelo conceitual a partir do mapa morfológico. Nas próximas seções serão descritos com mais detalhes esses dois passos para a obtenção de um modelo conceitual.

3.1.3.1.1 Criação do Mapa Morfológico

O mapa morfológico consiste em um grafo conectado de elementos morfológicos e conexões. Os elementos morfológicos são quaisquer componentes visíveis e distintos na morfologia que permitem interações com o sistema de forma estática ou dinâmica. As conexões são setas que mostram como os elementos são acessados na morfologia pelo usuário e como os elementos estruturam outros elementos (HSI, 2005). Esse mapa é construído a partir de uma navegação extensa pela aplicação, abordando todas as possíveis interações disponíveis. Para fazer a modelagem desse mapa, foi usado o software Microsoft Visio 2003 e

até o presente momento não foi desenvolvida nenhuma técnica que fizesse esse mapeamento de forma automática.

Para a construção do Mapa Morfológico devem ser seguidos cinco passos principais (HSI, 2005):

1. Analisar a morfologia;
2. Identificar os elementos morfológicos;
3. Diagramar os componentes em um mapa;
4. Diagramar as conexões;
5. Diagramar as navegações

3.1.3.1.1 Passo 1: Analisar a morfologia

Antes de qualquer forma de modelagem, a interface com o usuário deve ser verificada extensivamente a fim de tentar identificar todas as opções de interação disponíveis na aplicação. Dessa forma é garantida a cobertura de todos os conceitos usados no sistema escavado. A Figura 3 mostra a interface principal do CD Player.

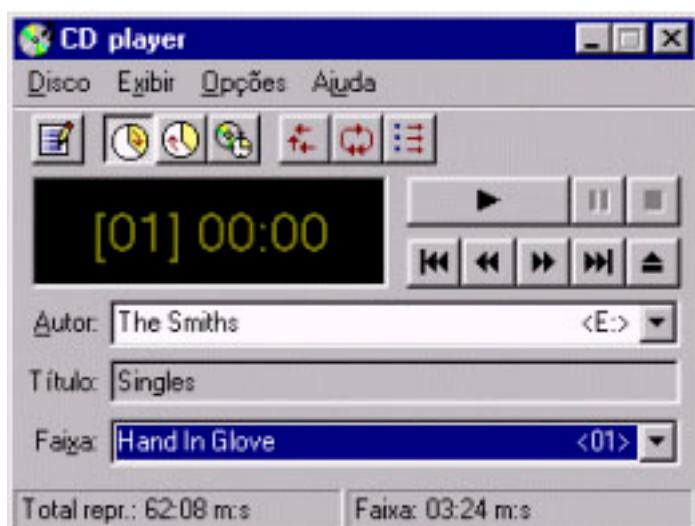


Figura 3 - Morfologia do CD Player

3.1.3.1.2 Passo 2: Identificar os elementos morfológicos

Os elementos morfológicos são representados no mapa com um ícone visual e sua descrição na interface gráfica de um sistema. Esses elementos são identificados a partir de

interação direta com o sistema. Estes são apresentados na morfologia através de elementos presentes na tela principal da aplicação (como botões, listas, caixas de texto (*text fields*)), como também através dos elementos encontrados a partir dos menus do sistema (caso possua).

(HSI, 2005) classificou os elementos morfológicos em quatro categorias. Contudo, nesse trabalho foi feita a separação em três categorias:

- *Interactors*: elementos acionados pelo usuário que realizam operações na aplicação (ex. botões, *text fields*, *checkboxes*);
- *Containers*: elementos que armazenam e estruturam os *interactors* (ex. janelas, caixas de diálogo, barras de ferramentas);
- *Displays*: elementos que apresentam ao usuário dados estáticos ou dinâmicos referentes a estados da aplicação (ex. *displays* de texto, barras de *status*).

A Figura 4 apresenta uma tela do CD Player com exemplos de alguns elementos morfológicos. A sintaxe desses elementos foram abstraídos do *framework* de componentes *Swing* da linguagem de programação Java (SUN, 2007).

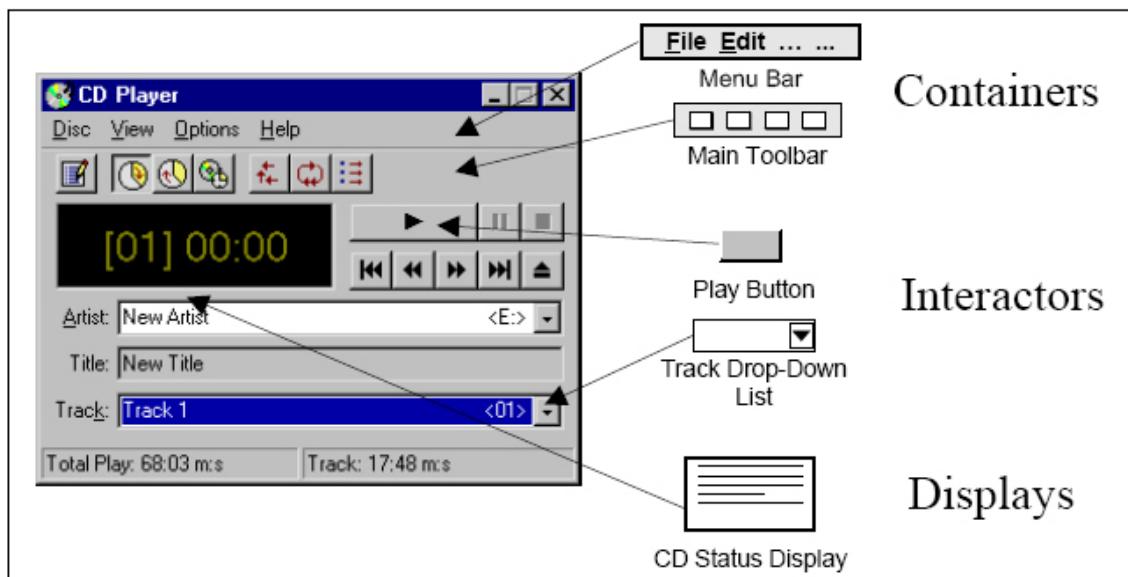


Figura 4 - Morfologia do CD Player com exemplos de elementos morfológicos

3.1.3.1.1.3 Passo 3: Diagramar os componentes em um mapa

Os elementos morfológicos identificados são diagramados usando símbolos e rótulos. Esses rótulos são apresentados de forma abreviada para identificar os símbolos e o seu conteúdo. Em alguns casos, as informações presentes na interface gráfica não são suficientes para que o elemento tenha uma identificação única. Dessa forma, podem ser inseridas mais informações no rótulo de forma a melhorar a descrição. Essas “novas” informações devem ser inseridas entre colchetes. A Figura 5 mostra alguns dos elementos morfológicos do CD Player.

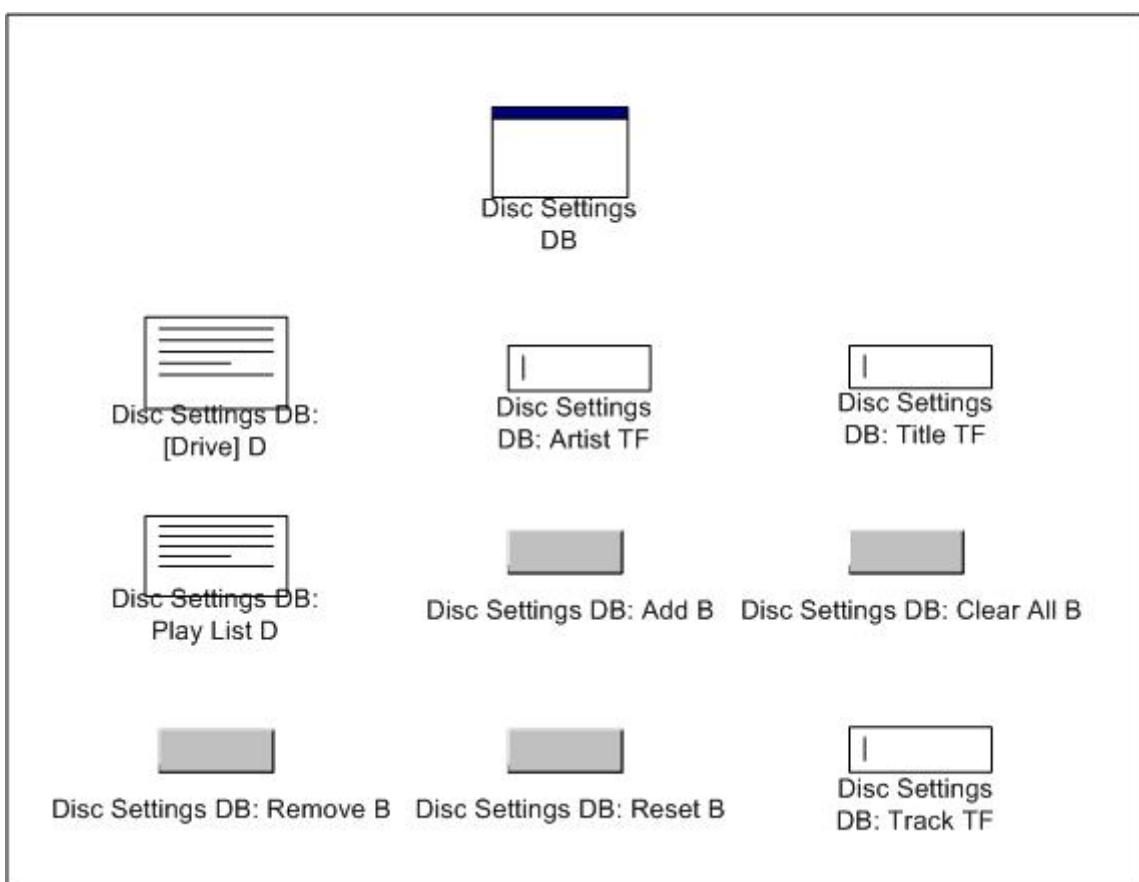


Figura 5 - Alguns elementos morfológicos do CD Player

A Figura 5 mostra o elemento morfológico “*Disc Settings DB: Artist TF*”. Esse rótulo indica que existe um *Dialog Box (DB)* rotulada como “*Disc Settings*” e dentro dela um *Text Field (TF)* rotulada com “*Artist*”, que indica qual o nome do artista principal do CD.

3.1.3.1.1.4 Passos 4 e 5: Diagramar as conexões e as navegações

Os elementos no mapa devem ser ligados através de setas direcionadas para mostrar seus *containers* (ex: uma barra de ferramentas contendo botões) ou seus pontos de ativação

(ex: um item de menu abrindo uma caixa de diálogo) (HSI, 2005). A Figura 6 mostra a navegação de alguns elementos no CD Player.

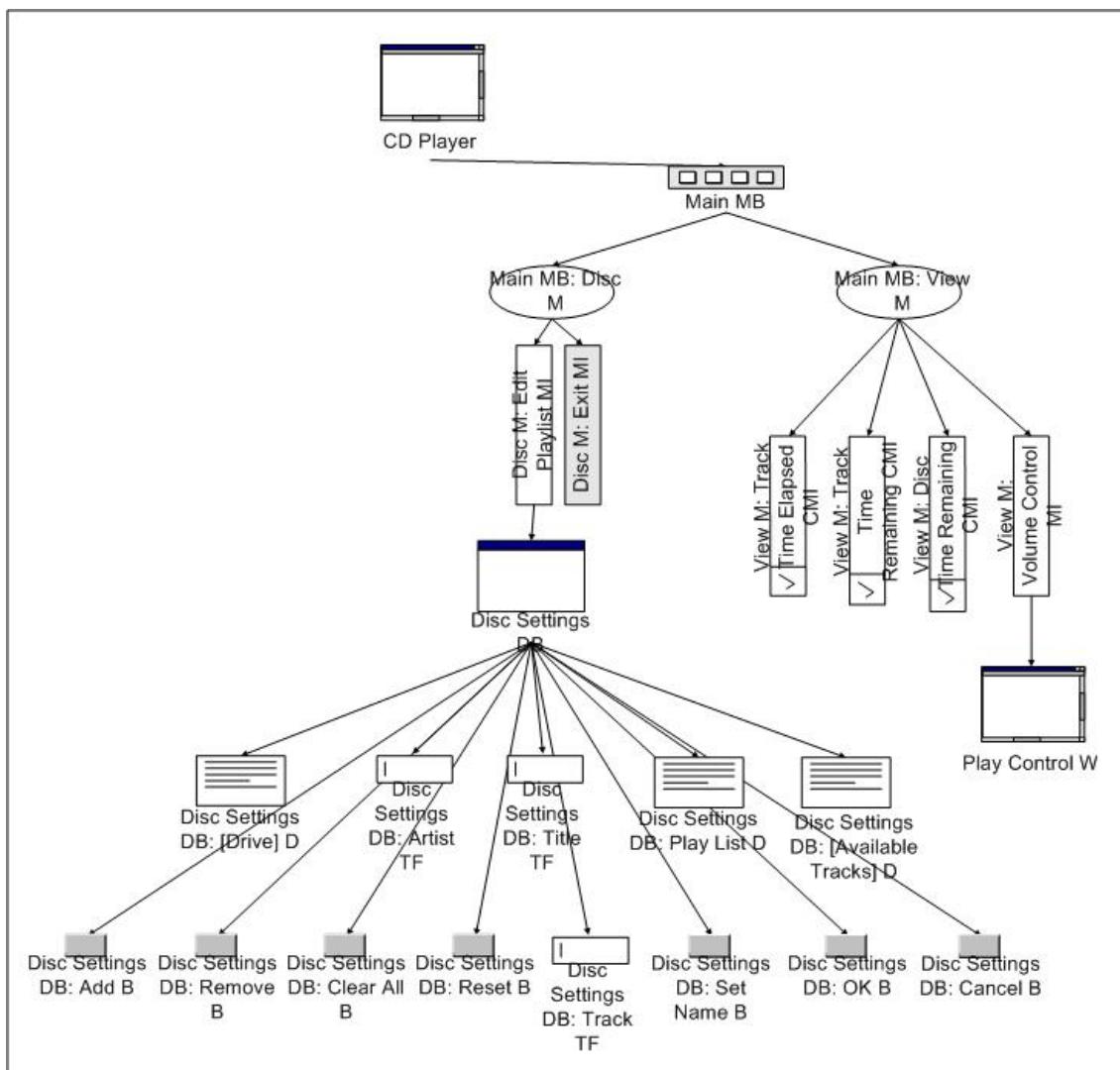


Figura 6 - Exemplos de navegação entre elementos morfológicos

Como podem ser notados, os rótulos dos símbolos seguem o seguinte padrão:

<nome do container> <abreviação do container>:

<nome do elemento> <abreviação do elemento>

O container me questão é o elemento que foi acionado para que o elemento subsequente fosse ativado. Esse padrão garante a rastreabilidade dos elementos morfológicos.

A Figura 6 mostra o elemento morfológico “Disc Settings DB: Title TF” que indica que o *Dialog Box (DB)* *Disc Settings* referencia o *Text Field (TF)* *Title*.

Para facilitar o entendimento de como são modeladas as navegações, a Figura 7 apresenta uma parte do mapa morfológico, onde cada um dos seus elementos morfológicos está apontando para o seu respectivo componente na tela principal da aplicação.

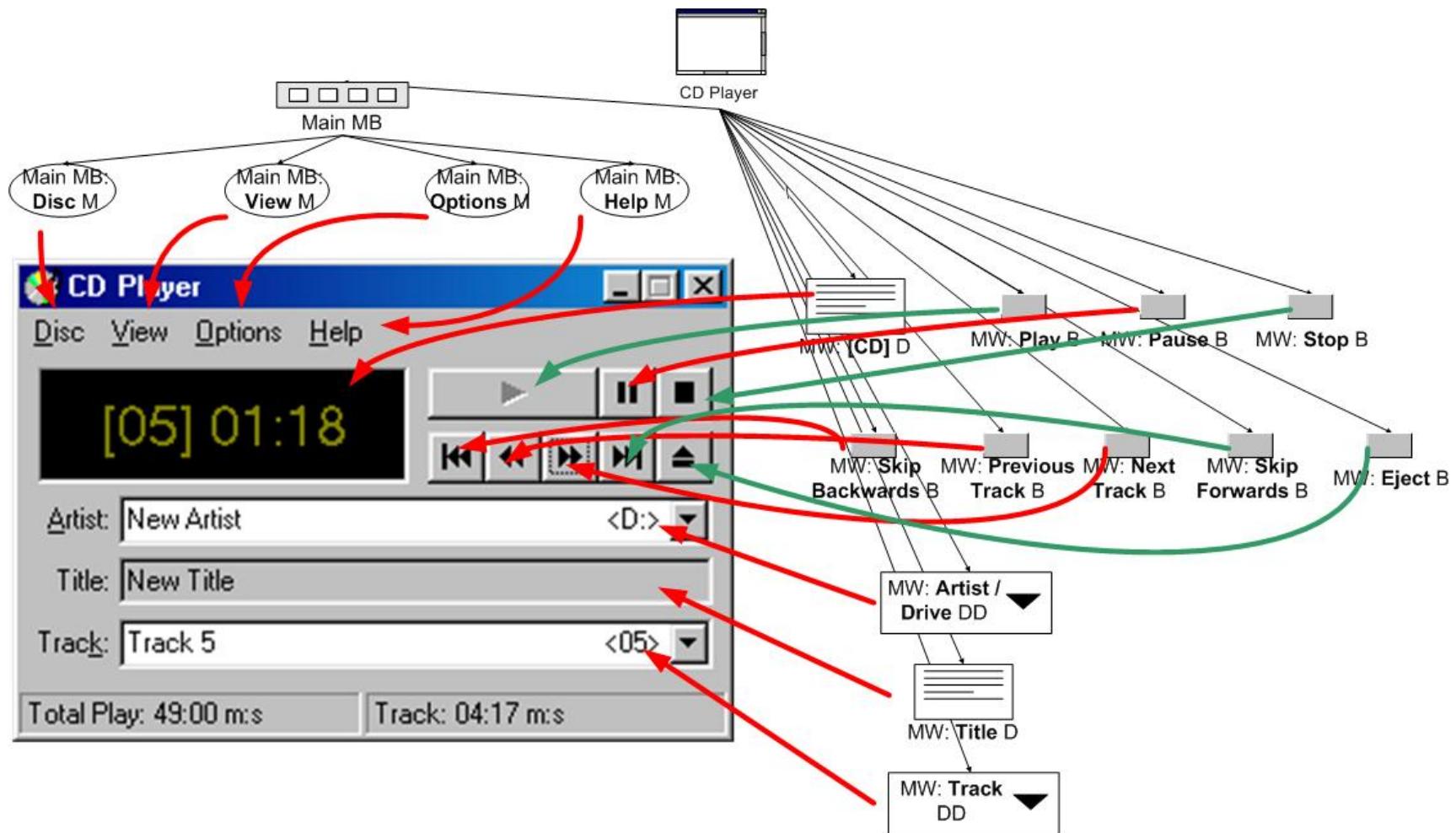


Figura 7 - Parte do Mapa Morfológico do CD Player com sua interface principal

A Figura 7 apresenta parte do mapa morfológico do CD Player refletindo a navegação entre os conceitos. O ícone superior (rotulado como “CD Player”) representa a aplicação principal. A aplicação tem duas formas de acessar as funcionalidades pela interface, a saber, 1) através dos menus e 2) através dos botões e demais formas de interação presentes na interface principal. Para as funcionalidades acessadas via menu, é modelado um *container* que representa a barra principal de menu (Main MB – *Menu Bar*) e os itens presentes nos menus estarão relacionados a partir dessa *container*. Já para as outras funcionalidades, a aplicação principal faz papel de *container*, já que esses acessos a funcionalidades não estão “amarrados” a nenhum *container* intermediário, por isso, a navegabilidade está refletida do ícone da aplicação principal para esses ícones representando as funcionalidades presentes na interface principal. Para cada um dos elementos morfológicos presentes na figura, o seu correspondente foi mapeado na interface principal através de setas direcionadas. Para melhorar a visualização, foram usadas setas de cores diferentes para não deixar a figura muito confusa.

3.1.3.1.1.5 Estabelecendo fronteiras na modelagem

Aplicações freqüentemente acessam funcionalidades dos sistemas operacionais, a internet e/ou outras aplicações. Analisar morfologias além das fronteiras do sistema estudado poderá produzir um modelo conceitual errado, que conterá muitos conceitos que não pertencem tecnicamente à aplicação (HSI, 2005).

Os elementos morfológicos que podem ser facilmente identificados como específicos do sistema operacional, ou seja, não construídos juntos com o sistema estudado, não são modelados, pois esses conceitos são específicos do sistema e não da aplicação escavada. Alguns exemplos comuns são as janelas que manipulam a impressão de documentos, as que manipulam a abertura e salvamento de arquivos, etc. Há também aplicações que usam funcionalidades de outras aplicações através de objetos embutidos. Um exemplo é o Microsoft Word que pode abrir uma planilha do Microsoft Excel como um objeto e utiliza todas as operações providas pelas planilhas do Excel. Assim, todas as funcionalidades que não são nativas do sistema estudado não devem ser modeladas.

A Figura 8 apresenta o mapa morfológico do CD Player finalizado.

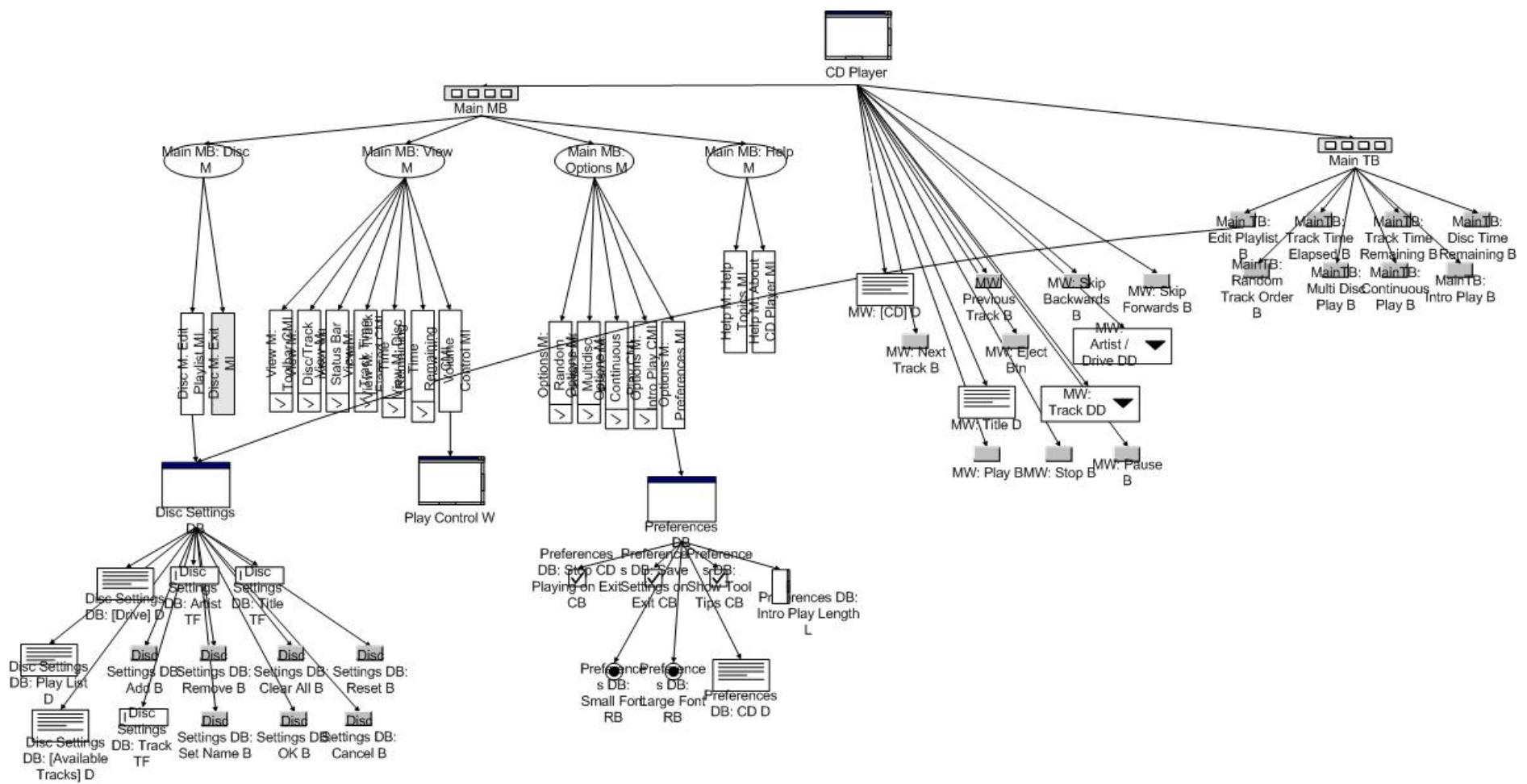


Figura 8 - Mapa Morfológico do CD Player

3.1.3.1.2 Construção do Modelo Conceitual

Nessa subseção será apresentado o método de construção do modelo conceitual através do mapa morfológico. Inicialmente conceitos são identificados a partir dos elementos morfológicos apontados no passo anterior. Em seguida, com uma combinação de interações estáticas e dinâmicas com a aplicação, são identificados como esses conceitos estão relacionados. Por fim, é modelado o modelo conceitual da ferramenta como uma Rede Semântica composta desses conceitos extraídos e seus relacionamentos.

Segundo (HSI, 2005), os passos básicos para a construção do modelo conceitual são:

1. **Escavar conceitos:** Identificar nomes e objetos analisando os rótulos dos elementos morfológicos;
2. **Identificar relacionamentos:** Para cada conceito, identificar o relacionamento que este tem com outros conceitos no modelo conceitual examinando e interagindo com a interface gráfica;
3. **Montar o Modelo Conceitual:** Representar conceitos e relacionamentos em uma rede semântica.

3.1.3.1.2.1 Passo1: Escavar conceitos

Segundo (SMITH, 1981), um conceito é uma idéia generalizada de um ser ou uma classe de seres. Nessa abordagem, um conceito é tudo que tem um nome ou alguma aplicação concreta na aplicação de interesse (HSI, 2005). Esses conceitos são organizados em três categorias: tipos de entidade, atributos e instâncias.

- Uma *entidade* é um ser nomeado que pode ser identificado de forma distinta (CHEN, 1976). Um conjunto de entidades que compartilham um conjunto de atributos é um tipo de entidade (ELMASRI, 1994). Exemplos de entidades podem ser Recurso, Atividade, Projeto, etc.
- Um *atributo* é uma propriedade intrínseca de um ser no mundo real (WAND, 1999). Exemplos de atributos podem ser Nome, Duração, Tipo, etc.
- Uma *instância* é uma manifestação concreta de um tipo de entidade (BOOCH, 1999) e essas não serão modeladas no modelo conceitual, pois para o propósito deste trabalho, os modelos conceituais não precisam chegar a esse nível de abstração.

3.1.3.1.2.1.1 Identificando Conceitos

A partir do mapa morfológico os conceitos são identificados baseados nos rótulos dados aos elementos morfológicos. A Figura 9 apresenta essa identificação mostrando parte do mapa morfológico e como os conceitos podem ser extraídos a partir de verificação dos rótulos.

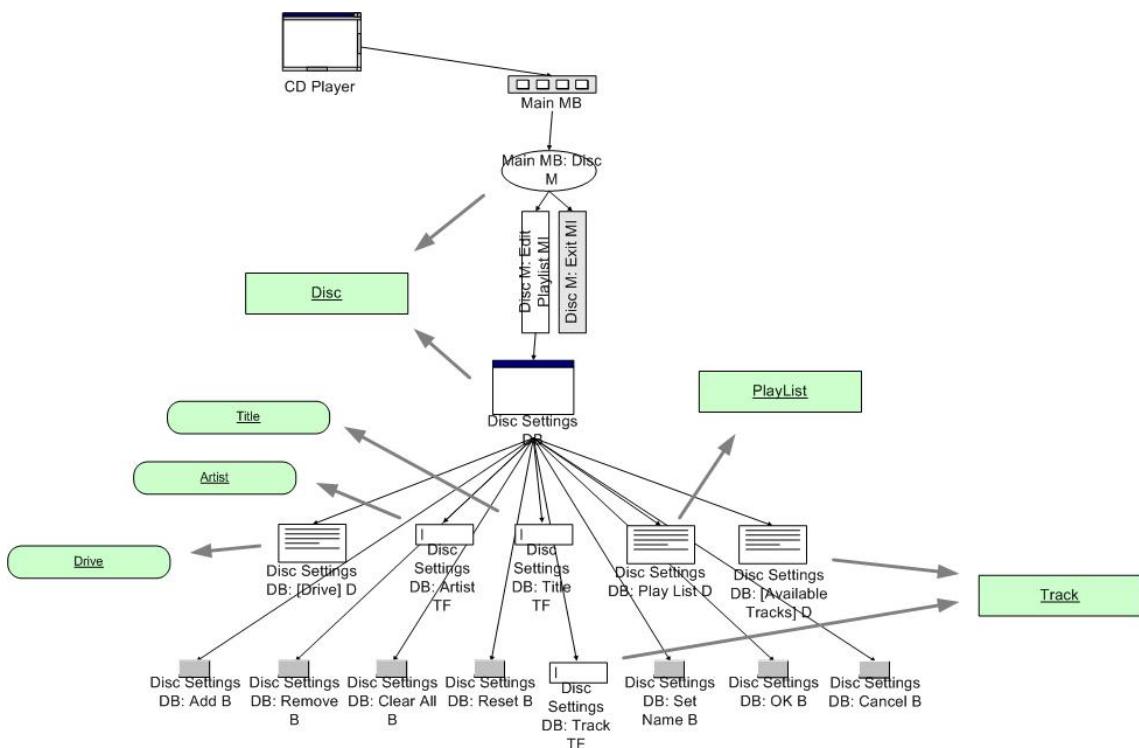


Figura 9 - Extração de conceitos a partir do mapa morfológico

As entidades em destaque são os conceitos extraídos do modelo. Como podem ser notados, esses conceitos são obtidos a partir da visualização do rótulo do elemento morfológico. Por exemplo, o elemento “*Disc Settings DB*” refere-se à existência de uma caixa de diálogo (*DB – Dialog Box*) que apresenta informações sobre uma faixa de música (*Track*). Dessa forma, é inferida a existência do conceito Faixa (*Track*). Assim, as entidades e atributos são identificados para que em seguida seja identificada a forma de como esses conceitos se relacionam. Como pode ser notado, é necessário um conhecimento prévio do domínio para que possa ser inferido se os conceitos extraídos da morfologia são entidades e atributos. Dessa forma, assim como em todo o processo, o responsável por esse mapeamento precisa ter um conhecimento relativo ao domínio.

3.1.3.1.2.2 Passo 2: Identificar relacionamentos

Depois de identificados os conceitos da morfologia, devem ser verificados como esses se relacionam. Essa verificação deve ser feita a partir da interação com o sistema, reconstruindo os passos dados pelo usuário que resultaram no mapa morfológico. Para a construção da rede semântica, são usados os relacionamentos básicos de modelagem de objetos: generalização (relação “*is-a*”), agregação (relação “*has-a*”) e associação (HSI, 2005).

3.1.3.1.2.2.1 Generalização

Uma generalização é um relacionamento entre um tipo de ser (uma superclasse, por exemplo) e um tipo mais específico desse ser (uma subclasse, por exemplo). Um subtipo é uma especialização de um tipo de entidade (HSI, 2005). No paradigma orientado a objetos, uma generalização indica que a classe filha herda todos os atributos da classe pai. Esse relacionamento é assumido nesse trabalho quando são identificados muitos objetos com o mesmo conjunto de atributos. Generalizações são modeladas usando uma seta direcionada rotulada com “*is-a*” (é um) partindo de um tipo de entidade até o seu pai (HSI, 2005), como pode ser visto na Figura 10.

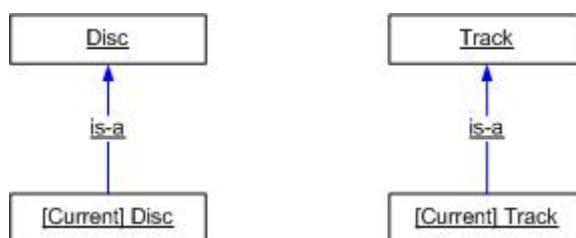


Figura 10 - Exemplo de generalização

3.1.3.1.2.2.2 Agregação

Uma agregação é um relacionamento todo-parte usado quando um conceito agrupa características de outros conceitos (entidades ou atributos). Os relacionamentos de agregação

são modelados com uma seta direcionada com o rótulo “*has-a*” (tem um), como pode ser visto na Figura 11.

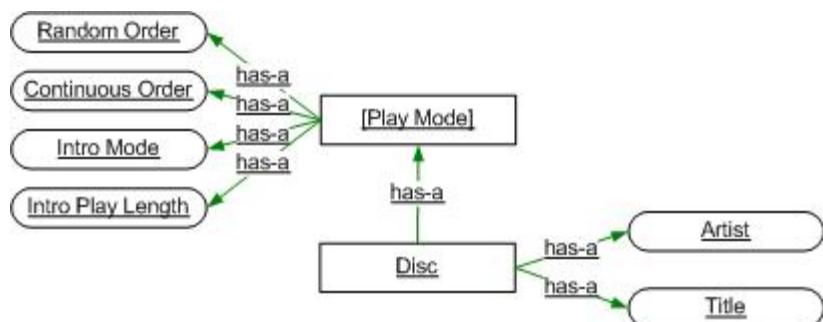


Figura 11 - Parte do modelo conceitual do CD Player com relações de agregação

A Figura 11 mostra que um relacionamento de agregação entre as entidades *Play Mode* (Modo de Execução) e *Disc* (Disco). Essa relação indica que um Disco possui um Modo de Execução (não será entrado no mérito de cardinalidade nessa modelagem por não ser relevante ao propósito deste trabalho). Outro tipo de agregação apresentado na figura é o entre uma entidade e um atributo. Um Disco possui um *Artist* (Artista) e um *Title* (Título).

Esses relacionamentos podem ser identificados em caixas de diálogo (*dialog boxes*) com especificações de algum item ou a partir de uma lista de propriedades associadas com o conceito em questão (HSI, 2005).

3.1.3.1.2.2.3 Associação

São usadas associações para indicar relacionamentos entre conceitos não cobertos por generalizações ou agregações, ou melhor, quando os relacionamentos entre dois conceitos não são do tipo “sub-tipo de” nem quando um conceito é um atributo de outro conceito. Uma associação é um relacionamento estrutural que especifica que um ser de um tipo interage com seres (conceitos) de outros tipos, como mostrado na Figura 12.

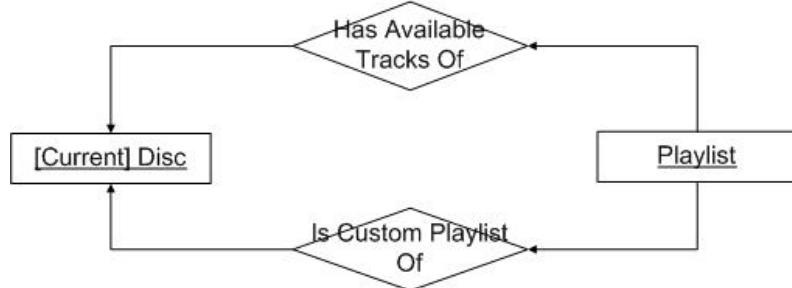


Figura 12 - Exemplo de relacionamento de associação no modelo conceitual

Muitos relacionamentos podem ser identificados apenas observando os elementos morfológicos, sem nenhuma interação com o sistema. Segundo o exemplo do CD Player, se estiver sendo apresentada uma tela com informações relativas a um “Disco”, alguns conceitos como “Nome”, “Artista”, “Duração”, etc., são relacionados como agregação para o conceito “Disco”. Por outro lado, alguns relacionamentos devem ser verificados a partir de uma manipulação dinâmica e extensa da aplicação. Ainda no mesmo exemplo, o relacionamento “*Is Custom Playlist Of*” entre os conceitos “[Current] Disc” e “Playlist” precisa de uma interação com o sistema para saber que uma lista de reprodução específica é uma lista customizada de um determinado disco que está sendo executado pelo sistema no momento. Segundo (HSI, 2005), visto que esse processo de escavação é feito de forma manual, a atenção do nível de detalhes na identificação dos relacionamentos pode afetar a precisão da modelagem.

3.1.3.1.2.3 Passo 3: Montar o Modelo Conceitual

Identificados os conceitos e relacionamentos, estes são reunidos em um só grafo, que representará o Modelo Conceitual dos conceitos do domínio em questão representados na ferramenta estudada. Os conceitos são modelados em formas de nós e os relacionamentos em formas de arestas direcionadas.

A Figura 13 apresenta o modelo conceitual completo do CD Player.

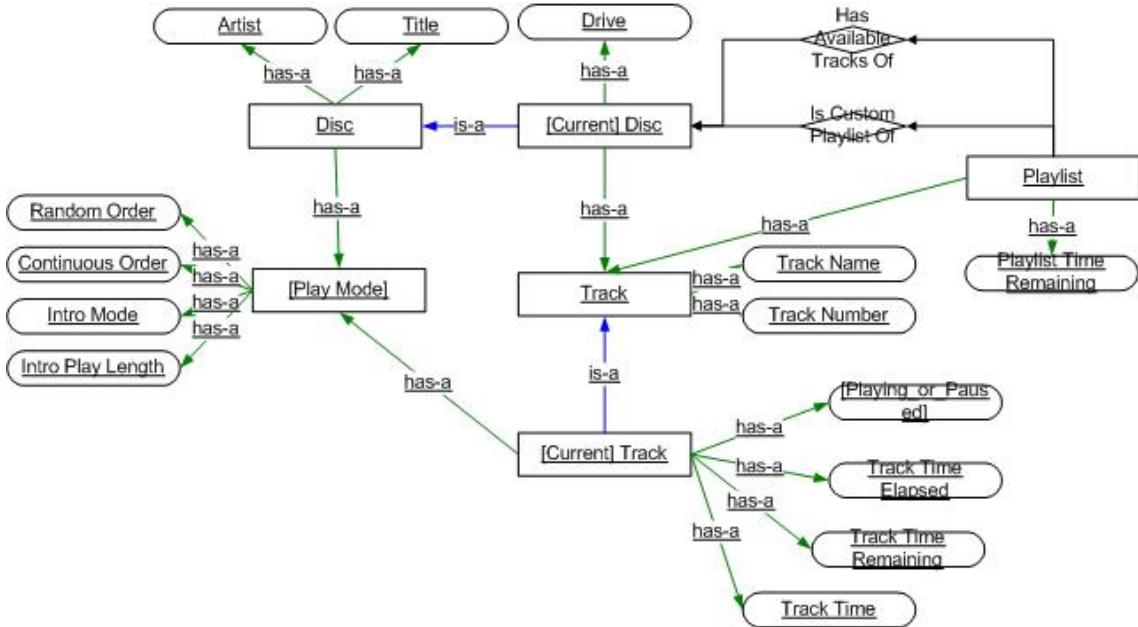


Figura 13 - Modelo Conceitual do CD Player

3.2 Abordagem

Este trabalho possui três objetivos principais: (1) explicitar a semântica subjacente a modelos conceituais de ferramentas de software, (2) avaliar se uma ferramenta pode ser usada no apoio de um processo de desenvolvimento de software e (3) apresentar uma integração semântica entre ferramentas que apóiam um processo de desenvolvimento.

Para atingir o primeiro e o segundo objetivos, será apresentada através de um caso de uso, descrito no Capítulo 4, a validação de uma ferramenta que apóia o gerenciamento de projetos que é amplamente utilizada por empresas de software, o Microsoft Project 2003. Para isso, inicialmente será feita a extração de seu modelo conceitual usando a técnica discutida nas sessões anteriores e definida em (HSI, 2005). Em seguida, serão apresentadas duas ontologias do domínio de Engenharia de Software, a saber, de Processos de Software e Riscos que serão tomadas como de referência para esses respectivos domínios. Essas ontologias serão usadas como base de conhecimento para que o modelo conceitual escavado da ferramenta seja fundamentado semanticamente. Dessa forma, a ferramenta poderá ser avaliada como apta ou não para apoiar um processo de desenvolvimento. Será feito um mapeamento conceitual entre o modelo conceitual extraído com as ontologias de referência e apresentadas as devidas conclusões.

Um ponto importante a ser observado na extração do modelo conceitual de ferramentas complexas como o Microsoft Project é que elas normalmente manipulam muitos conceitos e a maioria deles não estão relacionados ao domínio que a ferramenta está proposta

a abordar. Grande parte da ferramenta lida com conceitos relativos especificamente ao software, como visualização de informações, configurações da ferramenta, manipulação de arquivos, tópicos de ajuda, disposição de janelas, etc. Um problema nessas ferramentas é que os conceitos relativos ao domínio normalmente estão misturados com os específicos de software. Assim, em algumas situações é difícil diferenciar quando um conceito é relativo o domínio ou da aplicação.

Se tratando da construção do modelo conceitual da ferramenta, pode não ser necessária a escavação de todos os conceitos presentes na interface. Dessa forma, se todos os conceitos forem extraídos, o modelo final ficaria desnecessariamente extenso e complexo. Assim, para o propósito deste trabalho, apenas uma parte das funcionalidades da aplicação serão escavadas, ou melhor, apenas as funcionalidades relacionadas ao domínio de conhecimento que será abordado serão verificadas.

Para se fazer a distinção entre os conceitos a serem extraídos, a abordagem adotada foi definir alguns casos de uso de domínio e verificar como a ferramenta se comporta para atendê-los. Como essa e muitas das ferramentas usadas atualmente são baseadas em um conjunto de menus, a construção do modelo vai se basear nas informações contidas nos menus navegados que contêm as funcionalidades necessárias da ferramenta a fim de executar os casos de uso propostos. No Capítulo 4 serão apresentados os casos de uso usados para delimitar quais as funcionalidades a serem abordadas na escavação do modelo conceitual do Microsoft Project. Para cada um dos casos de uso, serão listadas as ações principais a serem executadas e para cada ação, serão listados os menus acessados na ferramenta para executar o caso de uso.

Para o terceiro objetivo, será apresentado um exemplo de integração conceitual entre duas ferramentas que apóiam processos de software, a saber, o Microsoft Project e o CVS, referente ao domínio de gerência de configuração. Para isso, serão apresentados uma ontologia de Gerência de Configuração de Software (GCS) e o modelo conceitual extraído do CVS. A partir disso, será feito um mapeamento entre o modelo conceitual do CVS e a ontologia de GCS, a fim de mostrar que essa ferramenta está apta para apoiar as atividades relacionadas à gerência de configuração. Em seguida, será feito um mapeamento entre as ontologias de Processos de Software e GCS, com o intuito de mostrar que esses domínios possuem pontes semânticas, logo, ferramentas que abordam esses domínios podem se comunicar. Por fim, será mostrado o mapeamento conceitual entre os modelos conceituais das ferramentas Microsoft Project e CVS e apresentadas as devidas conclusões.

Capítulo 4

Estudo de Caso

Esse capítulo apresentará um estudo de caso abordando os principais conceitos apresentados nesse trabalho nos capítulos anteriores. Será apresentada a escavação do modelo conceitual da ferramenta de gerência de projetos Microsoft Project 2003¹. A extração desse modelo tem como objetivo apresentar como a ferramenta manipula os conceitos relacionados ao domínio que a ferramenta aborda. A partir do modelo conceitual escavado, será apresentado um mapeamento entre o modelo, uma ontologia de Processos de Software desenvolvida em (FALBO, 1998) e atualizada em (BERTOLLO, 2006) e uma ontologia de Riscos de Software desenvolvida em (FALBO et al., 2004). Esse mapeamento tem como objetivo explicitar a semântica dos conceitos subjacentes à ferramenta e avaliar se uma ferramenta pode ser usada de forma adequada para apoiar um processo de desenvolvimento de software.

Esse capítulo está organizado da seguinte forma: a seção 4.1 faz uma breve apresentação do domínio de Gerência de Projetos, abordando apenas os conceitos que serão manipulados pelos casos de uso descritos na seção 4.2. A seção 4.2 apresenta os casos de uso abordados para restringir a escavação do modelo conceitual da ferramenta. A seção 4.3 apresenta o resultado da escavação do modelo conceitual da ferramenta Microsoft Project. Por fim, as seções 4.4 e 4.5 apresentam as ontologias de referência dos domínios de Processos e Riscos de Software, respectivamente e por fim, na seção 4.6 é apresentado um mapeamento entre as ontologias de referência e o modelo conceitual extraído da ferramenta.

4.1 Gerência de Projetos

Projeto é um empreendimento temporário que tem por finalidade criar um produto, serviço ou resultado único. Temporário, pois todo o projeto possui início e fim definidos – tem duração finita, onde esse fim só é alcançado quando: (i) os objetivos do projeto tiverem sido atingidos, (ii) quando se torna claro que os objetivos não serão ou não poderão ser mais

¹ <http://office.microsoft.com/pt-br/project/>

atingidos, (iii) ou quando não houver mais a necessidade do projeto. Único porque apesar de suas propostas finais muitas vezes possuírem a presença de vários elementos comuns a resultados de outros projetos, possuem pelo menos um tipo de singularidade, como por exemplo, apesar de existirem variados softwares que executam uma mesma funcionalidade, esses possuem fabricantes diferentes, processos de softwares diferentes, tempo de execução diferentes, e outras características que os tornam únicos (PMI, 2004).

Projetos são decompostos em atividades a serem executadas a fim de atingir o objetivo do projeto. Para cada uma das atividades são designados recursos responsáveis pela execução desta. Esses recursos podem tanto ser recursos humanos (Gerentes de Projetos, Programadores, Analistas, etc.), quanto recursos materiais (Ferramentas de Software, Servidores, etc. ou seja, recursos de hardware e recursos de software). Uma atividade, durante sua execução, pode consumir ou produzir artefatos que serão usados durante o processo de desenvolvimento. Exemplos de artefatos podem ser Documento de Análise e Projeto do sistema, Plano do Projeto, Plano de Testes, etc.

Projetos podem possuir Riscos associados, ou seja, eventos que podem ocorrer que não estão nos planos do projeto. Contudo, esses riscos podem ser mapeados e formas de mitigação e contingência podem ser aplicados para tratá-los. Esses riscos podem estar relacionados também a Atividades e Artefatos.

4.2 Casos de Uso

Para a escavação da ferramenta Microsoft Project foram definidos cinco casos de uso considerados representativos o suficiente para abranger o domínio de Processos de Software e as principais funcionalidades da ferramenta. Os casos de uso foram definidos como: *Cadastrar Projeto*, *Cadastrar Atividade*, *Cadastrar Recurso*, *Cadastrar Artefato* e *Cadastrar Riscos*. A Figura 14 apresenta o Diagrama de Casos de Uso que mostra os casos de uso propostos.

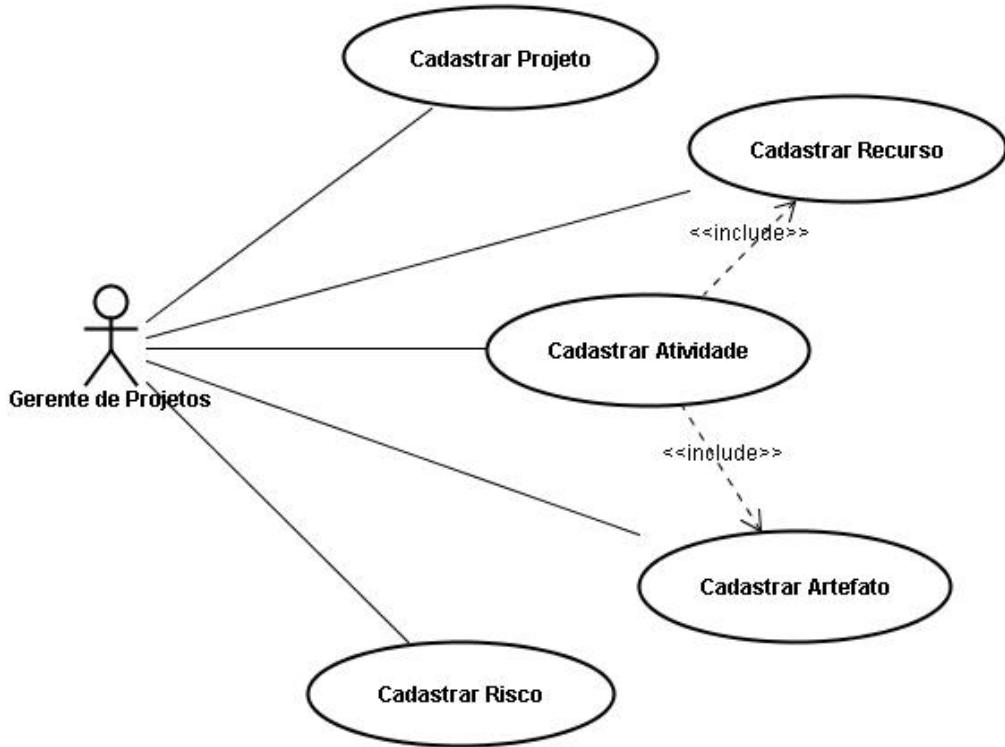


Figura 14 - Diagrama de Casos de Uso com as fronteiras de escavação da ferramenta Microsoft Project

Para cada um dos casos de uso, são definidos eventos que podem ser executados pelo ator, referentes às funcionalidades da aplicação. Como pode ser visto na Figura 14, o caso de uso Cadastrar Atividade inclui (através do relacionamento <<include>>) os casos de uso Cadastrar Recurso e Cadastrar Artefato. Isso indica que alguns eventos relativos aos casos de uso incluídos podem ser executados pelo ator quando este está executando o caso de uso que os inclui.

A seguir serão apresentados os casos de uso com uma breve descrição e seus respectivos eventos que podem ser executados pelo Gerente de Projetos. Para cada um destes eventos são listados os menus a serem seguidos na ferramenta para acessar as funcionalidades que proverão os conceitos do domínio a serem modelados.

Caso de Uso: Cadastrar Projeto

Descrição: Este caso de uso é responsável pelo cadastro de um projeto, permitindo que o Gerente de Projetos possa inserir um novo projeto, editar e/ou excluir um projeto existente. Esse caso de uso é dividido em dois eventos principais: *Inserir Projeto* e *Editiar Projeto*. Para cada um desses há um conjunto de menus no sistema a serem seguidos para que esses eventos sejam realizados.

Cursos Normais:

Inserir Projeto

O Gerente de Projetos adiciona um novo projeto a ser gerenciado.

Menus:

Barra de Menu Principal → File → New

Barra de Menu Principal → File → Properties

Barra de Menu Principal → Project → Project Information

Editar Projeto

O Gerente de Projetos modifica alguma informação do projeto.

Menus:

Barra de Menu Principal → File → Properties

Barra de Menu Principal → Project → Project Information

Barra de Menu Principal → Tools → Tracking → Update Project

Caso de Uso: Cadastrar Atividade

Descrição: Este caso de uso é responsável pelo controle das atividades do projeto, permitindo que o Gerente de Projetos possa inserir uma nova atividade, editar ou excluir uma atividade existente, bem como alocar recursos e artefatos a uma atividade. Esse caso de uso é dividido em cinco eventos principais: *Inserir Atividade*, *Edita Atividade*, *Excluir Atividade*, *Assinalar Recurso à Atividade* e *Assinalar Artefato à Atividade*. Esses dois últimos eventos são referentes aos casos de uso Cadastrar Recurso e Cadastrar Artefato, respectivamente, mas são incluídos neste caso de uso, pois esses dois eventos podem ser executados a partir desse caso de uso. Para cada um desses há um conjunto de menus no sistema a serem seguidos para que esses eventos sejam realizados.

Cursos Normais:

Inserir Atividade

O Gerente de Projetos adiciona uma nova atividade a ser executada durante o projeto.

Menus:

Barra de Menu Principal → Insert → New Task

Barra de Menu Principal → Project → Task Information

Editar Atividade

O Gerente de Projetos modifica as informações de alguma atividade relacionada ao projeto.

Menus:

Barra de Menu Principal → Project → Task Information

Barra de Menu Principal → Tools → Tracking → Update Tasks

Excluir Atividade

O Gerente de Projetos exclui uma atividade relacionada ao projeto.

Menus:

Barra de Menu Principal → Edit → Delete Task

Assinalar Recurso à Atividade

O Gerente de Projetos aloca um recurso necessário para a execução de uma atividade específica.

Menus:

Barra de Menu Principal → Tools → Assign Resources

Assinalar Artefato à Atividade

O Gerente de Projetos assinala algum artefato que será consumido ou que foi produzido pela atividade.

Menus:

Barra de Menu Principal → Collaborate → Documents → Interface com Project Server → Documents → Escolher Atividade → Link Document

Caso de Uso: Cadastrar Recurso

Descrição: Esse caso de uso é responsável pelo controle dos recursos envolvidos no projeto, permitindo que o Gerente de Projetos possa inserir um novo recurso, editar ou excluir um recurso existente, bem como alocar um recurso a uma atividade e relacionar um artefato específico a um recurso. Esse caso de uso é dividido em cinco eventos principais: *Inserir Recurso*, *Editar Recurso*, *Excluir Recurso*, *Assinalar Recurso à Atividade* e *Assinalar*

Artefato à Recurso. Para cada um desses há um conjunto de menus no sistema a serem seguidos para que esses eventos sejam realizados.

Cursos Normais:

Inserir Recurso

O Gerente de Projetos adiciona um novo recurso que será usado em alguma etapa na execução do projeto.

Menus:

Barra de Menu Principal → View → Resource Sheet → Lista de Recursos → Barra de Menu Principal → Add new Resource

Editar Recurso

O Gerente de Projetos altera as informações de algum recurso que está vinculado ao projeto.

Menus:

Barra de Menu Principal → View → Resource Sheet → Lista de Recursos → Dois cliques no Recurso

Excluir Recurso

O Gerente de Projetos exclui um recurso, desvinculando-o do projeto.

Menus:

Barra de Menu Principal → Edit → Remove Resource

Assinalar Recurso à Atividade

O Gerente de Projetos aloca um recurso necessário para a execução de uma atividade específica.

Menus:

Barra de Menu Principal → Tools → Assign Resources

Assinalar Artefato à Recurso

O Gerente de Projetos relaciona algum artefato com algum recurso que está relacionado ao projeto.

Menus:

Barra de Menu Principal → Collaborate → Documents → Interface com Project Server → Documents → Escolher Recurso → Link Document

Caso de Uso: Cadastrar Artefato

Descrição: Este caso de uso é responsável pelo controle dos artefatos produzidos e consumidos durante a execução do projeto. Este permite que o Gerente de Projetos possa cadastrar novos artefatos, excluir artefatos existentes, relacionar artefatos a atividades e relacionar artefatos a recursos. Esse caso de uso é dividido em quatro eventos principais: *Inserir Artefato*, *Excluir Artefato*, *Assinalar Artefato à Atividade* e *Assinalar Artefato à Recurso*. Para cada um desses há um conjunto de menus no sistema a serem seguidos para que esses eventos sejam realizados.

Inserir Artefato

O Gerente de Projetos adiciona um novo artefato relacionado ao projeto.

Menus:

Barra de Menu Principal → Collaborate → Documents → Interface com Project Server → Documents → Shared Documents → Upload Document

Excluir Artefato

O Gerente de Projetos exclui um artefato que está relacionado ao projeto.

Menus:

Barra de Menu Principal → Collaborate → Documents → Interface com Project Server → Documents → Shared Documents → Selecionar documento a ser excluído → Excluir Documento

Assinalar Artefato à Atividade

O Gerente de Projetos assinala algum artefato que será consumido ou que foi produzido pela atividade.

Menus:

Barra de Menu Principal → Collaborate → Documents → Interface com Project Server → Tasks → Escolher atividade → Link Documents → Escolher artefato

Assinalar Artefato à Recurso

O Gerente de Projetos relaciona algum artefato com algum recurso que está relacionado ao projeto.

Menus:

Barra de Menu Principal → Collaborate → Documents → Interface com Project Server → Resources → Escolher recurso → Link Documents → Escolher artefato

Caso de Uso: Cadastrar Risco do Projeto

Descrição: Esse caso de uso é responsável pelo controle dos riscos relacionados ao projeto. Este permite que o Gerente de Projetos relate riscos ao projeto, a atividades e a artefatos específicos. Esse caso de uso é dividido em cinco eventos principais: *Inserir Risco a Projeto, Inserir Risco a Atividade, Inserir Risco a Artefato, Editar Risco e Excluir Risco*. Para cada um desses há um conjunto de menus no sistema a serem seguidos para que esses eventos sejam realizados.

Inserir Risco a Projeto

O Gerente de Projetos adiciona um novo risco relacionado ao projeto.

Menus:

Barra de Menu Principal → Collaborate → Risks → Interface com Project Server → Risks → Escolher Projeto → New Risk

Inserir Risco a Atividade

O Gerente de Projetos adiciona um novo risco relacionado ao projeto e seleciona as atividades influenciadas por esse risco.

Menus:

Barra de Menu Principal → Collaborate → Risks → Interface com Project Server → Risks → Escolher Projeto → New Risk → selecionar as atividades relacionadas a esse risco

Inserir Risco a Artefato

O Gerente de Projetos adiciona um novo risco relacionado ao projeto e seleciona os artefatos influenciados por esse risco.

Menus:

Barra de Menu Principal → Collaborate → Risks → Interface com Project Server → Risks → Escolher Projeto → New Risk → selecionar os artefatos relacionados a esse risco

Editar Risco

O Gerente de Projetos modifica as informações relativas a um risco.

Menus:

Barra de Menu Principal → Collaborate → Risks → Interface com Project Server → Risks → Escolher Risco → alterar as informações necessárias do risco

Excluir Risco

O Gerente de Projetos julga que o risco não é mais um risco e o exclui da lista de riscos.

Menus:

Barra de Menu Principal → Collaborate → Risks → Interface com Project Server → Risks → Escolher Risco → Excluir Risco

4.3 Escavação da ferramenta Microsoft Project

Seguindo os passos descritos no Capítulo 3, seção 3.3.3.1, serão apresentadas as fases intermediárias para a escavação da ferramenta Microsoft Project 2003. Como já apresentado, o processo de escavação de modelos conceituais se divide basicamente em dois passos principais: criação do Mapa Morfológico e a Criação do Modelo Conceitual. Os conceitos subjacentes à ferramenta apresentados nesse mapa está limitada pelos conceitos relativos aos casos de uso descritos na seção anterior.

4.3.1 Mapa Morfológico

O Mapa Morfológico consiste em um grafo conectado de elementos morfológicos e suas conexões. Os elementos morfológicos são quaisquer componentes visíveis e distintos que permitem interação com o sistema, com displays ou com a estrutura da morfologia (HSI, 2005). Esse mapa é construído a partir de uma navegação extensiva pela aplicação, abordando todas as possíveis interações disponíveis.

De acordo com (HSI, 2005), devem ser seguidos cinco passos para a construção do mapa morfológico:

1. **Analisar a morfologia:** Procurar e identificar sistematicamente o maior número de elementos estruturais na morfologia;

2. **Identificar os elementos morfológicos:** Para cada componente identificado, determinar em qual categoria este pertence: *container*, *interactor*, *display* ou objeto interativo;
3. **Modelar os elementos em um mapa:** Inserir os símbolos e nomes apropriados para o objeto no mapa;
4. **Modelar as conexões:** Representar o relacionamento entre o *container* superior e o *container* inferior (graficamente representado por uma seta) para indicar a relação de “posse” entre estes dois elementos;
5. **Modelar as navegações:** Usar cada *interactor* ou objeto interativo, quando apropriado, para determinar as mudanças de comportamento no estado da morfologia (por exemplo, se pressionar um botão aparece uma caixa de diálogo ou muda algum *display*?). Se sim, representar o relacionamento entre o *interactor* e o *container* modificado (graficamente representado por uma seta) para indicar o comportamento da navegação.

A Figura 15 apresenta o mapa morfológico da ferramenta Microsoft Project criada seguindo os passos acima descritos. Como a figura é muito grande e o mapa inteiro fica inviável para visualização, em seguida o mapa será dividido em algumas partes para facilitar a visualização. As Figuras 16 a 22 apresentam as partes do Mapa Morfológico da ferramenta.

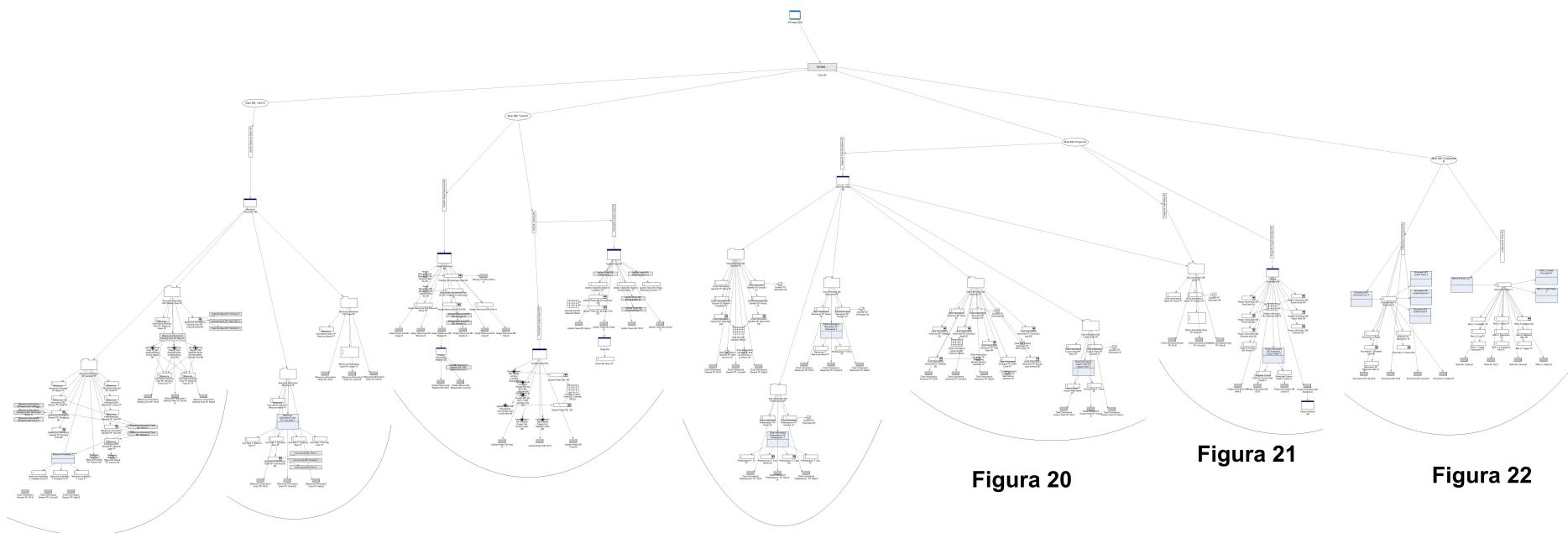


Figura 15 - Mapa Morfológico completo da ferramenta Microsoft Project 2003

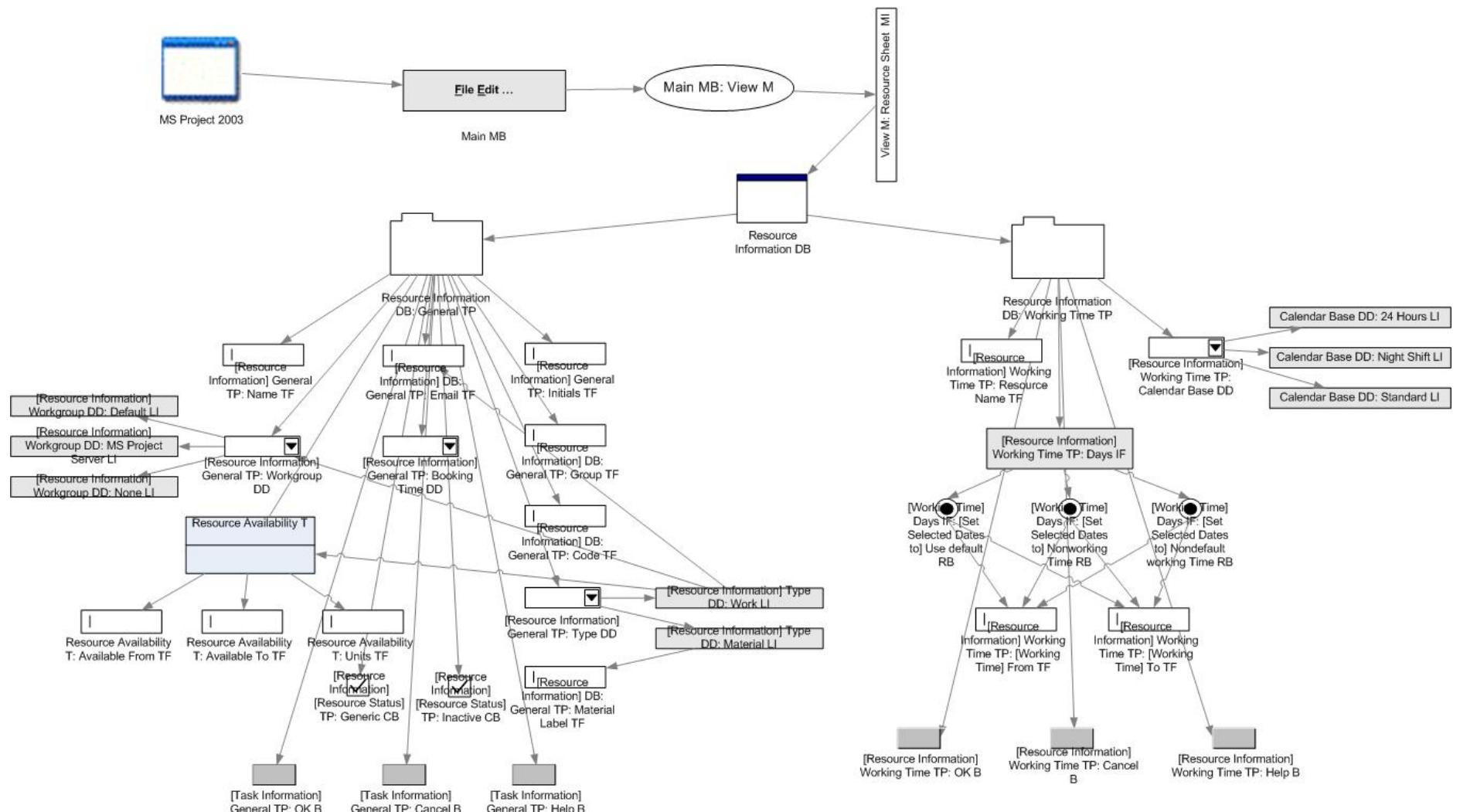


Figura 16 – Mapa Morfológico do Microsoft Project - Parte 1 de 7

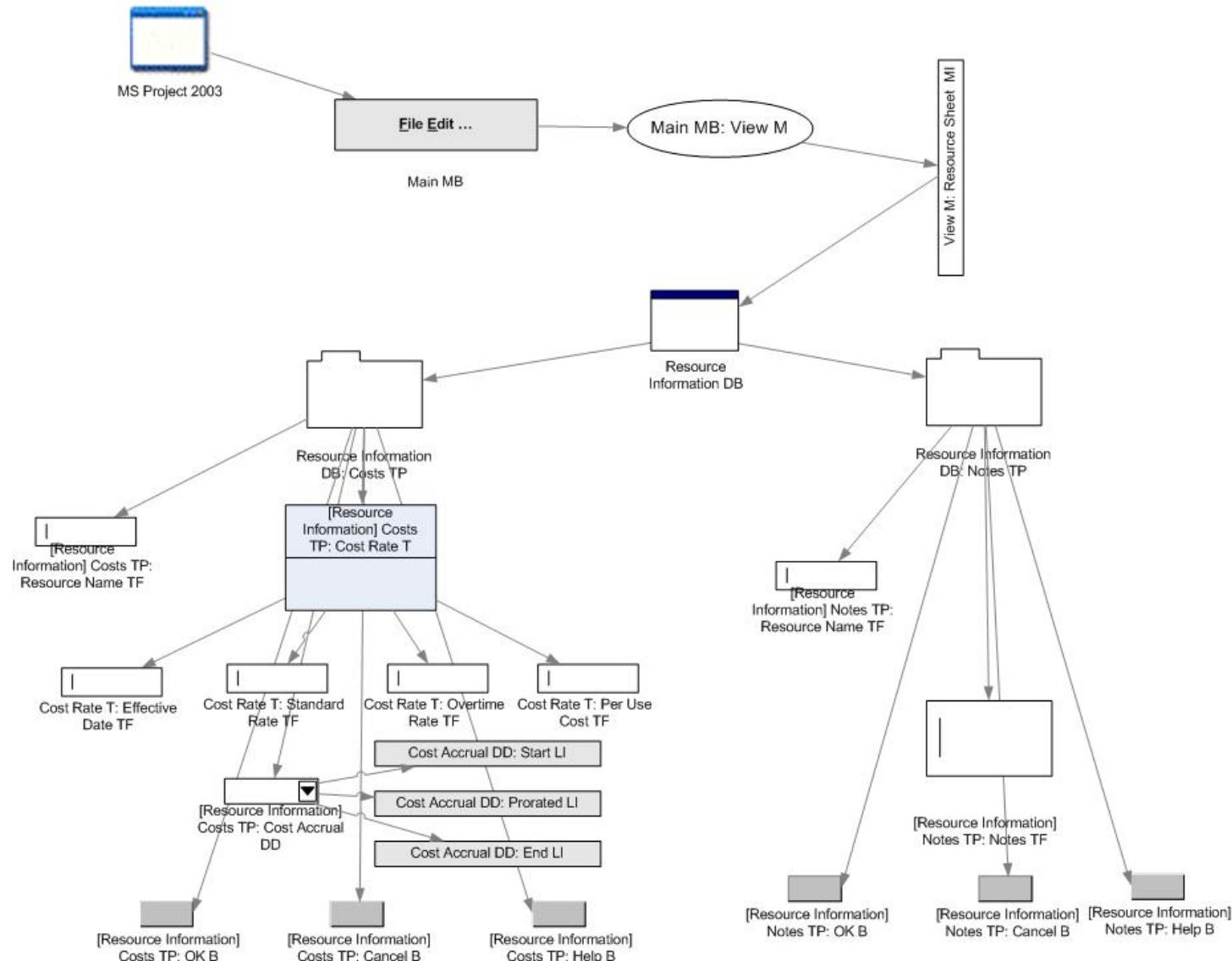


Figura 17 – Mapa Morfológico do Microsoft Project - Parte 2 de 7

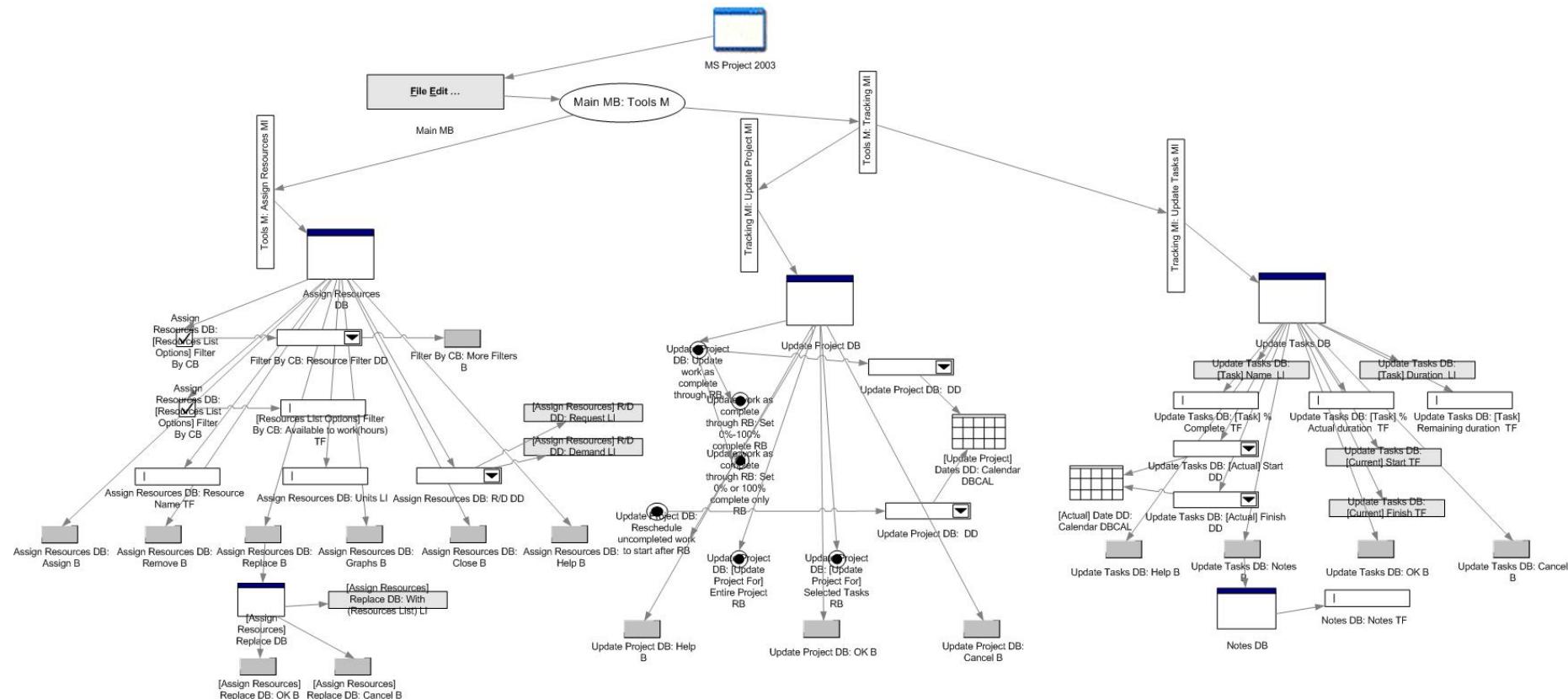


Figura 18 – Mapa Morfológico do Microsoft Project - Parte 3 de 7

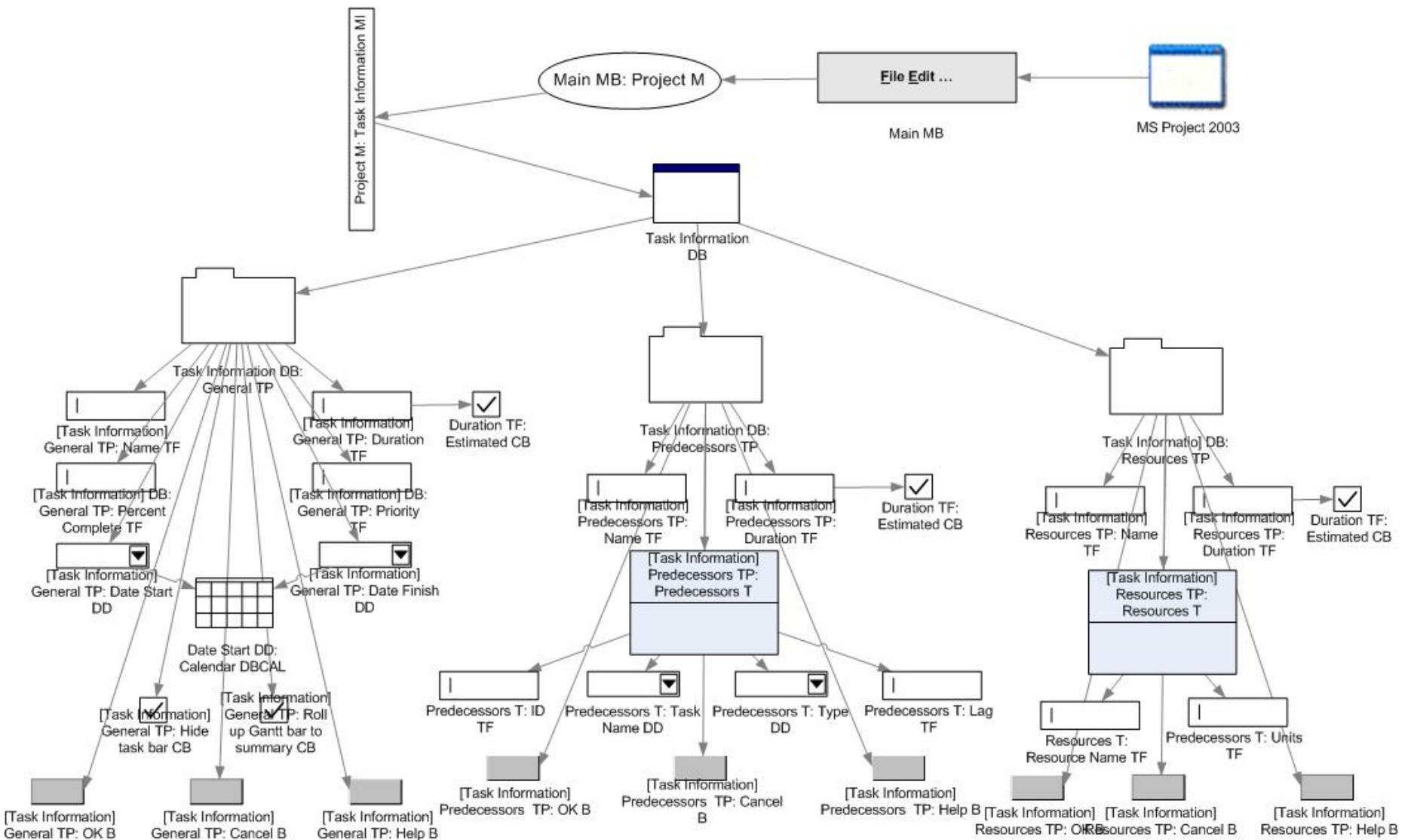


Figura 19 – Mapa Morfológico do Microsoft Project - Parte 4 de 7

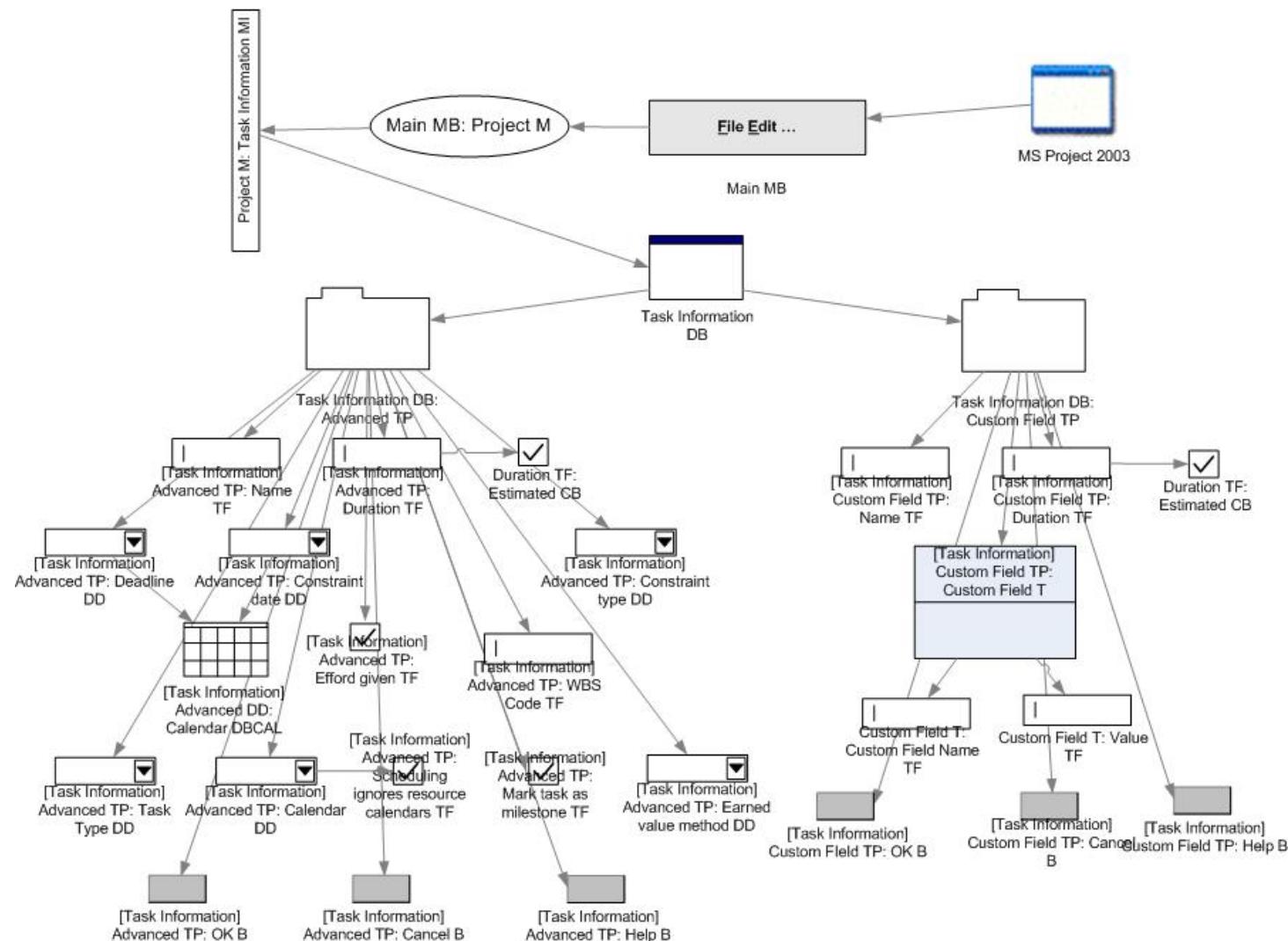


Figura 20 – Mapa Morfológico do Microsoft Project - Parte 5 de 7

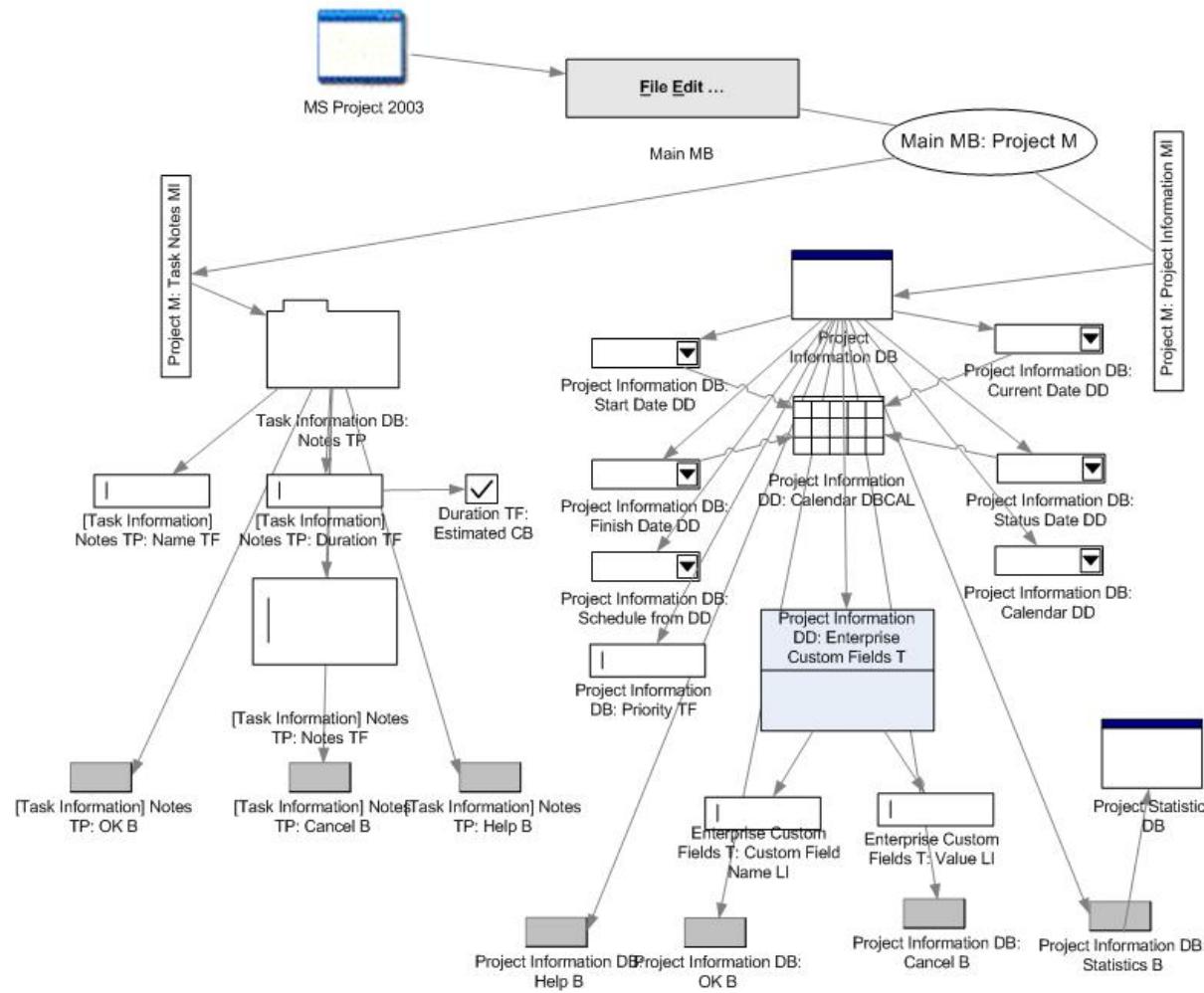


Figura 21 – Mapa Morfológico do Microsoft Project - Parte 6 de 7

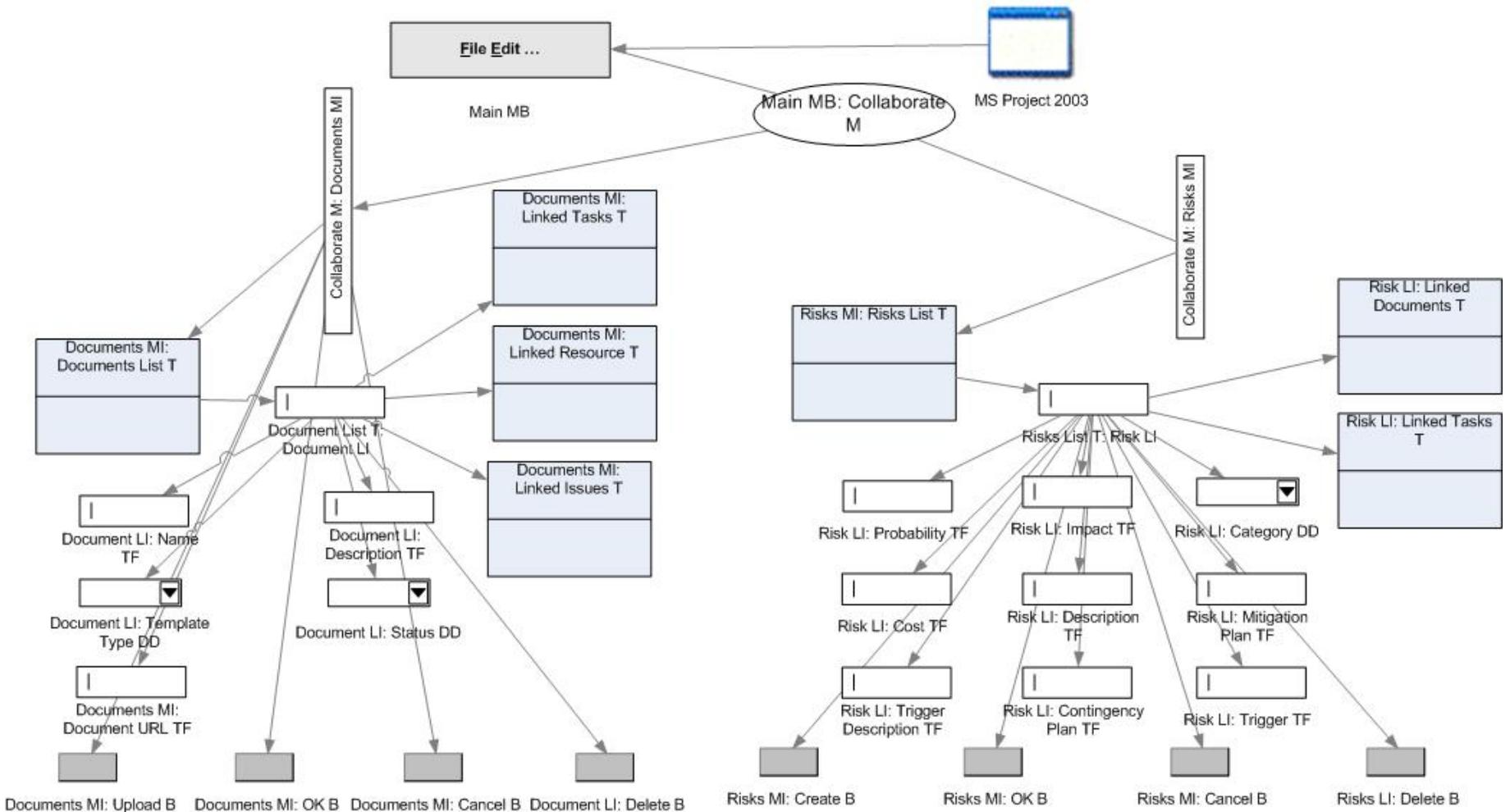


Figura 22 – Mapa Morfológico do Microsoft Project - Parte 7 de 7

A Figura 19 apresenta uma parte do Mapa Morfológico relacionado às informações referentes a atividades de um projeto. Seguindo o caminho de menus Project (*Project M*) → Task Information (*Task Information MI*), é aberta uma caixa de diálogo (*Task Information DB*) que contém as informações de uma atividade divididas em abas (*General TB*, *Predecessors TB* e *Resources TB*). A aba “General TP” contém as informações mais gerais sobre uma atividade, como nome (*Name TF*), Duração (*Duration TF*), Prioridade (*Priority TF*), Percentagem de execução completa (*Percent Complete TF*) e suas datas de início e fim. Para cada uma das datas, um Calendário (*Calendar DBCAL*) é aberto para facilitar a escolha das datas. A aba “Predecessors TP” contém informações sobre as atividades predecessores à atividade em questão. Esta contém uma tabela (*Predecessors T*) contendo uma lista de atividades que são predecessores e algumas informações relativas a essas atividades. A aba “Resources TP” contém os recursos que serão alocados para a execução dessa atividade. Uma tabela com uma lista com os Recursos presentes no projeto (*Resources T*) é apresentada para que um ou mais destes possam ser vinculados a essa atividade.

4.3.2 Modelo Conceitual

Modelado o mapa morfológico, os conceitos relativos ao domínio são extraídos do mapa para que o modelo conceitual seja construído. Para isso, é necessário que o responsável por essa modelagem conheça o domínio para que possa inferir quais são os conceitos a serem extraídos do mapa. Em seguida, os relacionamentos entre os conceitos são modelados, também sob influência do conhecimento do domínio por parte do modelador. Por fim, o modelo conceitual da ferramenta é modelado em forma de uma Rede Semântica composta dos conceitos extraídos e seus relacionamentos. De acordo com (HSI, 2005), os passos básicos para a construção do modelo conceitual são:

1. **Escavar os conceitos:** A partir dos rótulos dos elementos morfológicos encontrados no mapa mais o conhecimento do domínio, o responsável pela modelagem pode identificar os conceitos. Por exemplo, para o software Microsoft Project, se o mapa estiver indicando as informações de uma atividade (*Task Information DB*) e nessas existir um elemento morfológico rotulado como “*Duration TF*”, o responsável pela modelagem, conhecendo o domínio, sabe que existe o conceito de Atividade (*Task*) já que o mapa referencia as informações

de uma atividade. Da mesma forma, o elemento morfológico “Duration TF” indica que existe o conceito de Duração (*Duration*);

2. **Identificar os relacionamentos:** Para cada conceito, identificar os relacionamentos que esse tem com outros conceitos através de observação da interface com o usuário e com interação direta com o sistema. Da mesma forma que o passo anterior, é necessário também um conhecimento do domínio para facilitar a identificação dos relacionamentos entre os conceitos. Segundo o mesmo exemplo dado anteriormente, se no mapa foram identificados os conceitos Atividade e Duração, sabendo que o elemento morfológico que originou o conceito Duração está diretamente relacionado com o elemento que originou o conceito Atividade e sabendo que no domínio de Gerência de Projetos uma Atividade pode possuir uma Duração, é inferido que existe um relacionamento entre esses conceitos.
3. **Montar a Rede Semântica:** Identificados os conceitos e relacionamentos, estes devem ser modelados em uma Rede Semântica de forma a apresentar de forma clara o modelo conceitual.

A Figura 23 apresenta o modelo conceitual da ferramenta Microsoft Project, criada a partir dos passos descritos acima.

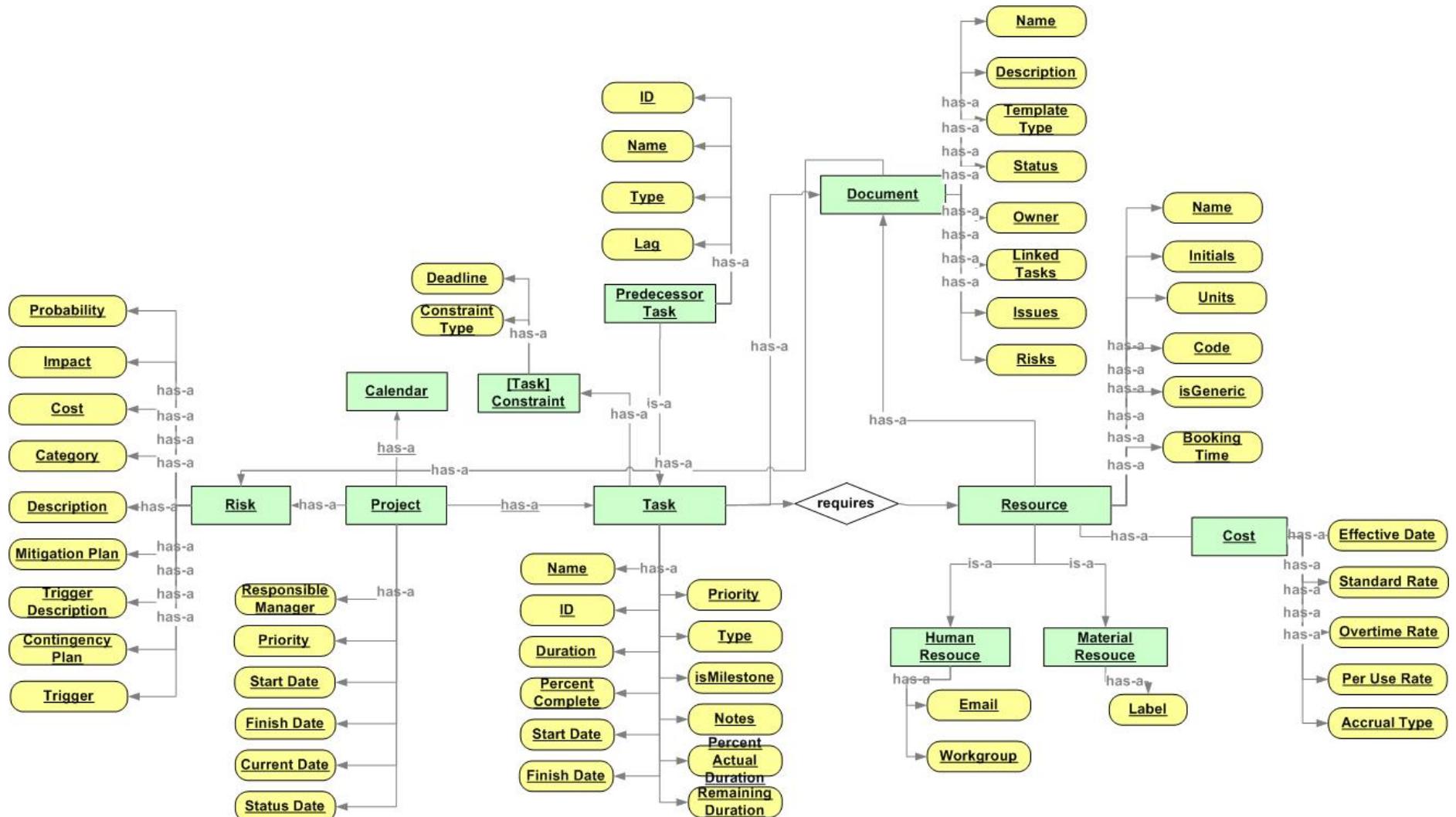


Figura 23 - Modelo Conceitual da ferramenta Microsoft Project 2003

4.4 Ontologia de Processos

Empresas de desenvolvimento de software estão cada vez mais preocupadas em melhorar seus processos a fim de tentar garantir uma melhor qualidade em seus produtos e serviços. Dessa forma, é interessante manter uma base de conhecimentos relativos aos processos usados na organização, e ontologias podem desempenhar um papel muito importante nesse agrupamento de informações.

Foi desenvolvida por (FALBO, 1998) uma ontologia de processo de software, que tinha como objetivo apoiar a aquisição, organização, reuso e compartilhamento de conhecimento sobre processos de desenvolvimento de software (FALBO, 1998). Contudo, com a evolução pela qual tem passado a área de processos de software nos últimos anos, notou-se que era necessário evoluir essa ontologia para que ela tivesse uma maior abrangência, incorporando novos conceitos, relações e restrições (BERTOLLO, 2006). Dessa forma, (BERTOLLO, 2006) apresenta uma evolução dessa ontologia de processos, que será tomada como referência ao domínio de Processos de Software. O método para a construção dessa ontologia está descrito em (FALBO et al., 2002).

A ontologia original (FALBO, 1998) foi construída em partes diferentes. Visto a complexidade desse domínio, ela foi dividida em subdomínios e para cada subdomínio foi construída uma sub-ontologia. Em seguida, essas sub-ontologias foram aplicadas de forma integrada, abrangendo assim grande parte do conhecimento relacionado a processos de software.

A Figura 24 apresenta como a Ontologia de Processo de Software (*Software Process Ontology*) foi dividida em subdomínios. Ela foi dividida nas sub-ontologias de Atividade (*Activity Ontology*), Recurso (*Resource Ontology*) e Procedimento (*Procedure Ontology*). Essa divisão foi feita originalmente em (FALBO, 1998) e mantida em (BERTOLLO, 2006).

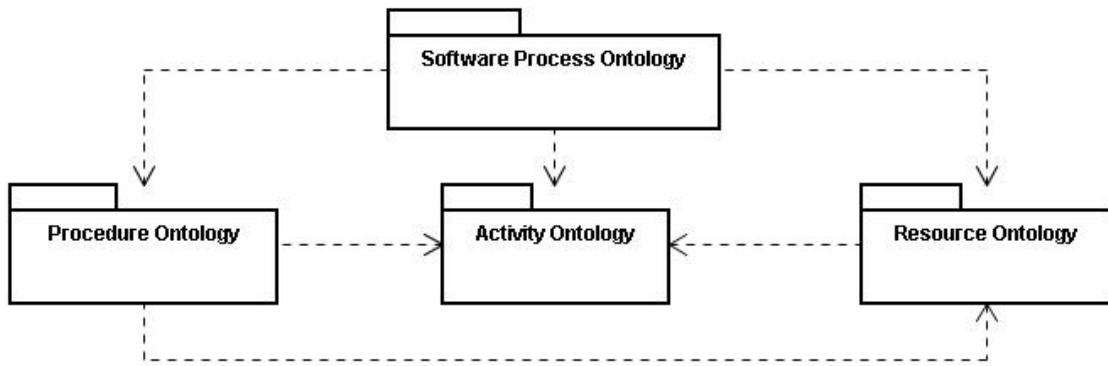


Figura 24 - Ontologia de Processos de Software e suas sub-ontologias

Nesse trabalho não será apresentada toda a ontologia de processos. Serão apresentados apenas os conceitos relevantes ao objetivo desse trabalho.

Nas próximas subseções serão apresentadas sucintamente as partes relevantes das ontologias de Atividade, Recurso, Procedimento e Processo de Software, que são os subdomínios relevantes para a realização desse trabalho.

4.4.1 Sub-ontologia de Atividade

O conceito de atividade está no cerne de qualquer modelo de processo de software. De fato, um processo de desenvolvimento é sempre orientado a atividade e, portanto, a habilidade de representar adequadamente atividades é um aspecto fundamental para uma ontologia de processo de software (FALBO, 1998).

A Figura 25 mostra a sub-ontologia de Atividade. Como mostrado na figura, uma atividade pode depender de outra atividade e ainda ser composta de outras atividades. Essas atividades podem produzir ou consumir artefatos.

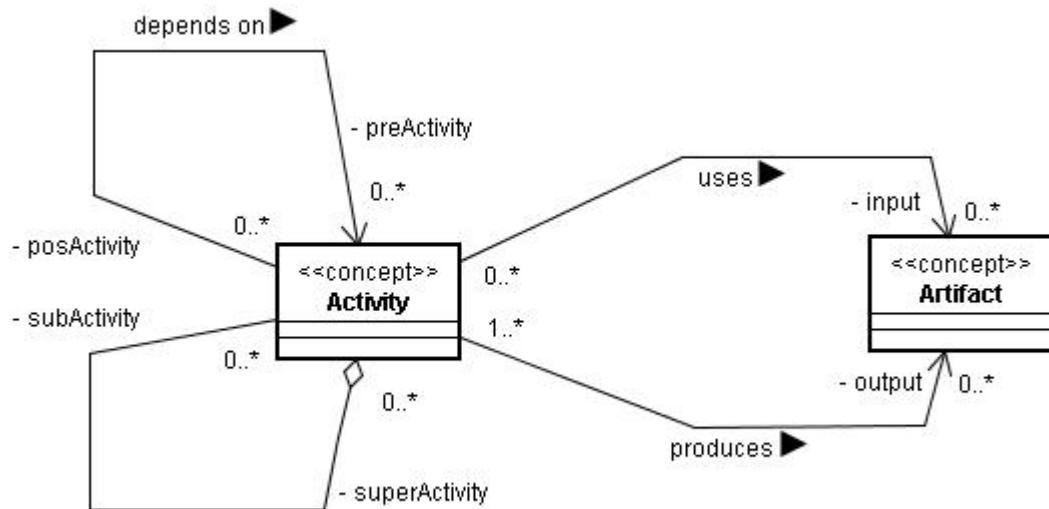


Figura 25 - Sub-ontologia de Atividades

4.4.2 Sub-ontologia de Recurso

Para realizar uma atividade, além dos artefatos de insumo, outros elementos são necessários. Esses elementos, tais como agentes humanos, equipamentos de hardware e ferramentas de software, atuam ou apóiam a realização da atividade. Contudo, eles apenas auxiliam o processo, mas não são incorporados ao produto de software. Tais elementos são considerados recursos para a atividade (BERTOLLO, 2006).

A Figura 26 mostra parte da sub-ontologia de Recurso. Como pode ser visto nesta figura, uma atividade pode requerer um recurso para sua realização. Esse recurso pode ser um Recurso Humano (*Human Resource*), um Recurso de Software (*Software Resource*) ou um Recurso de Hardware (*Hardware Resource*).

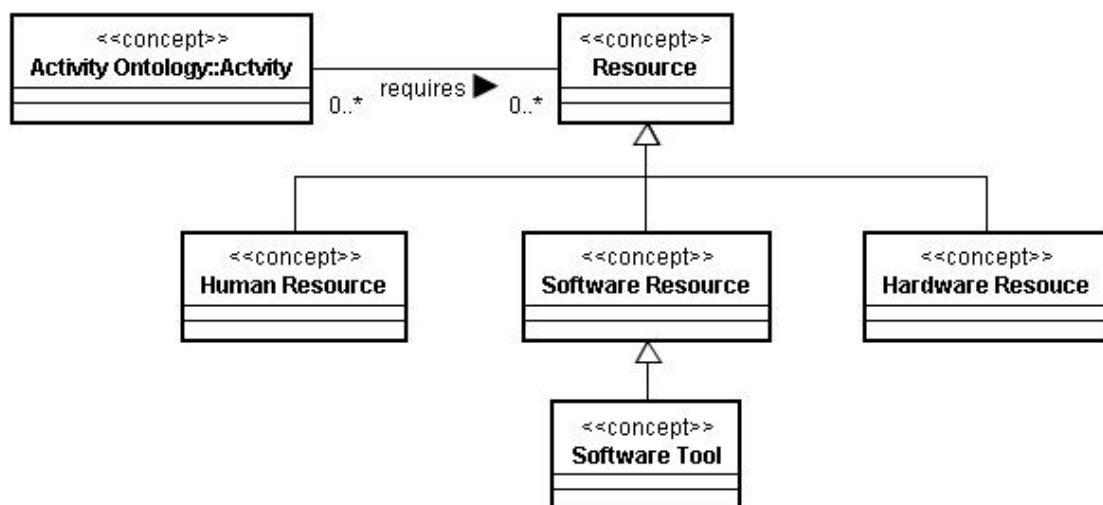


Figura 26 - Parte da sub-ontologia de Recurso

4.4.3 Sub-ontologia de Procedimento

Um procedimento deve ser adotado para controlar a forma como uma atividade deve ser executada. Existem vários tipos de procedimentos para a realização das atividades. Uma Diretriz (*Guideline*), por exemplo, rege os passos a serem seguidos para execução de uma atividade.

A Figura 27 mostra uma parte da ontologia de Procedimento definida em (FALBO, 1998). Como apresentado na figura, uma Atividade (*Activity*) adota um Procedimento (*Procedure*) para definir como essa será realizada. Esse Procedimento pode ser uma Diretriz (*Guideline*). Essa Diretriz pode ser um Roteiro para a execução dessa atividade, podendo esse ser um Modelo de Documento (*Document Model*). Esse Roteiro pode ser manipulado como um Artefato (*Artifact*), sob o formato de um Documento (*Document*).

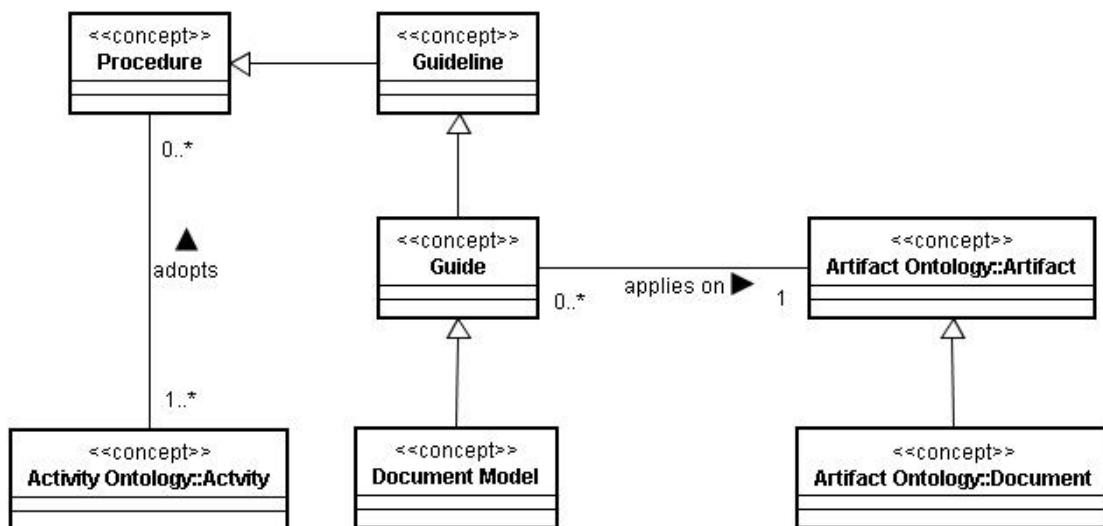


Figura 27 - Parte da sub-ontologia de Procedimento

4.4.4 Ontologia de Processos de Software

Um processo é definido para estabelecer uma abordagem sistemática no desenvolvimento e manutenção de software, e pode ser decomposto em atividades ou outros processos (BERTOLLO, 2006). Da mesma forma, um processo pode interagir com outros processos.

Segundo (BERTOLLO, 2006), processos de software (*Software Processes*), sejam processos padrão (*Standard Process*) ou processos de projeto (*Project Process*) podem ser definidos adaptando-se um processo padrão. Quando um processo padrão é adaptado para outro processo padrão, o processo adaptado é chamado processo especializado e todos os ativos de processo definidos para o processo padrão tornam-se parte do processo especializado. Um processo de projeto é definido para um projeto específico.

A Figura 28 apresenta parte da ontologia de Processos de Software definido em (FALBO, 1998).

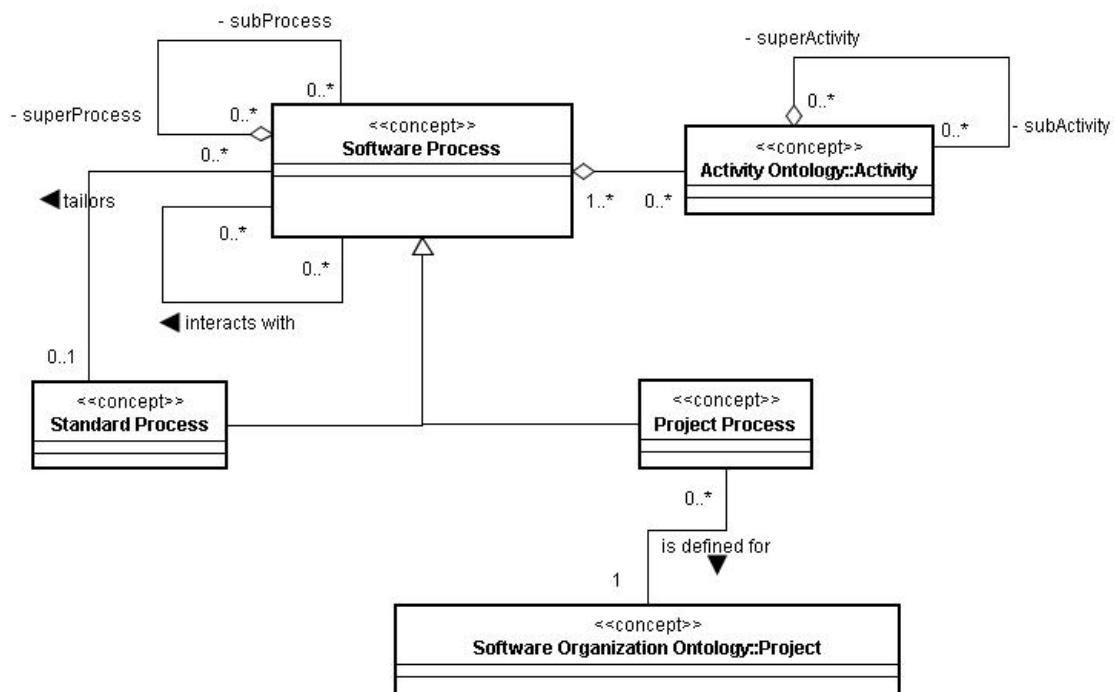


Figura 28 - Parte da ontologia de Processos de Software

A Tabela 1, definida em (FALBO, 1998), apresenta parte do dicionário de conceitos e relacionamentos, mostrando apenas os conceitos abordados nesse trabalho.

Conceito / Relacionamento	Descrição
Activity	Uma Atividade (<i>Activity</i>) é uma ação que transforma artefatos de entrada (insumos) em artefatos de saída (produtos).
adopts	Uma Adoção (relativo a <i>adopts</i>) é uma relação entre um procedimento e uma atividade, indicando que o procedimento pode ser adotado na realização da atividade.
applies on	Uma Aplicação (relativo a <i>applies</i>) é uma relação entre um artefato e

	um roteiro, indicando que um roteiro provê orientações para a elaboração de um artefato.
Artifact	Um Artefato (<i>Artifact</i>) é um insumo para, ou um produto de, uma atividade, no sentido de ser um objeto de transformação da atividade.
depends on	Uma relação de Dependência (relativo à <i>depends on</i>) entre Atividades que indica que uma atividade precisa ser realizada para que outra atividade também possa ser realizada.
Document	Um Documento (<i>Document</i>) é um artefato de software não passível de execução. Ex: documento de especificação de requisitos, plano de projeto, plano de qualidade, relatório de avaliação da qualidade, etc.
Document Model	Um Modelo de Documento (<i>Document Model</i>) é um tipo de roteiro que determina a estrutura de um documento e institucionaliza sua elaboração. Ex: Modelo de documento de Plano de Projeto, Modelo de relatório de Avaliação de Qualidade, etc.
Guide	Um Roteiro (<i>Guide</i>) é uma diretriz para a elaboração de documentos. Ex: roteiro de plano de projeto.
Guideline	Uma Diretriz (<i>Guideline</i>) é um procedimento que visa estabelecer um padrão para a realização de atividades. Diretrizes podem ser divididas em roteiros e normas
Hardware Resource	Um Recurso de Hardware (<i>Hardware Resource</i>) é um equipamento de hardware necessário para a realização de uma atividade. Ex: computadores, kit multimídia, etc.
Human Resource	Um Recurso Humano (<i>Human Resource</i>) é um agente humano necessário para a realização de uma atividade. Ex: engenheiro de software, programador, especialista de domínio, etc.
posActivity	Uma Pós-Atividade (<i>posActivity</i>) é uma faceta da relação de dependência entre duas atividades a_1 e a_2 . Se a realização de a_2 depende da realização de a_1 então a_2 é dita uma pós-atividade de a_1 .
preActivity	Uma Pré-Atividade (<i>preActivity</i>) é uma faceta da relação de dependência entre duas atividades a_1 e a_2 . Se a_1 precisa ser realizada para que a_2 o seja então, a_1 é dita uma pré-atividade para a_2 .
Procedure	Um Procedimento (<i>Procedure</i>) é uma conduta bem estabelecida e ordenada para a realização de uma atividade. Quanto à sua natureza,

	procedimentos podem ser classificados em métodos, técnicas e diretrizes.
Project	Um Projeto (<i>Project</i>) é uma série de atividades relacionadas, que podem produzir ou consumir artefatos, que requerem recursos para serem executados, com o intuito de se produzir um produto ou serviço.
Project Process	Um Processo de Software (<i>Project Process</i>) é um processo definido para um projeto específico, considerando suas particularidades. Pode ou não ser definido a partir de uma adaptação de um processo padrão.
produces	Produz (relativo à <i>produces</i>) é uma relação entre um artefato e uma atividade, indicando que o artefato é produzido pela atividade.
requires	Requer (relativo à <i>requires</i>) é uma relação entre um recurso e uma atividade, indicando que o recurso é necessário para a realização da atividade.
Resource	Um Recurso (<i>Resource</i>) é qualquer coisa que seja necessária para a realização de uma atividade, mas que não seja um insumo para a atividade, no sentido de não ser objeto de transformação por parte da atividade. Recursos podem ser classificados em recursos de hardware, recursos de software e recursos humanos.
Software Process	Um Processo de Software (<i>Software Process</i>) é um conjunto de sub-processos ou atividades inter-relacionadas que transforma insumos em produtos. Ex: Processo de Desenvolvimento de Software.
Software Resource	Um Recurso de Software (<i>Software Resource</i>) é um software necessário para a realização de uma atividade, mas que não é incorporado ao produto desta.
Software Tool	Ferramentas de Software (<i>Software Tool</i>) são recursos de software que apóiam apenas partes específicas de um processo, sendo utilizadas, geralmente, para (semi-) automatizar um procedimento.
Standard Process	Um Processo Padrão (<i>Standard Process</i>) é um processo definido em um nível organizacional, contendo os ativos (<i>assets</i>) de processo básicos que devem fazer parte de qualquer processo de projeto de uma organização.
subActivity	Uma sub-Atividade (<i>subActivity</i>) é uma faceta da relação de

	composição entre duas atividades a_1 e a_2 . Se a_2 é parte de a_1 então a_2 é dita uma sub-atividade de a_1 .
superActivity	Uma super-Atividade (<i>superActivity</i>) é uma faceta da relação de composição entre duas atividades a_1 e a_2 . Se a_1 é decomposta em outras atividades, dentre elas a_2 , então a_1 é dita uma super-atividade de a_2 .
Uses	Usar (relativo à <i>uses</i>) é uma relação entre um artefato e uma atividade, indicando que o artefato é utilizado como “matéria-prima” pela atividade, sendo de alguma forma incorporado a outro artefato, o produto da atividade.

Tabela 1 – Conceitos e Relacionamentos da ontologia de Processo de Software

Os demais conceitos definidos na ontologia de processo de software definidos em (FALBO, 1998) e (BERTOLLO, 2006), como por exemplo, *Software Kind* (Tipo de Software), *Process Category* (Categoria de Processo), *Paradigm* (Paradigma), etc., não foram abordados, pois não são relevantes para o desenvolvimento desse trabalho.

4.5 Ontologia de Riscos

O risco de um projeto é um evento ou condição incerta que, se ocorrer, terá um efeito positivo ou negativo sobre pelo menos um objetivo do projeto, como tempo, custo, escopo ou qualidade (ou seja, em que o objetivo de tempo do projeto é a entrega de acordo com o cronograma acordado; em que o objetivo de custo do projeto é a entrega de acordo com o custo acordado, etc.). Um risco pode ter uma ou mais causas e, se ocorrer, um ou mais impactos (PMI, 2004).

Empresas de todos os seguimentos de mercado devem lidar diariamente com riscos relativos a todos os seus setores. Gerenciamento de Riscos é uma atividade complexa que demanda recursos altamente qualificados e experientes que normalmente não estão disponíveis nas empresas. Esse controle depende de um amplo conhecimento de todos os setores da empresa, visto que os riscos podem estar associados a todas as seções da organização.

Com o intuito de estabelecer um nivelamento de conhecimento sobre esse domínio, foi desenvolvida em (FALBO et al., 2004) uma ontologia de riscos de software. O método para a construção dessa ontologia está descrito em (FALBO et al., 2002). A Figura 29 apresenta os conceitos e relacionamentos da ontologia de risco de software.

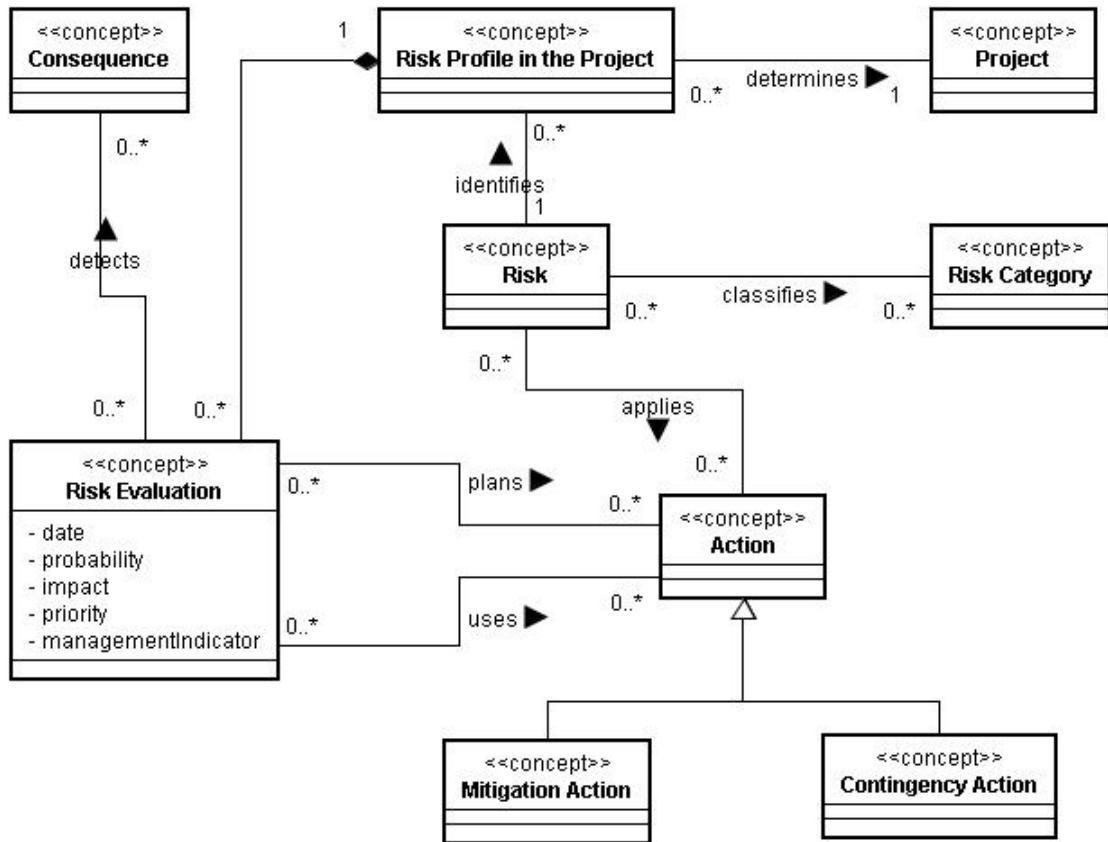


Figura 29 - Ontologia de Riscos de Software

Segundo (FALBO et al., 2004), um risco é qualquer condição, evento ou problema cuja ocorrência não é certa, mas que pode afetar o projeto negativamente, caso ocorra. Um risco pode ser classificado em várias categorias, como riscos associados com o tamanho do produto, riscos associados com a tecnologia a ser usada para construir o produto, riscos associados com as pessoas que trabalharão no projeto, etc. Durante o planejamento do projeto do software, riscos que podem afetar o projeto devem ser identificados e avaliados com o objetivo de priorizar a definir quais riscos serão tratados e como esses serão tratados. Cada risco associado com o projeto define um perfil de risco no projeto. Cada perfil de risco é composto por várias avaliações de risco. A primeira avaliação de risco é feita durante o planejamento do projeto. Outras são feitas em cada um dos marcos (*milestones*) do projeto, quando os riscos do projeto devem ser reavaliados. Uma avaliação de risco determina o estado de um risco em um dado momento do projeto, isto é, é registrado quando a avaliação foi feita, a probabilidade de ocorrência de um risco e seu impacto caso o risco ocorra no curso do projeto, sua prioridade e se o risco deve ser gerenciado. Essa avaliação ajuda na detecção das consequências oriundas da ocorrência de um risco.

Para cada risco, ações podem ser tomadas para tratá-los. Essas ações estão divididas entre Ações de Mitigação, que são ações para tratar a ocorrência de um risco e Ações de Contingência, que são ações para prevenir riscos. Durante a avaliação de um risco, essas ações podem ser planejadas ou usadas para controlar um risco.

A Tabela 2 apresenta um dicionário de conceitos e relacionamentos presentes na ontologia.

Action	Uma Ação (<i>Action</i>) é um procedimento a ser tomado para o gerenciamento de um risco.
applies	Relação entre um Risco e uma Ação, indicando que ações devem ser aplicadas (relativo à <i>applies</i>) para que o risco seja controlado.
classifies	Relação entre um Risco e uma Categoria de Risco, indicando que os riscos podem ser classificados (relativo à <i>classifies</i>) em categorias. Ex: riscos relacionados ao produto, a pessoas, a tecnologia, etc.
Consequence	Uma Conseqüência (<i>Consequence</i>) é um resultado obtido a partir da ocorrência de um risco.
Contingency Plan	Um Plano de Contingência (<i>Contingency Plan</i>) é um plano a ser seguido para evitar que um risco ocorra.
detects	Relação entre uma Avaliação de Risco e uma Conseqüência, indicando que uma avaliação detecta (<i>detects</i>) uma conseqüência caso o risco ocorra.
determines	Relação entre um Perfil de Risco e um Projeto, indicando que um projeto determina (<i>determines</i>) quais os perfis de risco que ocorrem especificamente no projeto.
identifies	Relação entre um Risco e um Perfil de Risco, indicando que um Risco pode identificar (relativo à <i>identifies</i>) vários perfis de riscos em um projeto.
Mitigation Plan	Um Plano de Mitigação (<i>Mitigation Plan</i>) é um plano a ser seguido caso um risco ocorra.
plans	Relação entre uma Avaliação de Risco e uma Ação, indicando que uma Avaliação de Risco pode planejar (relativo à <i>plans</i>) que uma Ação seja realizada, seja ela uma ação de mitigação ou uma ação de contingência.

Project	Um Projeto (<i>Project</i>) é um conjunto de atividades a serem desenvolvidas para atingir um determinado objetivo.
Risk	Um Risco (<i>Risk</i>) é qualquer condição, evento ou problema cuja ocorrência não é certa, mas que pode afetar o projeto negativamente caso ocorra.
Risk Category	Uma Categoria de um Risco (<i>Risk Category</i>) é como um risco está classificado. Um risco pode estar relacionado com o tamanho do produto, com a tecnologia envolvida para a construção do produto, com as pessoas envolvidas no desenvolvimento, etc.
Risk Evaluation	Uma Avaliação de Risco (<i>Risk Evaluation</i>) está relacionada a um projeto em um período pré-determinado. Essa avaliação determina o estado do risco em determinado momento.
Risk Profile in the Project	Um Perfil de Risco (<i>Risk Profile</i>) indica qual o tipo do risco identificado no Projeto (<i>Project</i>).
Date	Data (<i>Date</i>) em que foi feita a avaliação do Risco.
Probablility	Probabilidade (<i>Probability</i>) existente da ocorrência de um Risco.
Impact	Impacto (<i>Impact</i>) resultante da ocorrência de um Risco.
Priority	Caso haja mais algum Risco a ser avaliado, são definidas Prioridades (<i>Priority</i>) para determinar a ordem de verificação dos riscos.
managementIndicator	Indicador (<i>managementIndicator</i>) que determinará se o risco será gerenciado.
Uses	Relação entre uma Avaliação de Risco e uma Ação, indicando que uma Avaliação de Risco pode usar (relativo à <i>uses</i>) uma Ação, seja ela uma ação de mitigação ou uma ação de contingência.

Tabela 2 – Conceitos e Relacionamentos da ontologia de Riscos de Software

4.6 Mapeamento entre um modelo conceitual e uma ontologia

Modelos conceituais são usados para representar, de forma restrita, porções da realidade de uma forma estruturada. Assim como apresentado no Capítulo 3, os modelos conceituais abordados nesse trabalho são descritos através de uma rede semântica, não oferecendo nenhuma informação explícita além dos conceitos e seus relacionamentos,

deixando-os com um poder de expressividade muito baixo. O modelo conceitual neste caso é um modelo dependente da aplicação e funciona como um meta-modelo da ferramenta. A ontologia, por outro lado, é a representação do domínio de conhecimento e por isso pode ser usada como domínio semântico. Dessa forma, ao extrair modelos conceituais de determinada esfera de informação, os conceitos apresentados nessas representações precisam ser assinalados semanticamente.

Nesse ponto ontologias desempenham um papel fundamental no enriquecimento semântico de um modelo conceitual. Se as entidades do modelo conceitual estiverem referenciando basicamente os mesmos conceitos das entidades de uma ontologia, esse modelo conceitual poderá usar a ontologia de referência como base semântica, enriquecendo assim o seu poder de expressividade.

Uma das formas de fazer esse relacionamento entre um modelo conceitual e uma ontologia é através de um mapeamento direto. Esse mapeamento consiste na tentativa de arrolar as entidades de um modelo aos conceitos que desempenham o mesmo papel semântico na ontologia de referência.

Nesse trabalho, o mapeamento será feito de forma (1) *gráfica*, onde os modelos serão mapeados com setas direcionais partindo do modelo conceitual até sua respectiva ontologia de referência e (2) *tabular*, onde serão comparados os conceitos e relacionamentos do modelo conceitual com seus respectivos conceitos e relacionamentos na ontologia de referência.

Na próxima subseção será apresentado o mapeamento entre um modelo conceitual extraído da ferramenta Microsoft Project e duas ontologias de referência, a saber, a Ontologia de Processos de Software e a Ontologia de Riscos de Software.

4.6.1 Mapeamento entre a Ontologia de Processos de Software, Ontologia de Riscos e o Modelo Conceitual do Microsoft Project

Conforme explicado no Capítulo 3, uma premissa adotada neste trabalho é a de que se pode avaliar se uma ferramenta esta apta a apoiar um processo de desenvolvimento avaliando o quanto esta ferramenta está baseada em uma ontologia de referência. Para isso, é necessário verificar se os conceitos manipulados pela ferramenta estão de acordo com os propostos pela ontologia. Essa verificação pode ser feita através de um mapeamento conceitual entre esses dois artefatos.

Durante a escavação do modelo conceitual do Microsoft Project, foi verificado que esse aborda conceitos relativos a mais de um domínio. Além de manipular conceitos relativos

a processos de software, esse aborda conceitos do domínio de riscos, de colaboração, de eventos, etc. Dessa forma, para enriquecer o estudo de caso, a ferramenta será avaliada se é apta para apoiar os domínios relativos a processos de software e a riscos. Para isso, será feito o mapeamento entre o modelo conceitual e as ontologias de processo e de risco, já apresentadas nesse capítulo. A Figura 30 apresenta o mapeamento gráfico entre as ontologias de referência de o modelo conceitual escavado.

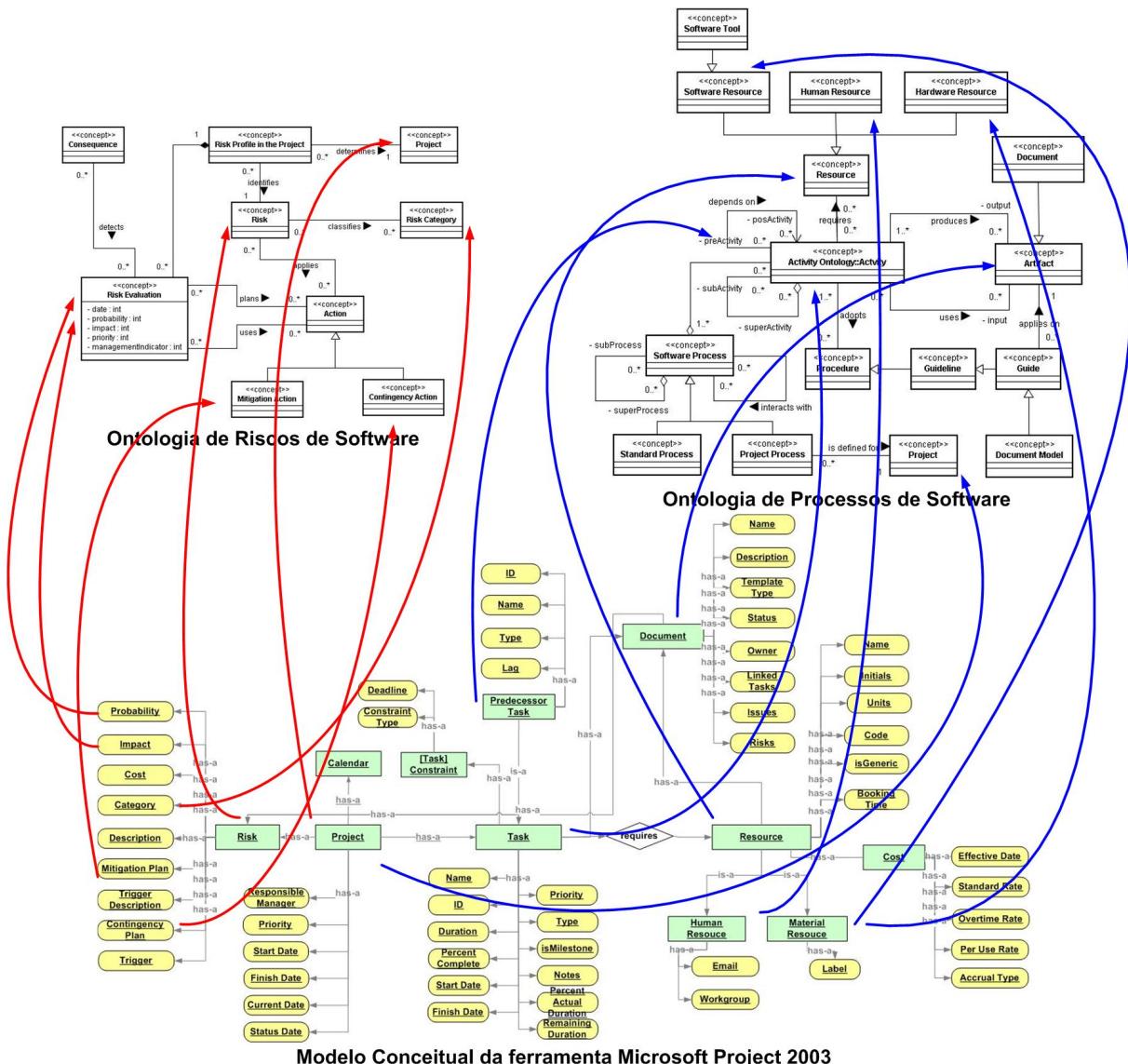


Figura 30 - Mapeamento entre as ontologias de Riscos e Processos de Software com o Modelo Conceitual do Microsoft Project

A Tabela 3 apresenta o mapeamento conceitual entre o Modelo Conceitual e a Ontologia de Riscos de Software.

Ontologia de Riscos	Modelo Conceitual	Relação
Risk	Risk	Ambos os conceitos tratam os Riscos

		relacionados a um projeto
Project	Project	Ambos os conceitos tratam os Projetos de uma organização
Risk Category	Category	Ambos os conceitos tratam as Categorias que os Riscos estão classificados
Mitigation Action	Mitigation Plan	Ambos os conceitos tratam das formas de lidar com riscos que ocorreram
Contingency Action	Contingency Plan	Ambos os conceitos tratam das formas de prevenir que possíveis riscos ocorram
Probability	Probability	Ambos os conceitos tratam da probabilidade de ocorrência de um risco
Impact	Impact	Ambos os conceitos tratam do impacto no projeto caso um risco ocorra

Tabela 3 – Mapeamento entre conceitos do Modelo Conceitual e Ontologia de Riscos de Software

A Tabela 4 apresenta o mapeamento conceitual entre o Modelo Conceitual e a Ontologia de Processos de Software.

Ontologia de Processos	Modelo Conceitual	Relação
Activity	Task	Ambos os conceitos tratam das Atividades existentes em um projeto
preActivity	Predecessor Task	A ontologia descreve o relacionamento de dependência entre atividades e o modelo conceitual explicita o conceito de uma atividade que é cronologicamente anterior a uma determinada atividade
Project	Project	Ambos os conceitos tratam do projeto específico da organização
Resource	Resource	Ambos os conceitos tratam dos recursos necessários para a execução de uma atividade
Human Resource	Human Resource	Ambos os conceitos tratam dos recursos humanos para a execução de uma atividade
Software Resource	Material	A ontologia descreve uma taxonomia mais

	Resource	detalhada para Recursos. Tanto o conceito de Recurso de Hardware quanto Recurso de Software são tratados como Recursos Materiais no modelo conceitual
Hardware Resource	Material Resource	A ontologia descreve uma taxonomia mais detalhada para Recursos. Tanto o conceito de Recurso de Hardware quanto Recurso de Software são tratados como Recursos Materiais no modelo conceitual
Artifact	Document	Ambos os conceitos tratam os artefatos produzidos ou consumidos no processo de desenvolvimento.

Tabela 4 – Mapeamento entre conceitos do Modelo Conceitual e Ontologia de Processos de Software

Analizando o mapeamento apresentado, é observado que o modelo conceitual extraído da ferramenta possui muitos conceitos que podem ser relacionados com as ontologias de referência. Alguns conceitos, porém, não puderam ser mapeados nas ontologias de referências usadas nesse trabalho. Isso significa que esta ferramenta manipula conceitos referentes a outros domínios de aplicação, não somente Processos e Riscos de Software.

Fazendo uma análise mais profunda, a partir de intensa manipulação do software, verificou-se que os conceitos, além de sintaticamente, são também semanticamente idênticos, pois são manipulados com o mesmo propósito, com os mesmos relacionamentos entre si. Isso indica que a ferramenta manipula os conceitos de forma consistente, no que diz respeito ao domínio que esta foi proposta a abordar. Dessa forma, pode-se concluir que esta ferramenta está apta a apoiar um processo de desenvolvimento relativo a atividades envolvidas no domínio de Processos e Riscos de Software visto que pode ser feito um mapeamento entre seu modelo conceitual e as ontologias referentes a esses dois domínios. Contudo, nada pode se afirmar no que diz respeito a essa ferramenta apoiar atividades relativas a outros domínios de aplicação.

Outro resultado importante é mostrar que uma ferramenta pode abranger mais de um domínio específico de aplicação. Nesse estudo de caso apresentado, foi verificado que uma ferramenta específica de gerenciamento de projetos pode abordar conceitos relacionados a processos de software e riscos de software. Para essa ferramenta específica poderiam ser apresentados outros mapeamentos em outros subdomínios relacionados, visto que ela abrange conceitos relativos ao domínio de Colaboração (Notas de atividades, por exemplo) e

Tempo/Eventos (Deadline de atividades e Calendário, por exemplo), mas isto foge ao escopo proposto por esse trabalho.

Capítulo 5

Exemplo de integração de ferramentas através de integração dos modelos conceituais

Durante o desenvolvimento das atividades de um projeto, muitos artefatos podem ser criados ou consumidos. Dessa forma é essencial que um gerenciamento desses artefatos seja feito de forma eficaz. As ferramentas de gerenciamento de projetos disponíveis no mercado atualmente não fazem um controle muito eficiente de seus artefatos. Elas simplesmente guardam uma cópia e oferecem a possibilidade de vinculá-las a uma atividade e/ou recurso. Dessa forma, um controle mais consistente deve ser feito e para isso podem ser usadas ferramentas específicas para esse fim, que fazem esse controle de forma automática. Existe um número considerável de ferramentas que provém o controle de artefatos de uma maneira geral. Dentre elas, podemos citar o *Subversion*² e o *Concurrent Versions System (CVS)*³.

Um problema em usar ferramentas independentes é que essas não se comunicam diretamente, necessitando ainda de certo esforço humano para relacionar os dados das ferramentas.

Esse trabalho pretende demonstrar que ferramentas que abrangem diferentes domínios em um processo de desenvolvimento de software podem ser integradas se estas estão baseadas em um modelo ontológico bem fundamentado. Para isso, são usadas ontologias de referência como base conceitual para verificar se a ferramenta trata os conceitos de forma consistente. Conforme apresentado no capítulo anterior, uma forma de verificar se uma ferramenta manipula seus conceitos de forma coerente é tentar fazer um mapeamento entre o modelo conceitual da ferramenta com uma ontologia de referência de mesmo domínio.

Após o processo de fundamentação ontológica dos modelos conceituais, é necessário verificar se as ferramentas possuem conexões semânticas, ou seja, se existem conceitos presentes nos modelos subjacentes a essas ferramentas que são aplicados com o mesmo propósito e que possam ser mapeados. O mapeamento entre os modelos conceituais subjacentes às ferramentas a serem integradas indica onde são os possíveis pontos de

² <http://subversion.tigris.org/>

³ <http://savannah.nongnu.org/projects/cvs/>

integração entre as ferramentas. Assim, com uma abordagem sistemática, as ferramentas podem ser integradas de forma a garantir a interoperabilidade semântica entre elas a partir dessas interfaces de integração.

Nesse trabalho será apresentada a integração semântica entre os domínios de Gerência de Projetos de Gerência e Configuração de Software. As ferramentas exploradas são o Microsoft Project 2003 para o domínio de gerência de projetos e a ferramenta de gerência de configuração CVS.

O modelo conceitual da ferramenta Microsoft Project 2003 e a Ontologia de Processos de Software foram apresentados nos capítulos anteriores. O modelo conceitual da ferramenta CVS foi desenvolvido em (FELHBERG *et al.*, 2007) e a Ontologia de Gerência de Configuração de Software foi desenvolvida em (NUNES, 2005) e atualizada em (ARANTES *et al.*, 2007).

Inicialmente serão apresentados brevemente a Ontologia de Gerência de Configuração e o Modelo Conceitual da ferramenta CVS. Em seguida serão apresentados os mapeamentos entre (1) do modelo conceitual do CVS e a ontologia de Gerência de Configuração, (2) as ontologias de Processos de Software e Gerência de Configuração e (3) os modelos conceituais das ferramentas Microsoft Project e CVS. O mapeamento entre a Ontologia de Processos e o modelo conceitual do Microsoft Project já foi apresentado no capítulo anterior.

5.1 Ontologia de Gerência de Configuração de Software

Durante o processo de desenvolvimento de um software, vários artefatos são produzidos e alterados constantemente. Ferramentas de software, tais como compiladores e editores de texto, também podem ser substituídas por versões mais recentes de seus fabricantes ou mesmo por outras. Assim, para que não haja inconsistência nos artefatos e ferramentas utilizados, é de suma importância o acompanhamento e controle de tais itens, através de um processo de gerência de configuração de software, durante todo o ciclo de vida do software (SANCHES, 2001) (NUNES, 2005).

A Gerência de Configuração de Software (GCS) compreende o conjunto de atividades para controlar modificações, identificando os produtos de trabalho que podem ser modificados, estabelecendo as relações entre eles e os mecanismos para administrar suas diferentes versões ou variantes, controlando as modificações, fazendo auditoria e preparando relatórios sobre tais modificações (PRESSMAN, 2001).

Foi desenvolvida em (NUNES, 2005) e atualizada em (ARANTES et al., 2007) uma ontologia do domínio de GCS com o objetivo de harmonizar os conceitos relacionados a esse domínio, visto que esse é tratado de forma diferente por algumas ferramentas.

A Figura 31 apresenta a ontologia de gerência de configuração atualizada em (ARANTES et al., 2007).

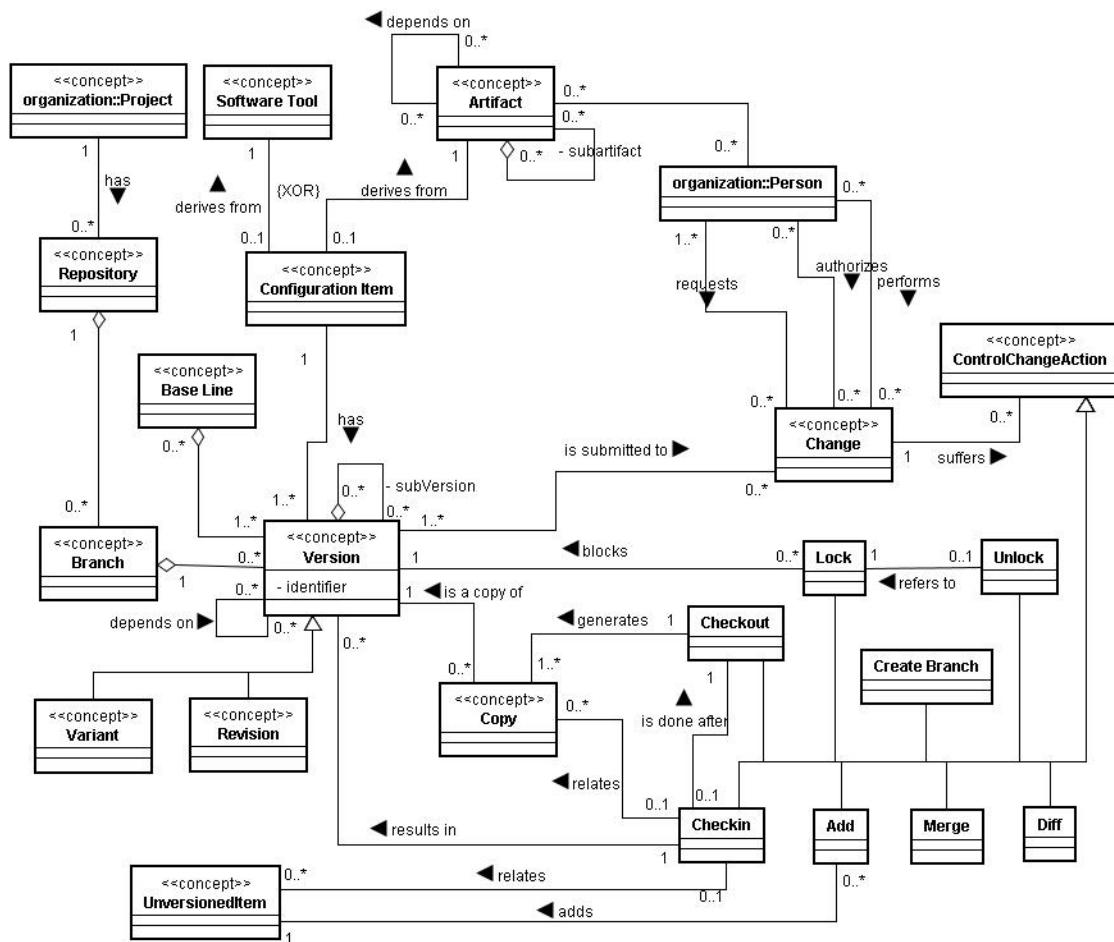


Figura 31 – Ontologia de Gerência de Configuração de Software

A Tabela 5 apresenta um dicionário de dados com os conceitos e relacionamentos apresentados na ontologia de GCS.

Add	Ação de GCS que cria a primeira versão de um item sem controle de versão, e para que essa ação tenha efeito, deve ser seguido da ação Checkin.
authorizes	Relação entre uma Person (pessoa) e uma Change (mudança), onde uma pessoa pode autorizar ou não uma determinada

	mudança.
Base Line	Conjunto de itens de configuração em determinadas variações.
Branch	Representa a linha de desenvolvimento, contemplando um conjunto de variações de itens de configuração. Um <i>Branch</i> (ramo) só pode corresponder a um repositório.
Change	Representa uma requisição de mudança executada por uma pessoa indicando quais variações de itens de configuração podem ser alteradas. Quando essa mudança é aprovada, esta pode ser executada pelas mesmas pessoas. Essa pessoa deve executar ações de GCS de acordo com sua requisição de mudança.
Checkin	Ação de GCS que retorna para o repositório cópias das variações geradas após uma ação prévia de <i>Checkout</i> . Novas variações são criadas a partir de ações de <i>Checkin</i> . Essas novas variações são relativas às cópias modificadas que sofreram ações de <i>Checkin</i> previamente ou a partir itens não versionados adicionados a partir de ações de adição.
Checkout	Ação de GCS que objetivam criar cópias de variações a serem submetidas a mudanças. Uma aprovação de mudança deve ser efetuada para que ações de <i>Checkout</i> possam ser executadas.
Configuration Item	Item que está sob gerencia de configuração. Este pode possuir muitas variações e pode ser um artefato ou uma ferramenta de software.
Copy	Representa a cópia de uma variação. Esta cópia é gerada quando uma ação de <i>Checkout</i> é feita. Cópias devem ser relativas apenas às variações submetidas para mudança.
copy of	Relação entre <i>Copy</i> (cópia) e <i>Variation</i> (variação) que indica que uma determinada cópia é a reprodução de determinada variação.
Create Branch	Ação de GCS que cria um novo ramo. Para essa ação ter efeito, uma ação de <i>Checkin</i> deve ser executada previamente. Todas as cópias ou itens não versionados que sofreram ações de <i>Checkin</i> gerarão variações para o novo ramo criado.
depends on	Relação entre duas <i>Variations</i> (variações) que indica que uma variação depende de outra variação. Dessa forma, é possível identificar qual variação será afetada em uma determinada

	mudança.
derives from	Relação entre <i>ConfigurationItem</i> (item de configuração), <i>Artifact</i> (artefato) e <i>SoftwareTool</i> (ferramenta de software) que um determinado artefato ou ferramenta de software está sob gerência de configuração se tornando um item de configuração.
Diff	Ação de GCS que representa as diferenças entre uma cópia e uma variação que foi usada para gerar essa cópia, ou a diferença entre duas variações.
executes	Relação entre <i>Person</i> (pessoa) e <i>Change</i> (mudança), indicando qual pessoa é responsável pela execução de determinada mudança.
generates	Relação entre <i>Checkout</i> e <i>Copy</i> (cópia) indicando quais são as cópias geradas por uma mudança.
Identifier	Propriedade de uma <i>Variation</i> (variação) que serve como identificador único de uma variação.
is submitted	Relação entre <i>Variation</i> (variação) e <i>Change</i> (mudança) indicando quais variações foram submetidas em determinada mudança.
Lock	Ação de GCS que é diretamente relacionado a uma variação. Quando uma variação está bloqueada, nenhuma ação de <i>Checkin</i> pode gerar novas variações que são relacionadas ao mesmo item de configuração e ramo até que a variação seja desbloqueada.
locks	Relação entre um <i>Lock</i> (bloqueio) e uma <i>Variation</i> (variação) indicando que uma dada variação foi bloqueada e que nenhuma ação de <i>Checkin</i> pode ser executada sobre essa variação enquanto essa estiver bloqueada.
Merge	Uma ação de GCS que indica a fusão versões diferentes de um mesmo item de configuração.
Remove	Uma ação de GCS que indica que uma variação em um ramo será desativada.
Repository	Representa a estrutura que mantém todos os ramos com suas variações.
requests	Relação entre <i>Person</i> (pessoa) e <i>Change</i> (mudança) indicando quais pessoas requisitam determinada mudança.

results	Relação entre <i>Checkin</i> e <i>Variation</i> (variação) indicando quais variações foram produzidas a partir de uma ação de checkin.
SCMAction	Ação de GCS que pode ser aplicada a um Item de Configuração.
suffers	Relação entre <i>Change</i> (mudança) e <i>SCMAction</i> (ação de GCS), indicando quais ações foram executadas em determinada mudança.
UnversionedItem	Artefato ou ferramenta de software que não está sob gerência de configuração.
Variant	Estado em que um item de configuração existe simultaneamente em duas ou mais formas diferentes que atendam a requisitos similares. Ex.: Um documento estava na versão 1.0 e após alterações passou para a versão 1.1. Porém, foi criada a variante 2.0 que, apesar de atender aos mesmos requisitos que a versão 1.1, foi construída de forma diferente.
Variation	Indica qual a versão ou variante de um determinado item de configuração. É caracterizada por um número único para cada variação de um mesmo item de configuração.
Version	Estado em que se encontra um item de configuração. Cada alteração realizada em um item de configuração gera uma nova versão deste item. Ex.: Um documento que estava na versão 1.0 e após algumas alterações passou para versão 1.1.

Tabela 5 – Dicionário de conceitos e relacionamentos da ontologia de GCS

5.2 Modelo Conceitual do CVS

O CVS é uma ferramenta de código aberto relativa ao domínio de GCS que permite que o usuário trabalhe com versões diferentes de um mesmo arquivo, controlando todas as versões existentes e um registro de quem as manipulou.

O modelo conceitual do CVS foi extraído usando a técnica *Ontological Excavation* (Escavação de Ontologias) desenvolvida em (HSI, 2005) e apresentada no Capítulo 3 deste trabalho. A escavação do modelo conceitual do CVS é apresentada em (FELHBERG *et al.*, 2007).

A Figura 32 apresenta o modelo conceitual da ferramenta de controle de versão CVS extraído e apresentado em (FELHBERG *et al.*, 2007).

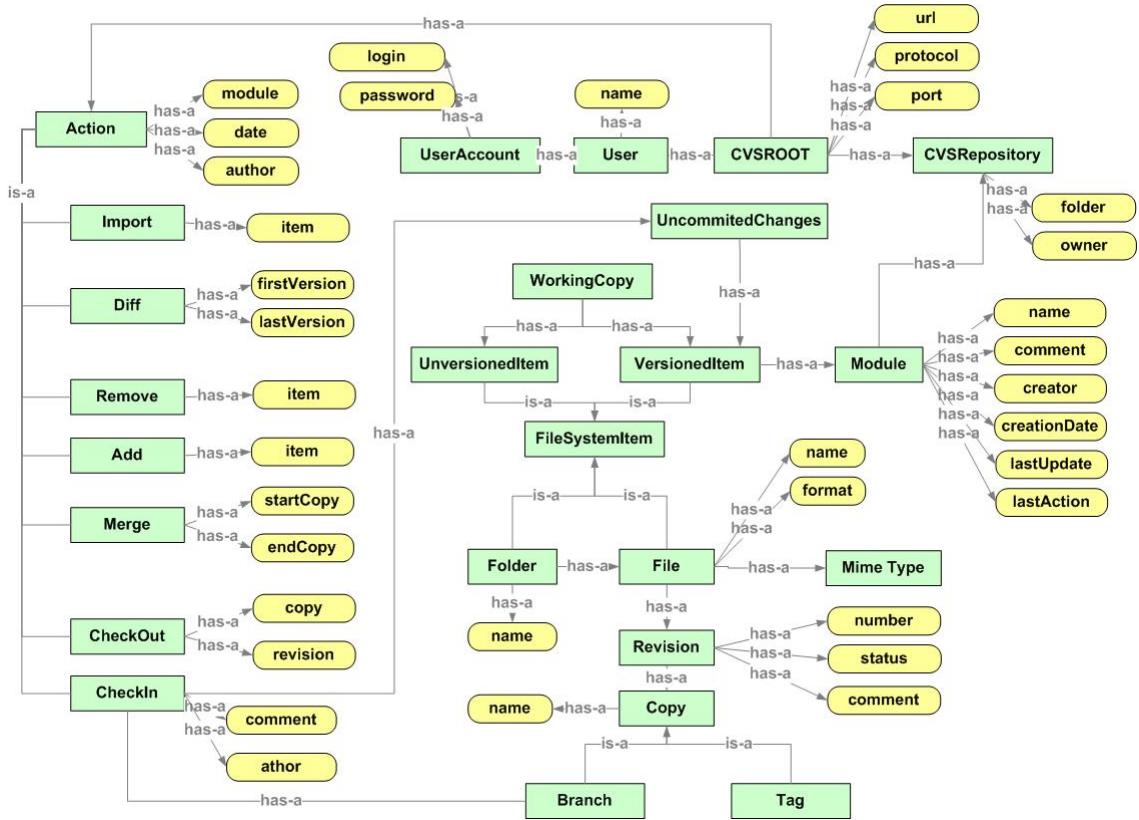
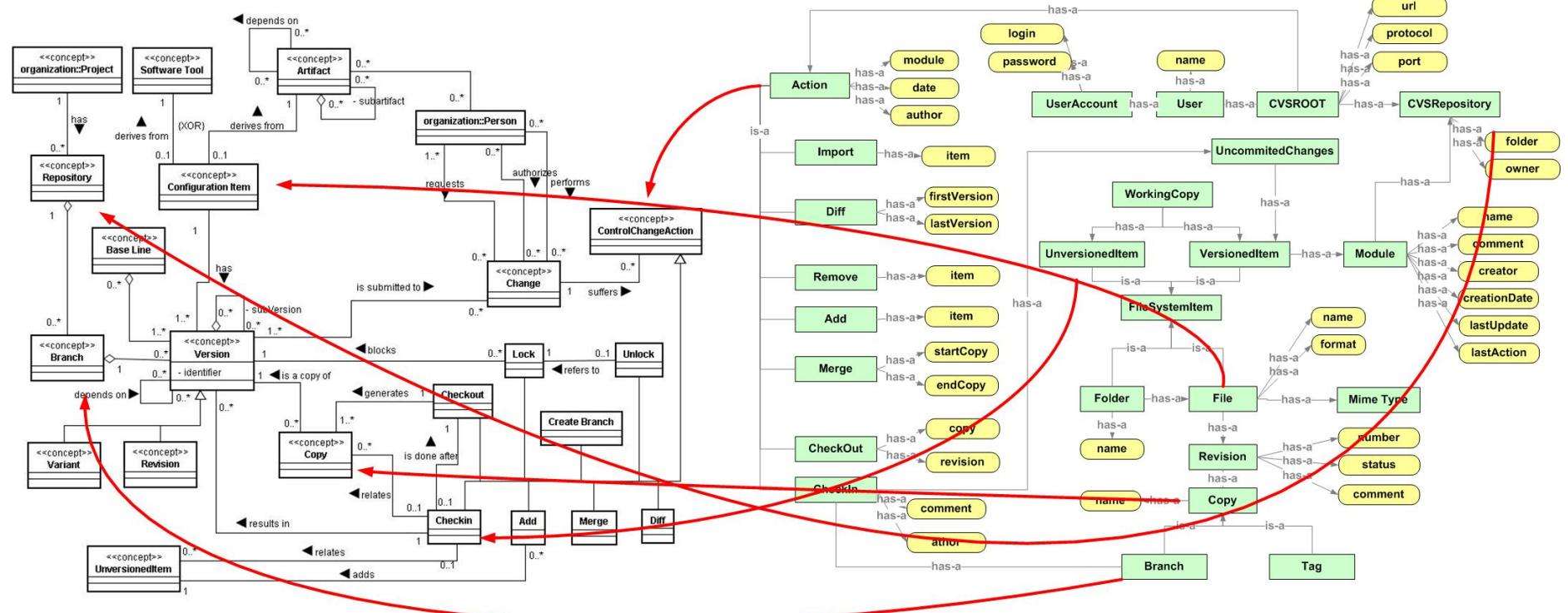


Figura 32 – Modelo Conceitual da ferramenta de gerência de configuração CVS

5.3 Mapeamento entre Ontologia de GCS e Modelo Conceitual do CVS

Assim como apresentado no Capítulo 4, uma ferramenta pode ser avaliada quanto a sua fidelidade na manipulação dos conceitos quando comparado seu modelo conceitual a uma ontologia de referência de mesmo domínio. Essa seção apresenta um mapeamento entre o modelo conceitual extraído da ferramenta de gerência de configuração de software CVS e a ontologia de domínio de GCS.

A Figura 33 apresenta o mapeamento gráfico entre a Ontologia de GCS e o modelo conceitual da ferramenta CVS.



Ontologia de Gerência de Configuração de Software

Modelo Conceitual do CVS

Figura 33 – Mapeamento entre a Ontologia de Gerência de Configuração de Software e o Modelo Conceitual do CVS

As ações de GCS não foram mapeadas graficamente com o objetivo de não deixar a figura muito carregada, com tantos mapeamentos. A Tabela 6 apresenta o mapeamento tabular com os conceitos relacionados entre o modelo conceitual do CVS e da ontologia de GCS.

Modelo Conceitual do CVS	Ontologia de Gerência de Configuração	Relação
Action	SCMAction	Ambos os conceitos estão relacionados a possíveis ações a serem tomadas sobre itens sob gerência de configuração. As ações especializadas tanto no modelo conceitual quanto na ontologia são basicamente as mesmas.
Add	Add	Ambos os conceitos estão relacionados com a ação de GCS usada para criar a primeira versão de um item não versionado
Branch	Branch	Ambos os conceitos estão relacionados aos ramos relativos a uma linha de desenvolvimento
Checkin	Checkin	Ambos os conceitos estão relacionados com a ação de GCS que retorna para o repositório cópias das variações geradas após uma ação prévia de <i>Checkout</i>
Checkout	Checkout	Ambos os conceitos estão relacionados a uma ação de GCS que objetivam criar cópias de variações a serem submetidas a mudanças
Copy	Copy	Ambos os conceitos estão relacionados a cópias de variações de itens que estão sob gerência de configuração
CVSRepository	Repository	Ambos os conceitos estão relacionados aos repositórios que armazenam os ramos com suas variações
Diff	Diff	Ambos os conceitos estão relacionados com a ação de GCS de comparar duas versões de um artefato
File	Configuration	Ambos os conceitos estão relacionados aos itens

	Item	que estão sob gerência de configuração
Merge	Merge	Ambos os conceitos estão relacionados com a ação de GCS que indica a fusão de versões diferentes de um mesmo artefato
Remove	Remove	Ambos os conceitos estão relacionados com a ação de GCS que indica que uma variação em um ramo será desativado.
Unversioned Item	UnversionedItem	Ambos os conceitos estão relacionados aos itens que não estão sob gerência de configuração

Tabela 6 – Mapeamento conceitual entre o modelo conceitual do CVS e da ontologia de GCS

Como podem ser notados, muitos dos principais conceitos relativos ao domínio de GCS podem ser espelhados entre a ontologia de referência e o modelo conceitual da ferramenta. Isso demonstra que essa ferramenta manipula seus conceitos seguindo um padrão proposto pelo senso comum da comunidade de estudiosos, cujo conhecimento levou ao desenvolvimento dessa ontologia de referência.

5.4 Mapeamento entre Ontologias de Referência

Nessa subseção será apresentado o mapeamento entre as ontologias de referência dos domínios de Processos de Software e de Gerência de Configuração de Software. Esse mapeamento é essencial para garantir que os domínios abordados possam ser integrados e assim, ferramentas que manipulam conceitos desses domínios possam da mesma forma ser integradas.

É importante ressaltar que esse mapeamento não é o mesmo que os já apresentados. Anteriormente, os mapeamentos eram entre modelos conceituais e ontologias e tinham a função de interpretação semântica enquanto que esse mapeamento entre ontologias tem como função a integração entre esses modelos. A Figura 34 apresenta o mapeamento gráfico entre as ontologias de referência.

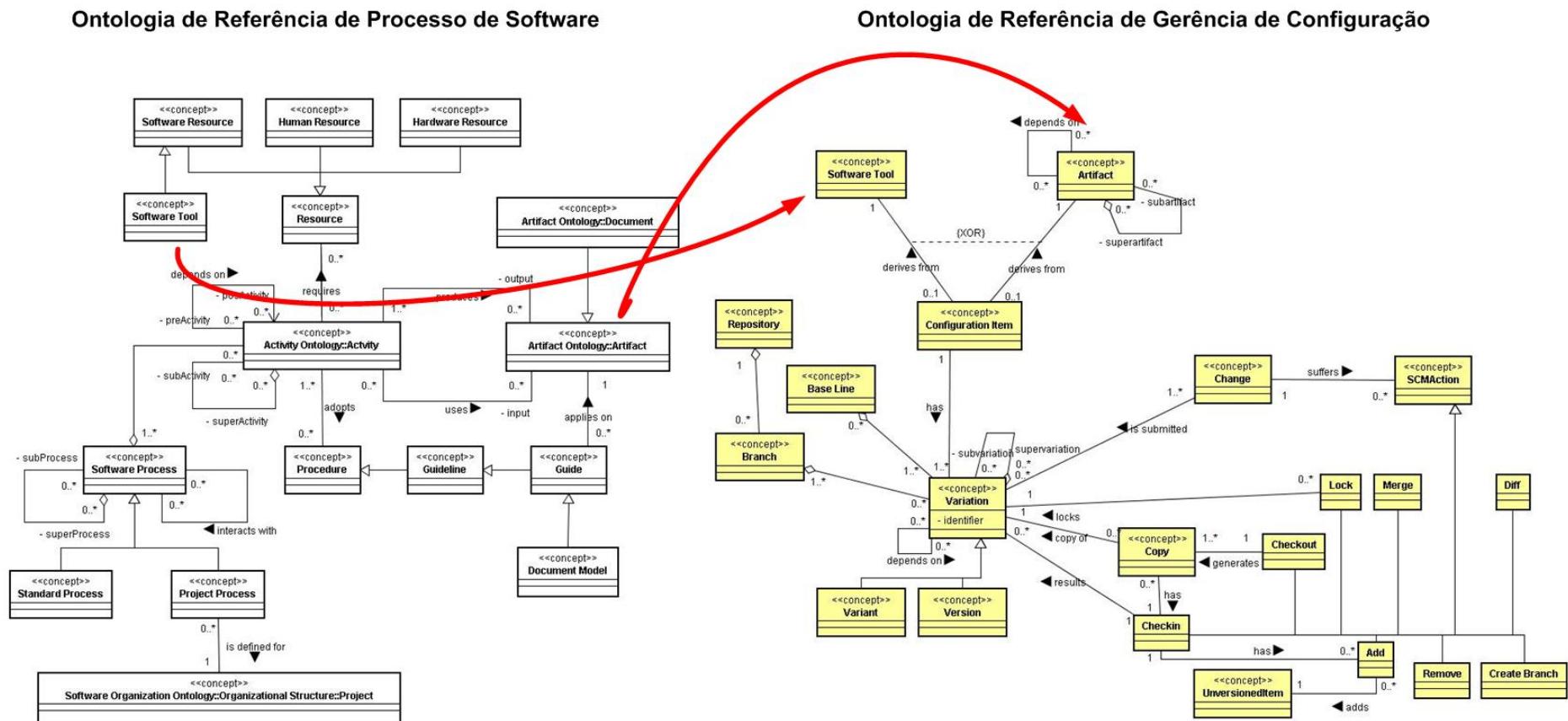


Figura 34 – Mapeamento entre as Ontologias de Referência de Processo de Software e Gerência de Configuração de Software

A Tabela 7 apresenta o mapeamento conceitual entre as ontologias de referência.

Ontologia de Processos de Software	Ontologia de Gerência de Configuração	Relação
Software Tool	Software Tool	Ambos os conceitos tratam as ferramentas de software usadas para auxiliar a execução de uma atividade. Uma ferramenta de software pode estar sob gerência de configuração, visto as várias versões que uma ferramenta pode ter.
Artifact	Artifact	Ambos os conceitos tratam os artefatos produzidos ou consumidos durante o ciclo de vida de um projeto. Num processo de desenvolvimento são gerados e consumidos muitos artefatos e é necessário que esses estejam sobre gerência de configuração.

Tabela 7 – Mapeamento entre conceitos das ontologias de Processo de Software e GCS

Esse mapeamento relaciona os conceitos de Artefato e Ferramenta de Software entre as duas ontologias. Esses são conceitos centrais, pois compõe o conceito chave de Item de Configuração na ontologia de GCS. Artefatos são insumos ou produtos no sentido de ser um objeto de transformação da atividade (FALBO, 1998). Visto que esses artefatos podem ser atualizados constantemente, é necessário que se faça um controle mais eficiente. Da mesma forma, Ferramentas de Software que servem para apoiar o processo de desenvolvimento sofrem atualizações de suas versões de um modo geral e como essas influenciam diretamente o desenvolvimento, é necessário manter um controle dessas versões, caso haja algum problema de incompatibilidade.

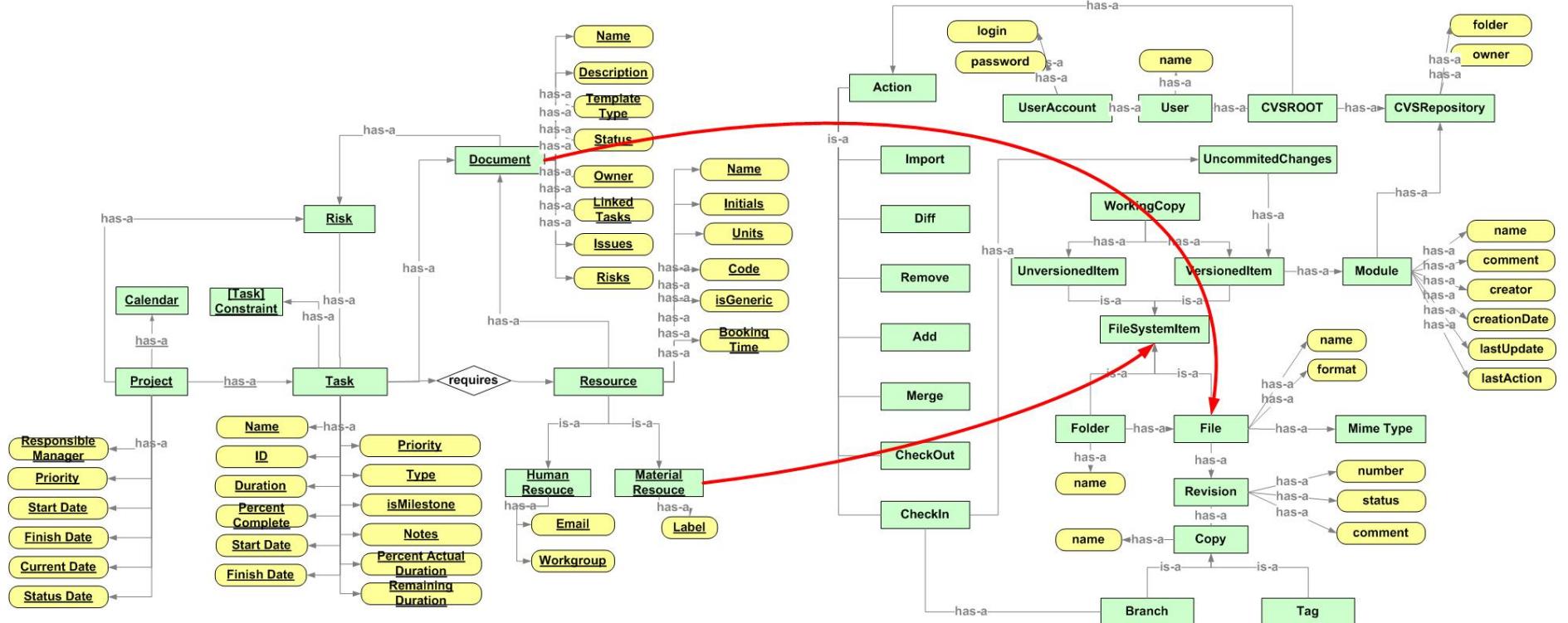
5.5 Mapeamento entre Modelos Conceituais

Com o mapeamento entre as ontologias de referência, foi demonstrado que os domínios de Processos de Software e Gerência de Configuração de Software possuem pontes semânticas, ou seja, possuem conceitos que detêm o mesmo significado. Foi verificado também que os modelos conceituais extraídos das ferramentas Microsoft Project e CVS,

referentes aos domínios de gerência de projetos e GCS, respectivamente, puderam ser mapeados em suas respectivas ontologias de referência. Ainda, foi verificado que o mapeamento entre os conceitos dos modelos conceituais é fortemente ligado com o mapeamento entre as ontologias de referência. Em outras palavras, os conceitos mapeados entre os modelos conceituais se referem, basicamente, aos mesmos conceitos mapeados entre as ontologias de referência. Isto indica que essas ferramentas manipulam seus conceitos de forma condizente com o proposto pelo senso comum.

Nessa seção será apresentado o mapeamento entre os modelos conceituais das duas ferramentas. Esse mapeamento tem como objetivo mostrar que essas ferramentas podem ser integradas, visto que compartilham conceitos semanticamente idênticos. Não é objetivo deste trabalho apresentar a integração e sim mostrar os pontos de integração entre as ferramentas.

A Figura 35 apresenta o mapeamento conceitual entre os modelos conceituais das ferramentas Microsoft Project e CVS. Por uma questão de visualização, alguns conceitos não relevantes a esse mapeamento não serão mostrados na figura abaixo.



Modelo Conceitual do Microsoft Project 2003

Modelo Conceitual do CVS

Figura 35 – Mapeamento entre os Modelos Conceituais do Microsoft Project e CVS

A Tabela 8 apresenta o mapeamento conceitual entre os modelos conceituais das ferramentas de gerência de projetos e gerência de configuração.

Modelo Conceitual do Microsoft Project	Modelo Conceitual do CVS	Relação
Document	File	Ambos os conceitos estão relacionados a artefatos. O Microsoft Project trata como um artefato produzido ou consumido durante um processo de desenvolvimento e o CVS trata como um arquivo que pode ser feito o seu controle de configuração.
Material Resource	File System Item	Material Resoucer pode ser especializado em recursos de hardware e ferramentas de software. Ferramentas de software podem estar sob gerência de configuração, visto as várias versões que uma ferramenta pode ter e está é referenciada como um File System Item pela ferramenta CVS.

Tabela 8 – Mapeamento entre os modelos conceituais do Microsoft Project e o CVS

Como pode ser visto, o mapeamento apresenta dois pontos de integração entre as ferramentas. Um deles se refere aos artefatos produzidos durante o ciclo de vida de um projeto, que devem ser controlados. O outro mapeamento se remete a ferramentas de software que podem apoiar atividades durante o ciclo de vida de um projeto e que podem estar sob gerência de configuração.

Com esse mapeamento, é demonstrado que ferramentas de diferentes domínios de conhecimento podem ser integradas. Para isso, além da integração conceitual entre os modelos conceituais, é necessário que 1) estes modelos devem estar semanticamente fundamentados por ontologias de referência de mesmo domínio, 2) essas ontologias devem possuir pontes semânticas e 3) o mapeamento entre os modelos conceituais devem refletir o mapeamento entre as ontologias de referência.

Capítulo 6

Considerações Finais

Organizações de um modo geral estão cada vez mais preocupadas em ter as informações relativas às suas áreas de atuação de forma rápida e consistente. O problema é que essas informações normalmente estão espalhadas em sistemas diferentes, que abrangem domínios diferentes, que não provém nenhum mecanismo de comunicação entre si. Dessa forma, é necessário prover um mecanismo de comunicação entre ferramentas, a fim de manterem uma base de informações única, evitando informações duplicadas e inconsistentes.

Este trabalho propôs uma forma de integração semântica entre ferramentas comercialmente disponíveis que apóiam um processo de desenvolvimento de software. Para isso, foram extraídos seus modelos conceituais a partir de técnicas também apresentadas neste trabalho. Esses modelos conceituais tiveram sua semântica explicitada a partir de seus mapeamentos em ontologias de referência, visto que os modelos conceituais, da forma que foram extraídos, não provêm nenhuma semântica referente a seus respectivos domínios. Dessa forma, os modelos conceituais puderam ser integrados semanticamente com outros modelos relativos a outras ferramentas, se baseando na integração entre suas respectivas ontologias de domínio. Outro objetivo atingido foi a avaliação de uma ferramenta se apta ou não para apoiar de forma adequada um processo de desenvolvimento. Isso foi feito a partir do mapeamento de seu modelo conceitual com uma ontologia de referência do mesmo domínio que a ferramenta está proposta a abordar.

Dessa forma, este trabalho abre as portas para a pesquisa de uma nova forma de integração de ferramentas. Foi apresentado uma abordagem de integração semântica entre ferramentas que pode ser aplicado em organizações de um modo geral.

Em termos de experiência adquirida, o trabalho foi muito importante, pois permitiu a consolidação dos conceitos aprendidos durante o curso de graduação em Ciência da Computação, como, por exemplo, modelagem conceitual, gerência de projetos e análise de sistemas. Outro fator importante, que não foi abordado com muita ênfase durante o curso, mas que foi bastante aplicado neste trabalho foram Ontologias. Um estudo extenso foi feito sobre esse assunto foi feito, abrindo um leque de possibilidades de aplicação dessa forma de representação de conhecimento. Uma experiência adquirida, talvez a mais importante durante todo este trabalho foi a prática de pesquisa científica. A experiência profissional que tive

durante o curso foi exclusiva em empresas de TI, logo, a prática em pesquisa científica foi inexistente. Este trabalho foi muito válido também neste sentido, já que tive que adentrar no mundo da pesquisa.

6.1 Perspectivas Futuras

Este trabalho apresenta algumas técnicas, mas praticamente todas devem ser feitas de forma manual. Algumas das perspectivas futuras estão relacionadas ao processo de escavação de ontologias desenvolvido por (HSI, 2005). Esse processo, além de manual, depende muito do conhecimento prévio a cerca do domínio por parte do responsável da escavação. Etapas como a identificação de conceitos e relacionamentos entre os conceitos dependem muito do conhecimento do domínio e sem esse conhecimento, essa etapa fica inviável de ser realizada. Dessa forma, seria interessante desenvolver alguma forma de automatização deste processo, diminuindo o grande trabalho de fazer a escavação dos conceitos e que isso possa ser feito de uma forma independente de domínio, para que mais ferramentas, dos mais diversos domínios possam ter seus modelos conceituais escavados de forma eficaz.

Outro ponto importante a ser tratado como trabalho futuro é refinar a forma de avaliação entre o mapeamento entre ontologias e modelos conceituais. Mapeamentos podem ser feitos, mas ainda não existe uma metodologia de avaliação própria para verificar se um modelo conceitual está bem fundamentado com uma ontologia de referência. Dessa forma, com uma avaliação mais detalhada, modelos conceituais de ferramentas poderão ser semanticamente explicitados com mais clareza e eficiência, facilitando sua possível integração com outras ferramentas.

Referências

- AMBRIOLA, V., CONRADI, R., FUGGETTA, A., “Assessing Process-Centered Software Engineering Environments”. ACM Transactions on Software Engineering and Methodology, v. 6, n.3, 283-328, July, 1997.
- ARANGO, G., “Domain Analysis – From Art Form to Engineering Discipline”, in Proc. of the 5th International Workshop on Software Specialization and Design, CS Press, Los Alamitos, California, USA, pp. 152-159, 1989.
- ARANGO, G., WILLIAMS, G., ISCOE, N., “Domain Modeling for Software Engineering”, Proceedings of ICSE 1991: The International Conference on Software Engineering, ACM Press, Austin, Texas, 1991.
- ARANTES, L. O., FALBO, R. A., GUIZZARDI, G., “Evolving a Software Configuration Ontology”, in Proceedings of the Second Brazilian Workshop on Ontologies and Metamodels for Software and Data Engineering (WOMSDE’07), 22nd Brazilian Symposium on Databases (SBBD)/21st Brazilian Symposium on Software Engineering (SBES), João Pessoa, Brazil, 2007.
- ARAÚJO, M.A.P., “Automatização do Processo de Desenvolvimento de Software nas Ambientes Instanciados pela Estação Taba”, Tese de Mestrado, Rio de Janeiro, 1998.
- BERTOLLO, G. Definição de Processos em um Ambiente de Desenvolvimento de Software. Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, 2006.
- CHANDRASEKARAN, B., JOSEPHSON, J.R., BENJAMINS, V.R., “What Are Ontologies and Why Do We Need Them?”. IEEE Intelligent Systems, pp. 20-26, Jan. 1999.
- CHEN, M., NORMAN, R.J., “A Framework for Integrated CASE”. IEEE Software, March, 1992.
- CHRISTIE, A. M., “Software Process Automation: The Technology and its Adoption”. Pittsburghm Pennsylvannia, Springer-Verlag Berlin Heidelberg, 1995.

ECLIPSE, THE ECLIPSE FOUNDATION, “Model Driven Development integration Project (MDDi)”, disponível em <<http://www.eclipse.org/proposals/eclipse-mddi/>>, acessado em 10 de Maio de 2007.

FALBO, R.A., “Integração de Conhecimento em um Ambiente de Desenvolvimento de Software”, Tese de Doutorado, COPPE/UFRJ, 1998.

FALBO, R.A.; GUIZZARDI, G.; DUARTE, K.C. "An Ontological Approach to Domain Engineering", XIV International Conference on Software Engineering and Knowledge Engineering (SEKE-2002), Ischia, Italy, ACM 2002.

FALBO, R.A., RUY, F.B., BERTOLLO, G., TOGNERI, D.F., “Learing How to Manage Risks Using Organizational Knowledge”, Advances in Learning Software Organizations (Proceedings of the 6th International Workshop on Learning Software Organizations - LSO'2004), Melnik G. and Holz, H. (Eds.): LNCS 3096, pp. 7-18, Springer-Verlag Berlin Heidelberg, Banff, Canada, June 2004.

FELHBERG, G. OLIVEIRA, F. ANTUNES, J. C., ARANTES, L. O., FALBO, R. A., GUIZZARDI, G., “Semantic Interoperability of COTS using ontologies and Process Models”, Relatório Técnico do Núcleo de Estudos de Ontologias (NEO), Universidade Federal do Espírito Santo (UFES), Vitória-ES, Brasil, 2007.

FERDINAND, M., et al., “Lifting XML Schema to OWL”, 4th International Conference on Web Engineering (ICWE), Munich, Germany, July, 2004.

GRUHN, V., “Process-Centered Software Engineering Environments – A Brief History and Future Challenges”. In: Annals of Software Engineering v. 14, 363-382, Netherlands, 2002.

GUIZZARDI, G., “Desenvolvimento Para e Com Reuso: Um Estudo de Caso no Domínio de Vídeo Sob Demanda”, Dissertação (Mestrado em Informática), Universidade Federal do Espírito Santo, 2000.

GUIZZARDI, G., “On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models”, Frontiers in Artificial Intelligence and Applications, Databases and

Information Systems IV, Olegas Vasilecas, Johan Edler, Albertas Caplinskas (Editors), ISBN 978-1-58603-640-8, IOS Press, Amsterdam, 2007.

GUZZARDI, G. et al., "An Ontology-based Approach for Evaluating the Domain Appropriateness and Comprehensibility Appropriateness of Modeling Languages", 8th Int. Conf. on Model Driven Engineering Languages and Systems, Jamaica, 2005.

GRUHN, V. Process-Centered Software Engineering Environments: A Brief History and Future Challenges. In: Kluwer Academic Publishers, 2002. Annals of Software Engineering 14, 2002. p. 363-382.

HARRISON, W., OSSHER, H., TARR, P., "Software Engineering Tools and Environments: A Roadmap". In: Proceedings of the Conference on the Future of Software Engineering - International Conference on Software Engineering, 261-277, Limerick, Ireland, 2000.

HAYES, P. "The Naïve Physics Manifesto", In D. Ritchie (Ed.) Expert Systems in Microelectronics age. Edinburgh University Press, pp 242-270, 1978.

HSI, I., "Analyzing the Conceptual Integrity of Computing Applications Through Ontological Excavation", PhD Thesis, August, USA, 2005.

IEEE, "SWEBOK - Guide to the Software Engineering Body of Knowledge", 2004 Version, IEEE Computer Society, 2004.

ISIS, INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS, "Web-based Open Tool Integration Framework (WOTIF)" disponível em <http://escher.isis.vanderbilt.edu/tools/get_tool?WOTIF>, acessado em 10 de Maio de 2007.

KAPPEL, G., KRAMLER, G., KAPSAMMER, E., REITER, T., RETSCHITZEGGER, W., SCHWINGER, W.: "ModelCVS - A Semantic Infrastructure for Model-based Tool Integration", Technical Report, 2005.

KAPSAMMER, E., REITER, T., SCHWINGER , W., "Model-Based Tool Integration - State of the Art and Future Perspectives". In: Proceedings of the 3rd International Conference on

Cybernetics and Information Technologies, Systems and Applications (CITSA 2006), Orlando, USA, July 2006.

MEALY, G. H., “Another Look at Data”, Proc. of the Fall Joint Computer Conference, Anaheim, California (AFIPS Conference Proceedings, Volume 31), Washington, DC: Thompson Books, London: Academic Press, 525–534, 1967.

NEIGHBORS, J., “Software Construction Using Components”, PhD Thesis, University of California, Irvine, EUA, 1981.

NOY, N., “Semantic Integration: A Survey of Ontology-Based Approaches”, SIGMOD Record, 33(4), December 2004.

OMG, OBJECT MANAGEMENT GROUP, “Ontology Definition Metamodel (ODM)”, disponível em <<http://www.omg.org/cgi-bin/doc?ad/2003-03-40>>, acessado em 10 de Maio de 2007.

OMG, OBJECT MANAGEMENT GROUP, “Unified Modeling Language (UML) Superstructure specification”, Version 2.2.1, 2005.

NUNES, V. B. Integrando Gerência de Configuração de software, documentação e gerência de conhecimento em um ambiente de Desenvolvimento de Software. Tese de Mestrado. UFES, Vitória, Brasil, 2005.

PRESSMAN, R.S., “Software Engineering: A Practitioner’s Approach”, 5^a Edição, Nova York: McGraw-Hill, 2000.

REICHMANN, C., “GeneralStore – a CASE-tool integration platform enabling model level coupling of heterogeneous designs for embedded electronic systems”, 11th Int. Conf. on Engineering of Computer Systems, May 2004.

RUSSEL, S., NORVIG, P. “Artificial Intelligence: A Modern Approach”. Prentice-Hall, Cap. 8, Building a Knowledge Base, 1995.

RUY, F.B., “Semântica em um Ambiente de Desenvolvimento de Software”, Dissertação (Mestrado em Informática), Universidade Federal do Espírito Santo, 2006.

SANCHES R. *Gerência de Configuração*. In: Qualidade de Software: Teoria e Prática, Eds. A.R.C. Rocha, J.C. Maldonado, K. Weber, Prentice Hall, 2001.

TRAVASSOS, G.H, “Ambientes de Desenvolvimento de Software”, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, 1994.

VOLZ, R., OBERLE, D., STAAB, S., STUDER, R. “OntoLift prototype”. Technical Report D11, WonderWeb project deliverable, 2002.

W3C WORLD WIDE WEB CONSORTIUM, “OWL Web Ontology Language Overview” disponível em <<http://www.w3.org/TR/owl-features/>>. Recomendação de W3C, Fevereiro, 2004.

WACHE, H. ,VÖGELE, T., VISSER, U., STUCKENSCHMIDT, H., SCHUSTER, G., NEUMANN, H., HÜBNER, S., “Ontology-Based Integration of Information — A Survey of Existing Approaches. In IJCAI-01 Workshop: Ontologies and Information Sharing, pages 108–117, 2001.

WEBER, R., “Ontological Foundations of Information Systems”. Coopers & Lybrand Research Methodology Monograph No. 4, Coopers & Lybrand, Melbourne (1997).