Resumo

Com a maior utilização e aumento das funcionalidades dos Sistemas de Computação, os Softwares tornaram-se maiores e mais complexos aumentando assim a necessidade de utilização de ferramentas, técnicas e metodologias para o gerenciamento de riscos apoiando os projetistas e os gerentes de projetos de Tecnologia da Informação e Comunicação (TIC).

Ainda hoje, muitos gerentes de projetos utilizam a estratégia de gerenciamento reativo ao risco, ou seja, tratar o risco de acordo com sua ocorrência. Para uma melhor utilização dos recursos (principalmente financeiro e temporal), é necessário ter uma metodologia de tratamento de riscos iniciando pelo planejamento da gerência de risco sendo um processo de âmbito organizacional.

Desenvolvimento de software é um esforço criativo, complexo e coletivo, é também uma tarefa de conhecimento intensivo, pois muitos recursos de informação são gerados e consumidos. Daí surge a necessidade da incorporação, no ambiente de desenvolvimento, de facilidades de gerência de conhecimento. Neste sentido Ontologias vêm sendo utilizadas como uma estrutura unificadora para dar semântica e uma representação comum à informação.

Este trabalho tem por objetivo redefinir a Ontologia *Onto*PRIME *(Ontology for Project Risk Management)* que faz parte da ferramenta *m*PRIME *Tool* (*Multiple Project Risk Management*) e que foi desenvolvida em Lógica de Primeira Ordem (LPO). A nova Ontologia, chamada de *m*PRIME *Ontology (**M**ultiple **P**roject **RI**sk **M**anag**E**ment **Ontology**)* foi desenvolvida em Lógica Descritiva (LD) em uma linguagem mais popular em Ontologias - *OWL-DL* - com o objetivo de dar uma maior expressividade à *Onto*PRIME e criar uma Ontologia no domínio de Gerência de Riscos em Ambientes de Desenvolvimento de Software de Múltiplos Projetos, permitindo uma maior organização e coerência dos conteúdos relacionados com esta área.

Para a realização do trabalho foi utilizada, além da linguagem OWL-DL, a ferramenta Protégé em sua versão 3.3.1, pelo fato da mesma dar suporte à linguagem utilizada, à construção de Ontologias e ainda à utilização de máquinas de raciocínio onde podem verificar a consistência e corretude da Ontologia construída.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

TIC	Tecnologia da Informação e Comunicação
***Onto*PRIME**	*Ontology for Project Risk Management*
***m*PRIME**	*Multiple Project Risk Management*
LPO	Lógica de Primeira Ordem
m*PRIME *Ontology	*Multiple Project Risk Management Ontology*
LD	Lógica Descritiva
OWL	*Web Ontology Web*
SEI	*Software Engineering Institute*
TQM	*Total Quality Management*
W3C	*World Wide Web Consortium*
≡	Equivalência
¬	Negação
∃	Existência
∀	Para todo
⊒	Inclusão
⊓	Operação Lógica “E”
⊔	Operação Lógica “OU”

Agradecimentos

Agradeço primeiramente Àquele que sempre me deu forças para superar todas as barreiras nas quais passei, DEUS.

Agradeço à minha mãe Cristina, ao meu pai Aristides, ao meu irmão Rafael, ao meus tios Irandi, Salvador e à todos os meus outros tios, minhas tias, primos e primas, meu avô Irineu, minha vó Helena, Paulo, Conceição à minha namorada Janaína e aos seus pais Vanderlei e Etiene.

Agradeço também à minha avó Walterlita e ao meu tio Washington, que, apesar de não estarem entre nós, sempre torceram e estão vibrando com mais esta conquista.

Agradeço à professora Dra. Cristine Gusmão, por ter sempre acreditado na realização deste trabalho e por me ajudar muito, em tudo que precisei, para a realização do mesmo. Não poderia esquecer do meu orientador da Iniciação Científica e do Estágio Supervisionado, professor Dr. Sérgio Soares.

Agradeço aos meus amigos que me acompanharam durante toda essa jornada Fábio, Érick, Alan, Eduardo e Rodrigo. Da mesma forma me acompanharam todos os integrantes da "Máfia POLI", grupo formado por amigos que se conheceram na Escola Politécnica de Pernambuco.

Agradeço aos donos da Econsultre, empresa na qual trabalho, que me liberaram em todos os momentos em que precisei para que minha formação fosse possível.

Também agradeço ao Neocaos, banda na qual faço parte, e ao glorioso Clube Náutico Capibaribe.

Enfim, agradeço a todos aqueles que colaboraram com mais esta realização em minha vida e que sempre confiaram em mim.

Capítulo 1

Introdução

Imprevisivibilidade e diversos fatores que são difíceis de controlar, como por exemplo, saída de um membro da equipe ou constantes mudanças nos requisitos do cliente, tornam a atividade de desenvolvimento de software bastante complexa sendo necessário um gerenciamento eficaz buscando o sucesso dos projetos. Desta forma, procura-se evitar que projetos de software tenham seus orçamentos e prazos ultrapassados, assim como o não cumprimento com às expectativas do cliente e um produto de qualidade inferior ao planejado.

O gerenciamento de riscos trabalha justamente com a incerteza, visando a identificação de problemas potenciais e de oportunidades antes que ocorram com o objetivo de eliminar ou reduzir a probabilidade de ocorrência e o impacto de eventos negativos para os objetivos do projeto além de potencializar os efeitos da ocorrência de eventos positivos [PMI, 2004]. Nesse contexto de incertezas, a Gerência de Riscos vem se tornando cada vez mais relevante nas organizações. Mesmo com a importância da Gerência de Riscos nas organizações, ainda é bastante comum encontrar gerentes que realizam apenas a gerência reativa, reagindo aos riscos apenas quando eles ocorrem, enquanto o mais indicado seria uma gerência pró-ativa, na qual fosse identificado o risco antes que ele ocorresse.

Muitos desses problemas se devem ao fato de que ainda não existe uma cultura de Gerência de Riscos dentro das empresas, a maioria dos gerentes tem um conhecimento limitado da área e não se sentem seguros, nem aptos a realizar essa gerência. Para difundir os conhecimentos acerca da Gestão de Riscos e, ao mesmo tempo, prover uma maior segurança para o gerente de projetos é necessária a definição de padrões, técnicas, ferramentas e metodologias sobre essa disciplina.

Dessa necessidade e visto que a Gerência de Riscos está intimamente relacionada com o sucesso dos projetos de software surge a *m*PRIME *Ontology*, uma versão aprimorada da *Onto*PRIME [Gusmão et al, 2004], uma Ontologia de riscos, criada com o intuito de dar suporte ao processo de Gerência de Riscos em Ambientes de Desenvolvimento de Software de Múltiplos Projetos, na qual também leva em consideração o relacionamento entre os riscos nos projetos. A *m*PRIME *Ontology* procura formalizar, compartilhar e definir conceitos, relacionamentos e axiomas no domínio de Gerência de Riscos provendo, assim, conhecimento sobre essa área.

O objetivo principal desse trabalho foi redefinir a *Onto*PRIME, desenvolvida em Lógica de Primeira Ordem, através de uma nova sintaxe utilizando linguagem descritiva.

O motivo para a passagem da Ontologia de Lógica de Primeira Ordem para Lógica Descritiva foi a necessidade de utilização de uma linguagem que desse uma maior expressividade à *Onto*PRIME. Dessa forma foi escolhida a linguagem *OWL-DL* [W3C, 2004] que, atualmente, é uma linguagem bastante utilizada na definição de Ontologias. Mais detalhes sobre a linguagem utilizada podem ser encontrados nas seções 3.4.

Para garantir a correta definição dos relacionamentos foi utilizada a ferramenta *Protégé* v*3.3.1* [Protégé, 2007] que possui suporte para a linguagem *OWL-DL*. Sua escolha se deve ao fato dela ser uma das ferramentas mais divulgadas e utilizadas no meio acadêmico na construção de Ontologias. Assim pretende-se, com o uso do *Protégé*, dar um maior suporte à futuras eventuais mudanças e correções na *m*PRIME *Ontology*.

1.1 Motivação

A Gerência de Riscos procura antecipar, minimizar e até mesmo mitigar os efeitos de eventos que possam gerar impactos negativos nos objetivos dos projetos de software. Aumento do orçamento do projeto, do prazo de entrega do produto e um produto de qualidade inferior ao desejado são algumas conseqüências da falta ou má gestão dos riscos.

É notável a necessidade de conhecimento dinâmico e acumulado do relacionamento existente entre os projetos, dos vários fatores adversos existentes e das soluções e atividades realizadas em situações similares anteriores, principalmente em Ambientes de Desenvolvimento de Software de Múltiplos Projetos.

Neste cenário surge a necessidade de ferramentas, métodos e técnicas que possam dar suporte às atividades de Gerência de Riscos fazendo com que uma maior quantidade de organizações sejam adeptas às atitudes pro-ativas aos riscos, ao invés das reativas, e uma maior quantidade de projetos tenham sucesso.

A utilização de Ontologias tem se tornado popular, em grande parte, pelo fato de terem como objetivo promover um entendimento comum e compartilhado sobre um domínio, que pode ser comunicado entre pessoas e sistemas de aplicação e oferecem um meio de lidar com a representação de recursos de informação sendo capaz de montar uma estrutura unificadora para dar semântica e uma representação comum à informação [J. Davies et al, 2003].

Tendo em vista o potencial do uso de Ontologias para lidar com o problema da semântica de recursos de informação, elas têm sido largamente explorado pelas áreas de pesquisa da *Web* Semântica [T. Berners-Lee et al, 2001] e da Gerência de Conhecimento, sendo neste momento aplicada a Gerência de Riscos, mais precisamente, na identificação de riscos.

1.2 Objetivos do Trabalho

Este trabalho tem por objetivo a definição e modelagem da *m*PRIME *Ontology* a qual dará suporte ao processo de Gerência de Riscos em Ambientes de Desenvolvimento de Software de Múltiplos Projetos e pelo desenvolvimento de um estudo científico sobre os relacionamentos dos

fatores de risco de software que influenciam o sucesso dos projetos, em ambientes organizacionais.

A definição da *m*PRIME *Ontology* parte da análise e adaptação de sua primeira versão da Ontologia, *Onto*PRIME, definida em Lógica de Primeira Ordem. Sendo um dos motivos para o uso da Lógica Descritiva, linguagem largamente usada na construção de ontologias, dando-se o nome dessa nova versão de *m*PRIME *Ontology*.

1.2.1 Objetivo Geral

Além de fornecer um vocabulário comum que poderá ser utilizado para representar conhecimento útil para os desenvolvedores de software sobre os riscos e oportunidades que podem afetar um projeto de software em uma organização, a *m*PRIME *Ontology* tem a finalidade de dar uma maior expressividade à Ontologia de riscos já desenvolvida, *Onto*PRIME, através de uma linguagem utilizada com maior freqüência na construção de Ontologias, *OWL-DL*.

1.2.2 Objetivo Específico

Através da *m*PRIME *Ontology* se procura formalizar, compartilhar e definir conceitos, relacionamentos e axiomas no domínio de Gerência de Riscos com intuito de fornecer uma estrutura que permita a organização do conhecimento neste domínio. Além disso, a *m*PRIME *Ontology* procura contemplar informações sobre riscos, processos de gerência de riscos, métodos, ferramentas e técnicas para identificação e controle de riscos.

1.3 Metodologia

Para um melhor entendimento do trabalho a ser realizado foi definida uma metodologia, onde um conjunto de atividades foi agrupado em 5 grandes fases. A metodologia empregada na realização desse trabalho pode ser detalhada na Tabela 1-A e explicada a seguir:

Fase 1: Inicialmente foi necessário um estudo do estado atual da *Onto*PRIME e de como riscos são tratados na Engenharia de Software. Também foram verificadas as fases pela qual a *Onto*PRIME passou, sendo analisada a origem da Taxonomia do SEI (*Software Engineering Institute*) [SEI, 2003], e verificados os axiomas gerados durante sua elaboração.

Fase 2: A seguir foi realizado um estudo sobre Ontologias conhecendo melhor os conceitos e definições inerentes a Ontologias, analisando seus componentes, classificações e linguagens de desenvolvimento. Também se fez necessário um estudo da Lógica Descritiva. Dessa forma as fontes utilizadas foram artigos, tutoriais, sites e trabalhos já realizados.

Fase 3: Por ter sido escolhido a ferramenta de suporte para a modelagem da *m*PRIME *Ontology*, o *Protégé*, que é um dos editores de Ontologias mais populares na comunidade acadêmica, se fez necessário um estudo detalhado sobre sua forma de utilização e funcionalidades.

Fase 4: Após as fases anteriores os axiomas da Ontologia original, *Onto*PRIME, definida em Lógica de Primeira Ordem, foi transformada em Lógica Descritiva fazendo as devidas alterações e adaptações, quando necessário, para que se possa tirar proveito da maior expressividade da Lógica Descritiva.

Fase 5: Por fim os axiomas foram definidos no *Protégé* tendo sido feito os devidos ajustes. Todas atividades e materiais desenvolvidos foram compiladas para o Trabalho de Conclusão de Curso.

Tabela 1-A. Descrição das Fases da Metodologia Aplicada

Fase	Descrição da Fase	Atividades
Fase 1	Estudo da *Onto*PRIME e Riscos na Engenharia de Software	Estudo do estado atual da *Onto*PRIME
		Estudo das fases da *Onto*PRIME
		Verificação dos axiomas da *Onto*PRIME
		Estudo dos riscos na Engenharia de Software
Fase 2	Estudo sobre Ontologias	Estudo sobre o conceito de Ontologia
		Elementos de Ontologias
		Classificação de Ontologias
		Linguagem de desenvolvimento de Ontologias
		Início da Escrita da Monografia
Fase 3	Estudo do *Protégé*	Estudo das funcionalidades do *Protégé*
		Linguagem de Implementação
		Continuação da Escrita da Monografia
Fase 4	Passagem dos Axiomas de LPO para LD	Verificação dos axiomas da *Onto*PRIME
		Estudo da Lógica Descritiva
		Continuação da Escrita da Monografia
Fase 5	Implementação no *Protégé*	Implementação dos axiomas no *Protégé*
		Ajustes necessários nos axiomas
		Criação de novos axiomas

1.4 Estrutura do Trabalho

Após este capítulo introdutório, os demais capítulos são descritos a seguir:

- **Capítulo 2 – Riscos na Engenharia de Software –** Neste capítulo são apresentados alguns conceitos de riscos assim como sua relação com a área de Engenharia de Software.

- **Capítulo 3 – Ontologias –** Capítulo em que é apresentado o conceito de Ontologias, classificação e linguagens de desenvolvimento. Também é realizada uma abordagem da importância da utilização de Ontologias descrição de um domínio específico. Ainda neste capítulo, é feita uma abordagem sobre a linguagem utilizada, *OWL-DL*, para o desenvolvimento da Ontologia.

- **Capítulo 4 – *m*PRIME *Ontology* –** Esse capítulo descreve os detalhes, desde o que foi necessário até os passoas de como a Ontologia foi contruída.

- **Capítulo 5 – Conclusão –** Capítulo que aborda os principais trabalhos relacionados assim como os trabalhos futuros. Relata a contruibuição, de forma geral, para a comunidade de Engenharia de Software.

As referências utilizadas ao longo do desenvolvimento desse Trabalho de Conclusão de Curso, bem como as implementações geradas, encontram-se disponibilizados no final desse documento, nas Referências Bibliográficas e no Apêndice A, respectivamente.

Capítulo 2

Riscos na Engenharia de Software

Neste capítulo serão apresentados os principais conceitos associados à Gerência de Riscos, suas carcaterísticas e relevância nos ambientes de desenvolvimento. Desta forma a Seção 2.1 faz uma introdução sobre a área apresentando alguns conceitos de riscos. Em seguida a Seção 2.2 trata das atividades de gerenciamento de riscos, onde vêm sendo aplicada com mais freqüência nas organizações. Na seção 2.3 relata algumas técnicas e métodos de identificação de riscos dando ênfase à taxonomia do SEI e Ontologias. Por fim a seção 2.4 traz um resumo do que foi apresentado no capítulo.

2.1 Introdução

Dentre as definições de risco existentes, algumas possuem um certo destaque. Knight, por exemplo, definiu risco como sendo a exposição a eventos incertos com probabilidades conhecidas [Knight, 1921]. Já o guia PMBOK define risco como sendo um evento ou condição incerta que, se ocorrer, terá um efeito positivo ou negativo em pelo menos um objetivo do projeto [PMBOK, 2004]. Peter Bernstein afirma que a origem da palavra risco vem do italiano antigo, *risicare* no qual significa "ousar", que por sua vez, deriva do Latin *risicu*, *riscu* [Bernstein, 1997]. De acordo com a última definição risco seria uma conseqüência da decisão livre sendo uma escolha e não cumprimento do destino.

Além de teorias modernas de gerenciamento, como por exemplo, Gerenciamento de Qualidade Total *(TQM – Total Quality Management)*, a Gerência de Riscos é baseada em teorias que provêem estratégias para tomada de decisões, onde, inicialmente, nas quais possuiam o objetivo de exercer um maior controle sobre os fenômenos naturais. Algumas aplicações dessas teorias podem ser vistas na obras *De Ludo Aleae* de Girolano Cardano [Larousse, 1998] *apud* [Gusmão & Moura, 2005], e *De Retiocinnis In Ludo Aleae* de Christian Huygens [Larousse, 1998] *apud* [Gusmão & Moura, 2005], onde o principal objeto de estudo é o jogo de azar. Ainda hoje, conceitos fundamentais do século XVII são utilizados pela Gerência de Riscos nas organizações.

2.2 Atividades de Gerenciamento de Riscos

Ainda que o conceito de risco esteja bastante associado a perigos e impactos negativos, ele já vem sendo utilizado como "exposição a conseqüências da incerteza", sendo cada vez mais aplicado tanto no gerenciamento de perdas quando no gerenciamento de ganhos. Como a incerteza é inerente ao início dos projetos, a gestão de risco sofre as conseqüências, podendo apresentar informações confusas, dúvidas e falta de conhecimento.

Barry W. Boehm [Boehm, 1991], representou o risco de forma sistemática, na Engenharia de Software, através do modelo em Espiral [Pressman, 1995], o qual tem como princípio ser incremental e dirigido à análise de riscos. O termo incremental se deve ao fato do progresso ser obtido em pequenos passos, pela divisão de um problema maior em problemas menores e a posterior combinção das soluções encontradas.

Para uma real eficácia, a Gerência de Riscos, deve possuir uma infra-estrutura adequada, metodologia lógica e sistemática para administrar os riscos associados à função, processos ou projeto. Dessa forma ela tem buscado alcançar o adequado balanceamento entre aproveitar as oportunidades e minimizar os eventos adversos sendo assim parte integrante das boas práticas de gestão empresarial.

É comum observarmos, nos ambientes organizacionais, atraso em cronogramas, gastos orçamentários além do previsto, produtos com qualidade inferior ao desejado. Embora esses problemas sejam considerados inerentes ao desenvolvimento de software por muitos autores, eles podem ser minimizados e controlados pelo contínuo gerenciamento de risco de projetos. Além disso a Gerência de Risco tem como objetivo a redução ou eliminação da probabilidade de que qualquer evento, que atrapalhe os objetivos do projeto, ocorra. Para que a Gerência de Riscos seja realizada de forma efetiva se faz necessário o seguimento de um conjunto de atividades básicas listadas a seguir e representadas na Figura 2-1:

- **Planejamenjar o gerenciamento de riscos:** É o processo que define como abordar e executar as atividades de gerenciamento de riscos de um projeto, assim como os recursos necessários para a realização do processo. O planejamento dos processos de gerenciamentode riscos define metodologia aplicada no gerenciamento de risco do projeto, funções e responsabilidades, orçamentos, freqüência da execução do processo de gerenciamento de riscos, categorias de riscos, probabilidade e impacto de riscos e modelos de relatórios. É importante notar que esse subconjunto de atividades pode gerar artefatos, como por exemplo, a matriz de probabilidade e impacto dos riscos. O processo planejamento do gerenciamento de riscos deve ser realizado no início do planejamento do projeto, já que ele é essencial para a execução com sucesso das atividades posteiores da Gerência de Riscos. Uma técnica importante para a produção do Plano de Gerenciamento de Riscos é a de realização de reuniões com os membros envolvidos no projeto. Esses membros podem incluir o gerente de projetos, membros da equipe do projeto selecionados e partes interessadas, qualquer pessoa da organização que tenha responsabilidade no gerenciamento das atividades de execução e planejamento de riscos, e outras pessoas, conforme necessário.

- **Identificar riscos:** Esta atividade procura identificar e documentar os riscos potenciais que podem afetar o projeto. Ela provê uma documentação que formaliza os dados coletados. Como novos riscos podem ser descobertos durante o desenvolvimento do projeto, esta atividade se

faz necessária ser um processo iterativo. Dessa atividade pode participar gerente de projetos, membros da equipe do projeto, equipe de gerenciamento de riscos, clientes e usuários finais. Algumas técnicas são utilizadas para dar suporte à esta atividade. Essas técnicas serão descritas na seção 2.3.

- **Análisar os riscos:** É a atividade que qual são estudados os aspectos mais importantes dos riscos, explorando as melhores estratégias para eliminação ou redução da probabilidade dos riscos. A análise dos riscos pode ser realizada de forma qualitativa, na qual utiliza a probabilidade deles ocorrerem, o impacto se os riscos realmente ocorrerem, o prazo e tolerância a risco das restrições de custo, cronograma, escopo e qualidade do projeto. Também pode ser realizada uma análise quantitativa, na qual realiza um estudo sobre os efeitos dos riscos e atribui uma classificação numérica a esses riscos.

- **Planejar respostas a riscos:** Envolve a determinação dos riscos a serem gerenciados assim como os planos de ação para evitá-los e os planos de contigência para os riscos que estão além da capacidade de eliminação, fazendo com que ocorra um aumento nas oportunidades e reduza as ameaças aos objetivos do projeto. Recursos e atividades de cronograma e plano de gerenciament do projeto são alocadas nesta atividade.

- **Monitorar riscos:** Atividade que faz o acompanhamento dos riscos identificados e faz o planejamento dos novos riscos encontrados. Observa se os procedimentos e políticas de gerenciamento de riscos estão sendo executados de forma correta. Seu objetivo é manter o controle dos riscos a fim de que se possa previní-los. Esta atividade utiliza dados gerados na execução do projeto para que técnicas, como análise de tendências e de variações, possam ser aplicadas. O monitoramento dos risco permite que a equipe de desenvolvimento tenha uma visão geral do progresso do projeto e é realizado em todo o ciclo de desenvolvimento do projeto.

- **Controlar riscos:** Atividade que engloba as alterações necessárias nas estratégias planejadas de eliminação dos riscos. Utiliza a ações do plano de contigência para diminuir ou eliminar um risco planejado ou que surgiu no decorrer do desenvolvimento do projeto. A utilização de um cronograma ajuda o acompanhamento do progresso e aumenta a eficácia dos planos de contingência.

- **Comunicar riscos:** É através da Comunicação dos Riscos que os riscos identificados na organização e as ações corretivas são demonstrados para as partes interessadas, tanto para os grupos estratégicos quanto para os grupos operacionais. Dessa forma, se faz necessária uma estrutura de comunicação bem montada para dar suporte a comunicação entre as equipes e membros do projeto.

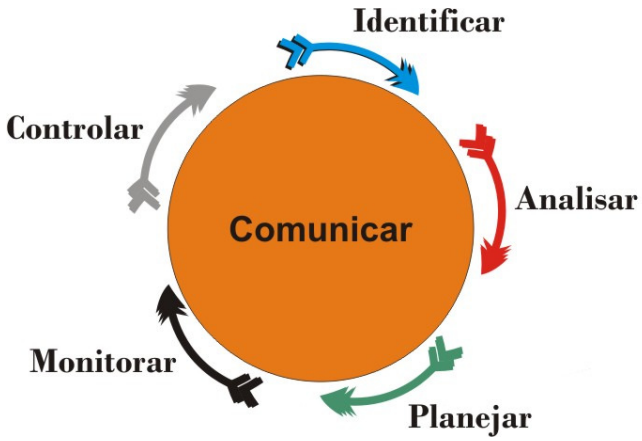

Figura 2-1. Atividades da Gerência de Riscos Segundo o SEI

2.3 Técnicas e Métodos de Identificação de Riscos

Como citado na seção anterior, a atividade de identificação de riscos, faz um levantamento dos possíveis riscos que podem influenciar de alguma forma no objetivo do projeto. Para dar suporte à realização desta atividade algumas técnicas são utilizadas. Veremos a seguir algumas dessas técnicas, inclusive *Brainstorming*, Técnica Delphi e Entrevista que são referenciados pelo guia PMBOK [PMBOK, 2004]:

2.3.1 Brainstorming

A técnica *Brainstorming*, desenvolvida por Alex Osborn [Alex Osborn, 1930], consiste na atividade em que os vários membros da equipe fazem sugestões a cerca dos prováveis riscos. Uma sessão de *Brainstorming* é feita com o único objetivo de produzir um grande número de idéias.

Tal atividade poderá envolver um conjunto multidisciplinar de especialistas que não fazem parte da equipe. As várias idéias geradas são documentadas com o objetivo de obter uma lista abrangente de riscos do projeto para posterior análise. Dessa forma As idéias devem ser organizadas segundo categorias lógicas e apresentadas ao grupo de avaliação para revisão. É comum a utilização de checklist tornando o processo mais prático. O grupo de avaliação deve verificar as melhores idéias de forma a sujeitá-las a testes práticos.

A *Brainstorming* é baseada no princípio do atraso do julgamento. Esse princípio diz que quando geramos idéias, é necessário ignorar as considerações à importância da idéia, à sua usabilidade, à sua praticabilidade. Assim. Nesse momento, todas as idéias são iguais. É necessário atrasar o julgamento enquanto ainda não se terminou a geração das idéias.

Outro princípio do *Brainstorming* é o da quantidade e qualidade da criatividade. Ele revela que quanto maior for o número de idéias geradas, mais provável será encontrar uma boa idéia. Assim a técnica *Brainstorming* vem sendo bastante utilizada na identificação de riscos nos ambientes organizacionais.

2.3.2 Técnica Delphi

A técnica *Delphi* foi definida por Turoff e Linstone [Turoff and Linstone, 1975] como sendo "um método para estruturar um processo de comunicação grupal de maneira que o processo é efetivo em permitir a um grupo de indivíduos, como um todo, a lidar com um problema complexo, ou seja, Delphi é uma ferramenta de pesquisa qualitativa que busca um consenso de opiniões de um grupo de especialistas a respeito de eventos futuros.

Delphi permite analisar dados qualitativos e permite descobrir as opiniões de especialistas através da realização de uma série de questionários. São apresentadas uma série de proposições específicas aos participantes para que, cada um individualmente, as ordenem mediante um dado critério estabelecido. Os resultados depois agregados são entregues aos especialistas, para que possam reformular as proposições apresentadas.

Este método distingue-se essencialmente por três características básicas, o anonimato, a interacção com "*feedback*" controlado e as respostas estatísticas do grupo. O especialista tem em cada rodada que responder a um questionário, definindo os vários itens apresentados por ordem de importância. Da segunda rodada faz parte as proprosições da primeira rodada ordenadas, além de novas proposições. O número de rodadas varia de acordo com o grau de consenso atingido pelos especialistas.

A cada nova rodada são introduzidas novas questões, o que pode ocasionar mudança nas opiniões dos especialistas em relação às questões que considera mais importantes. Para as rodadas, é necessário um facilitador que realiza que solicita as respostas do questionário. Como conseqüência da aplicação da técnica, temos um maior consenso entre os especialistas a respeito dos riscos e a dimuição de qualquer influência por parte dos participantes. Um ponto negativo nessa técnica é a limitação do domínio pelo questionário.

2.3.3 Entrevista

Essa técnica fornece o suporte para quantificar a probabilidade e conseqüências dos riscos nos objetivos do projeto. Dessa forma é necessário que os entrevistados sejam identificados e que o questionário sobre os riscos que possam afetar o projeto seja criado. Os entrevistados são partes envolvidas do projeto e especialistas no assunto.

Além disso, as entrevistas podem ser realizadas individualmente ou em grupos [Victoria et al, 2000], este último sendo necessário a utilização de um facilitador para a coleta de informações geradas.

A técnica de entrevista é bastante utilizada nos ambientes organizacionais, visto ao seu bom grau de fidelidade com situações reais. Dentre as devantagens podemos citar que o questionário criado pode não limitar as respostas dos entrevistados e também pode existir algum tipo de dependência entre o entrevistado e o entrevistador.

2.3.4 Taxonomia do SEI

A taxonomia do SEI foi desenvolvida em 1993. Ela é composta por um questionário [Carr M. et al, 1993] que tem o objetivo de identificar riscos de forma sistemática auxiliando gerentes de

projetos e fazendo com que mais projetos tenham sucesso. Classes, elementos e atributos compõem os três níveis da taxonomia de riscos.

A taxonomia é composta pelas seguintes categorias, como mostra a Figura 2-2:

- **Engenharia de Produto:** Englobam as atividades que mencionam questões técnicas, envolvidas na criação do produto que satisfaça requisitos específicos e expectativas do cliente, que podem provocar o insucesso do projeto. Essas atividades incluem análise e especificação dos requisitos de sistema e software, modelagem de software e implementação, integração de componentes de hardware e software e testes de software e sistema. Riscos da Engenharia de Produto geralmente resultam de requisitos que são tecnicamente difíceis ou impossíveis de implementar.

- **Ambiente de Desenvolvimento:** Classe que menciona o ambiente no qual o projeto está sendo desenvolvido assim como o processo utilizado. Processo de desenvolvimento e sistema, métodos de gerenciamento e ambiente de trabalho estão envolvidos nessa categoria.

- **Restrições de Programa:** Englobam fatores externos ao projeto, ou seja, fatores que normalmente estão fora do controle do projeto e mesmo assim influenciam o sucesso ou constituiem fonte de riscos para o projeto.

TAXONOMIA DE RISCOS		
Engenharia do Produto	**Ambiente de Desenvolvimento**	**Restrições de Programa**
Requisitos	**Processo de Desenvolvimento**	**Recursos**
Estabilidade	Formalidade	Cronograma
Completude	Adequabilidade	Equipe
Claridade	Controle de Processo	Orçamento
Validade	Familiaridade	Facilidades
Viabilidade	Controle de Produto	**Contrato**
Precedente	**Sistema de Desenvolvimento**	Tipos de Contrato
Escala	Capacidade	Restrições
Design	Adequabilidade	Depêndencias
Funcionalidade	Usabilidade	**Interfaces de Programas**
Dificuldade	Familiaridade	Cliente
Interfaces	Confiabilidade	Contratantes Associados
Performance	Siuporte do Sistema	Subcontratos
Testabilidade	Entrega	Contratante Principal
Limitações de Hardware	**Processo de Gerenciamento**	Gerenciamento Corporativo
Software Não Desenvolvido	Planejamento	Vendedores
Tesde de Código e Unidade	Organização do Projeto	Política
Viabilidade	Experiência em Gerenciamento	
Testes	Interfaces de Programa	
Codificação / Implementção	**Métodos de Gerenciamento**	
Integração e Teste	Monitoramento	
Ambiente	Gerenciamento de Pessoal	
Produto	Garantia de Qualidade	
Sistema	Gerenciamento de Configuração	
Engenharia de Especialidades	**Ambiente de Trabalho**	
Manutenibilidade	Atitude de Qualidade	
Confiança	Cooperação	
Proteção	Comunicação	
Segurança	Moral	
Fatores Humanos		
Especificações		

Figura 2-2. Taxonomia de Riscos do SEI [SEI, 2003]

Como dito anteriormente a taxonomia do SEI é baseada em um questionário, onde as questões se encontram divididas de acordo com as categorias conforme a Figura 2-2. O questionário pode conter questões que não são relevantes para todos os estágios do ciclo de desenvolvimento de software, seja pelo domínio específico do software ou pela organização específica do projeto. O objetivo do questionário é levantar riscos através das respostas que possam indicar um problema potencial.

2.3.5 Ontologias

Com o objetivo de promover entendimento comum e compartilhado sobre um determinado domínio, a crescente utilização de Ontologias é notável [Gusmão, 2007]. Desta forma o conhecimento pode ser comunicado e padronizado. Na área específica de Gerência de Riscos, uma ontologia pode definir um vocabulário para riscos,tendo em vista uma certa realidade, podendo ainda definir um conjunto de relações e definições de forma rigorosa a fim de proporcionar a identificação de riscos. Como exemplo podemos citar a *Onto*PRIME [Gusmão et al, 2004], que é uma ontologia de riscos na qual busca identificar riscos de projeto em um ambiente de múltiplos projetos. A *Onto*PRIME utiliza a Lógica de Primeira Ordem para definir os axiomas e também a relações existentes entre os riscos. No capítulo 3 veremos com mais detalhes Ontologias citando suas classificações, elementos e linguagens de definição.

2.4 Resumo do Capítulo

Este capítulo apresentou algumas atividades de gerenciamento de riscos, onde buscam a redução ou eliminação da probabilidade de que qualquer evento, que atrapalhe os objetivos do projeto. Dentre tais atividades destaca-se, por ser de suma importância para o presente trabalho, a identificação de riscos, onde busca identificar e documentar os riscos potenciais que podem afetar o projeto.

O capítulo ainda abordou algumas técnicas e métodos de identificação de riscos. Dentre elas destacamos a taxonomia do SEI, na qual é dividida nas categorias Engenharia de Produto, Ambiente de Desenvolvimento e Restrições de Programa. Ainda destacamos a Ontologia que é capaz definir um conjunto de relações e definições a fim de proporcionar a identificação de riscos.

Desta forma procuramos mostrar a relevância desde trabalho nos ambientes organizacionais que procuram obter uma maior quantidade de projetos que tenham sucesso.

Capítulo 3

Ontologias de Domínio

Historicamente o termo Ontologia vem do grego *"Ontos"* no qual significa "ser" e *"Logia"* que significa "estudo ou conhecimento". O termo Ontologia deu nome a um ramo da metafísica e foi introduzido por Aristóteles. Assim Ontologias vêm sendo bastante utilizadas no auxílio a sistemas computacionais. Uma aplicação prática é a procura, organização, acesso e a manutenção das informações na *Web* por usuários pelo fato de, atualmente, o crescimento do volume de informações se dar de forma rápida. Uma Ontologia é criada por especialistas e define as regras que regulam a combinação entre termos e relações em um domínio do conhecimento. Assim consultas podem ser realizadas de acordo com os conceitos definidos pela ontologia

Este capítulo tratará os conceitos básicos associados a ontologias, onde na Seção 3.1 apresentamos a visão de alguns autores sobre a definição de ontologia. A seção 3.2 mostra os elementos mais comuns de uma Ontologia, já que nem sempre possuem a mesma estrutura.

3.1 Definições

Segundo Borst, Ontologia pode ser definida como sendo "Uma Ontologia é uma especificação formal e explícita de uma conceitualização compartilhada" [Borst, 1997]. Guarino, por sua vez, define Ontologia como o que se refere a um artefato constituído por um vocabulário usado para descrever uma certa realidade, mais um conjunto de fatos explícitos e aceitos que dizem respeito ao sentido pretendido para as palavras do vocabulário. Este conjunto de fatos tem a forma da teoria da lógica de primeira ordem, onde as palavras do vocabulário aparecem como predicados unários ou binários [Guarino, 2001]. Já Goméz-Perez, afirma que uma ontologia é um conjunto de termos ordenados hierarquicamente para descrever um domínio que pode ser usado como um esqueleto para a base de conhecimentos [Gomez & Perez, 1999].

O vocabulário formado por predicados lógicos forma a rede conceitual que confere o caráter intencional às ontologias. A ontologia define as regras que regulam a combinação entre os termos e as relações. As relações entre os termos são criadas por especialistas, e os usuários formulam consultas usando os conceitos especificados. Uma ontologia define assim uma "linguagem" (conjunto de termos) que será utilizada para formular consultas. Ontologias vêm sendo utilizadas em diversos áreas, como por exemplo, na *Web* Semântica. Nesse contexto o objetivo da Ontologia é associar conhecimento do significado aos recursos da *Web*, tipicamente através da

utilização de (meta) dados processáveis por máquinas. Cada conceito pode estar relacionado a outros conceitos e pode ter um grupo de recursos de informação associados. Em outras áreas, como Inteligência Artificial, Medicina e Engenharia de Software, também é possível encontrar Ontologias definidas.

3.2 Elementos de uma Ontologia

Nem sempre as Ontologias possuem a mesma estrutura, mas alguns componentes e características se tornam comuns em sua grande maioria. Segundo Gruber [Gruber, 1996] os componentes básicos de uma Ontologia são, conforme Figura 3-1:

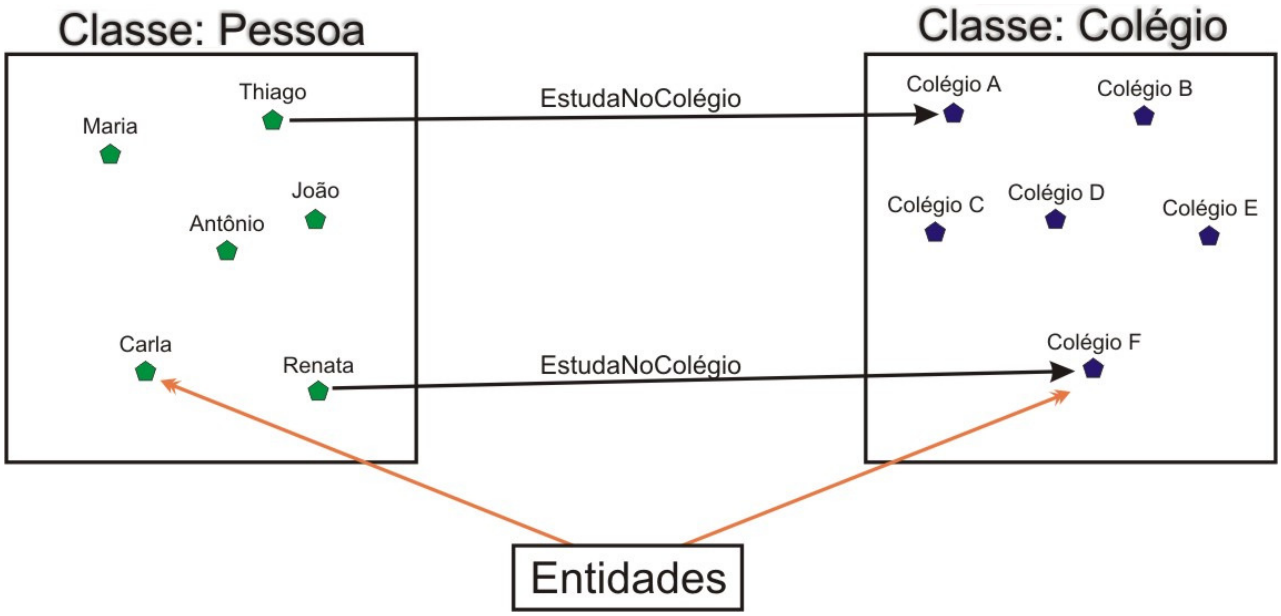

Figura 3-1. Exemplo de Relação em Ontologias

- **Classes:** Também conhecidas como conceitos, são organizadas em uma taxonomia. Como exemplo podemos citar que a classe "Pessoa" seja a classe que representa todas as pessoas. Sendo a classe "Colégio" a classe que representa todos os colégios de uma bairro, cidade, etc., dependendo da aplicação;

- **Relações:** Representam a interação entre os conceitos de um determinado domínio. Para ilustrar poderíamos dizer que a relação "EstudaNoColegio" liga a classe "Pessoa" à classe "Colégio". Uma ilustração gráfica pode ser visualizada na Figura 3-1;

- **Axiomas:** Modelam sentenças que são sempre verdadeiras;

- **Instâncias:** Representam elementos específicos do domínio. Poderíamos dizer que Maria é uma instância da classe "Pessoa".

3.3 Classificação de Ontologias

Existem vários classificações para Ontologias. As classificações de Ontologia pode variar de acordo com o que se pretende modelar, com a finalidade de utilização ou ainda com o nível de complexidade que a mesma deve possuir. Neste trabalho utilizaremos Ontologia de Domínio, na qual está inseria da classificação de Guarino [Guarino, 2001].

A classificação de Guarino, que leva em consideração o nível de generalidade necessária, engloba: Ontologias de Alto-Nível em que descrevem conceitos gerais que são independentes de um problema particular ou domínio; Ontologias de Tarefas em que descrevem um vocabulário relacionado a uma tarefa ou atividade genérica, através da especialização de conceitos introduzidos nas ontologias de alto-nível; Ontologias de Aplicação que são Ontologias mais específicas e podem especializar conceitos das Ontologias de Tarefas ou de Domínio; e por fim temos as Ontologias de Domínio em que descrevem um vocabulário relacionado a um domínio específico.

Como dito anteriormente, este trabalho trata da redefinição de uma Ontologia de Domínio, onde o objetivo é a modelagem de riscos em Ambientes de Desenvolvimento de Software de Múltiplos Projetos. No capítulo 4 veremos com mais detalhes como foi realizada a elaboração da Ontologia de riscos, chamada *m*PRIME *Ontology*. O motivo da escolha das Ontologias de Domínio se deve ao fato delas serem capazes de definir um vocabulário comum ao domínio, permitindo assim compartilhar, formalizar e definir conceitos.

3.4 Linguagem OWL

A *OWL (Web Ontology Language)* é uma linguagem recomendada pela *W3C (World Wide Web Consortium)* [W3C, 2004] que desenvolve padrões para a criação e a interpretação dos conteúdos para web. A *OWL* foi projetada para o uso por aplicações que precisam processar o conteúdo da informação ao invés de apenas apresentá-la aos humanos. Assim ela é capaz de definir Ontologias através de um conjunto de classes, de propriedades e de restrições.

Para formar a base do conhecimento, *OWL* utiliza alguns operadores como *and, or, some* (representado por ∃), *only* (representado por ∀), etc. Com a linguagem ainda é possível trabalharmos com máquinas de raciocínio na qual verifica a consistência da Ontologia além de auxiliar na corretude da hierarquia construída.

A *OWL* ainda pode ser dividida em três sublinguagens, *OWL-Lite, OWL-DL e OWL-Full*. Estas sublinguagens foram assim classificadas devido às suas expressividades e possuem as seguintes características de acordo com a *W3C*:

- ***OWL-Lite:*** Dá suporte a usuários em que sua necessidade é obter uma hierarquia e restrições simples. Por exemplo, embora suporte restrições de cardinalidade, ela só permite valores de cardinalidade 0 ou 1 e pode absorver migrações de tesauros e outras taxonomias de forma mais rápida. *OWL-Lite* é menos complexa que a *OWL-DL.*

- ***OW- DL:*** É mais expressiva que a *OWL-Lite*. *OWL-DL* inclui todas as construções da linguagem *OWL*, porém elas somente podem ser usadas com algumas restrições, como por

exemplo, uma classe não pode ser instância de outra classe, mesmo podendo ser subclasse de outras classes. Como o próprio nome diz, *OWL-DL* (o *DL* vem de *Descriptive Logic)* é baseada em lógica descritiva e permite o uso de máquinas de raciocínio.

- ***OWL-Full:*** É mais expressiva que a *OWL-DL*. Como exemplo de uma maior expressividade podemos citar que em *OWL-Full* uma classe pode ser tratada simultaneamente como uma coleção de indivíduos e como um indivíduo por si mesma.

De acordo com as definições acima podemos dizer:

- Toda ontologia *OWL-Lite* válida é uma ontologia *OWL-DL* válida;
- Toda ontologia *OWL-DL* válida é uma ontologia *OWL-Full* válida;
- Toda conclusão *OWL-Lite* válida é uma conclusão *OWL-DL* válida;
- Toda conclusão *OWL-DL* válida é uma conclusão *OWL-Full* válida.

Algumas ferramentas no mercado dão suporte a linguagem *OWL*. Uma dessas ferramentas é o *Protégé* [Protégé, 2007], um ambiente para criação e edição de ontologias e bases de conhecimento, na qual foi utilizado neste trabalho assim como a linguagem *OWL-DL*.

3.5 Resumo do Capítulo

Neste capítulo foi apresentado a origem do termo Ontologia, onde "*Ontos*" significa "ser" e "*Logia*" significa "estudo ou conhecimento". Na seção 3.1 foram apresentadas algumas definições do termo Ontologia, onde destacamos a de Guarino, em que define Ontologia como sendo o que se refere a um artefato constituído por um vocabulário usado para descrever uma certa realidade, mais um conjunto de fatos explícitos e aceitos que dizem respeito ao sentido pretendido para as palavras do vocabulário.

Posteriormente, na seção 3.2, foi introduzido os elementos mais comuns em Ontologias, já que nem todas as Ontologias possuem estruturas iguais. Dentre tais elementos podemos destacar as classes, que representam conceitos e são representados em uma taxonomia, relações que representam a interação entre os conceitos de um determinado domínio, axiomas, sentenças sempre verdadeiras, e instâncias que representam elementos específicos no domínio.

Na seção 3.3 tratamos de algumas classificações de Ontologias, dando ênfase às Ontologias de Domínio, em descrevem um vocabulário relacionado a um domínio específico, já que este trabalho trata da redefinição de uma Ontologia de Domínio, onde o objetivo é a modelagem de riscos em Ambientes de Desenvolvimento de Software de Múltiplos Projetos.

Por fim na seção 3.4 apresentamos a linguagem *OWL*, onde apresentamos suas sublinguagens: A *OWL-Lite*, *OWL-DL* e *OWL-Full*, sendo a *OWL-Lite* a sublinguagem menos expressiva das três. A *OWL-DL* é mais expressiva quando a comparamos com a *OWL-Lite*, porém menos expressiva quando a comparamos com a *OWL-Full*, conseqüentemente a mais expressiva de todas. Nesta mesma seção apresentamos a ferramenta utilizada na realização deste trabalho, o *Protégé.*

Capítulo 4

mPRIME Ontology

Algumas premissas se fizeram necessárias para a realização deste trabalho. A seção seguinte 4.1 traz essas prmissas assim como suas respectivas explicações. A seção 4.2 trata da metodologia utilizada para a construção da Ontologia, e faz uma abordagem da sub-Ontologia de Ambiente de Desenvolvimento e de seus elementos conforme o SEI.

4.1 Premissas

4.1.1 *Onto*PRIME

A *Onto*PRIME foi proposta com o objetivo de representar uma Ontologia de riscos que desse suporte ao processo de Gerência de Riscos em Ambientes de Desenvolvimento de Software de Múltiplos Projetos. Desenvolvida em Lógica de Primeira Ordem, ela também aborda o relacionamento dos fatores dos riscos software que influenciam o sucesso dos projetos, em ambientes organizacionais e fornece um vocabulário comum que pode ser utilizado para representar conhecimento útil para os desenvolvedores de software sobre os riscos e oportunidades que podem afetar um ou mais projetos dentro de uma organização que desenvolve software.

Sendo construída com base na taxonomia do SEI (ver Figura 2-2), a *Onto*PRIME possui três níveis: classes, elementos e atributos. As três maiores classes que compõem a Ontologia são: Engenharia de Produto que inclui as atividades que mencionam questões técnicas, envolvidas na criação do produto que podem provocar o insucesso do projeto; Ambiente de Desenvolvimento que envolve os fatores de riscos do ambiente no qual o projeto está sendo desenvolvido; e a de Restrições de Programa que menciona os fatores que normalmente estão fora do controle do projeto mas que de certa forma influenciam o projeto podendo gerar riscos.

A Figura 4-1 mostra a relação existente da relação das sub-Ontologias, onde foi definida por Gusmão [Gusmão et al, 2004], inicialmente identificando e introduzindo termos e frases potenciais ao domínio em questão.

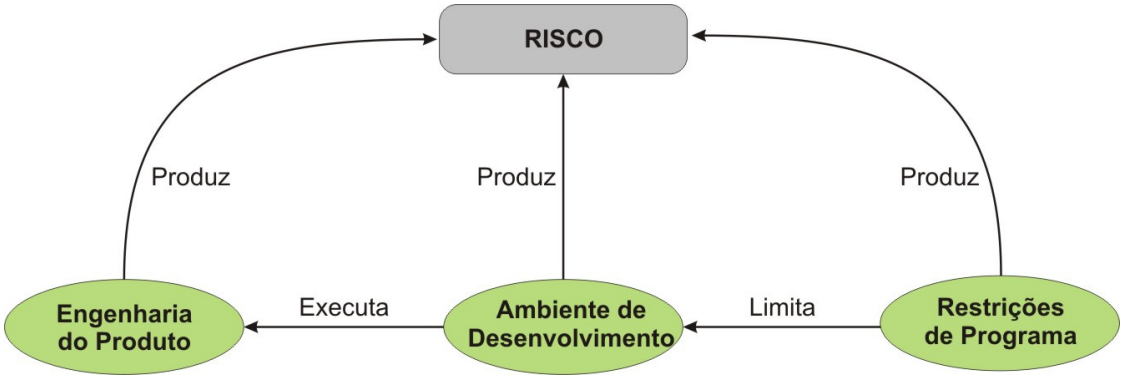

Figura 4-1. Sub-Ontologias da Ontologia de Riscos [Gusmão et al, 2004]

A seguir apresentamos alguns exemplos de axiomas da classe Ambiente de Desenvolvimento da *Onto*PRIME em Lógica de Primeira Ordem. Lembramos que o intuito dos exemplos citados é obter um melhor entendimento de onde partiram a definição da maioria dos axiomas da *m*PRIME *Ontology*. Por esse motivo não se faz necessário a apresentação de todos os axiomas.

Axioma da subclasse Familiaridade da classe Risco de Processo de Desenvolvimento

Definição: Se existe algum projeto P no qual não há experiência, conhecimento e conforto com o processo utilizado, então existe um risco de familiaridade com processo de desenvolvimento para P.

Considerando x sendo um processo do projeto p, temos:

$\exists$ p, x . Project(p) $\wedge$ Process(x, p) $\wedge$ $\neg$ Knowledge(x) $\wedge$ $\neg$ Experience(x) $\wedge$ $\neg$ Comfort(x) $\rightarrow$ DevelopmentProcessFamiliarityRisk(p)

Axioma da subclasse Adequabilidade da classe Risco de Sistema de Desenvolvimento

Definição: Se existe algum projeto P que possui um baixo grau de auxílio por parte dos modelos de desenvolvimento, ou processos, ou procedimentos, ou atividades requeridas e selecionadas para o programa, então existe um risco de adequabilidade de sistema de desenvolvimento para P.

Considerando x sendo o modelo de desenvolvimento para o projeto p, y como sendo um processo de p, z como sendo procedimento do projeto p e w como sendo aitividades requeridas para a execução do do projeto p, temos:

$\exists$ p, x, y , z, w . Project(p) $\wedge$ DevelopmentModel(x, p) $\wedge$ Process(y, p) $\wedge$ Procedures(z, p) $\wedge$ RequiredActivities(w, p) $\wedge$ (LowSupportiveDegree(x) $\vee$ LowSupportiveDegree(y) $\vee$ LowSupportiveDegree(z) $\vee$ LowSupportiveDegree(w)) $\rightarrow$ DevelopmentSystemSuitabilityRisk(p)

Axioma da subclasse Gerenciamento de Pessoal da classe Risco de Métodos de Gerenciamento

Definição: Se existe um projeto P no qual não é fácil para os membros do projeto serem gerenciados, então existe risco de gerenciamento de pessoal.

Considerando y sendo algum membro da equipe envolvida no projeto p, x sendo uma instância da relação que o membro da equipe possui com o projeto e z sendo uma instância da propriedade *Easy*, temos:

$\exists$ p, x, y, z . Project(p) $\wedge$ People(y) $\wedge$ Managed(y, p) $\wedge \neg$ Easy(z) PeopleHave(x, p) $\rightarrow$ PersonnelManagement(p)

Axioma da subclasse Experiência da classe Risco de Processo de Gerenciamento

Definição: Se existe algum projeto P para o qual os gerentes não sejam experientes, então existe risco de experiência de gerenciamento.

Considerando x sendo a característica de determinado requisito que o gerente de projetos, representado pela letra y, possui ou não.

$\exists$ p, x, y . Project(p) $\wedge$ Experience(x) $\wedge$ Manager(y) $\wedge \neg$ SoftwareManagment(x, y) $\wedge \neg$ HandsOn(x, y) $\wedge \neg$ DevelopmentProcess(x, y) $\wedge \neg$ ApplicationDomain (x, y) $\wedge \neg$ ProgramComplexity(x, y) $\rightarrow$ ManagmentExperienceRisk(p)

Axioma da subclasse Comunicação da classe Risco de Ambiente de Trabalho

Definição: Se existe um projeto P no qual não existe boa comunicação entre seus membros, então existe risco de comunicação.

Considerando x como sendo algum membro da equipe envolvida no projeto p, temos:

$\exists$ p, x. Project(p) $\wedge$ People(x) $\wedge \neg$ GoodCommunication(x, p) $\rightarrow$ CommunicationRisk(p)

4.1.2 Lógica Descritiva

Nos anos 80 as Lógicas Descritivas com a tentativa suprir falta de definição formal adequada e semântica clara das redes semânticas proporcionando bases formais a elas. São úteis para representar praticamente todos os formalismos de representação baseados em classes como taxonomias, tesauros e bancos de dados.

Bases de conhecimento, em Lógica Descritiva, são compostas basicamente por conceitos que representam um conjunto de indivíduos, também chamados de classes, e por propriedades que representam uma relação binária entre dois indivíduos.

A modelagem da Lógica Descritiva é realizada através de uma *T-BOX (Terminological Box)* - componente intencional (atemporal e que não se modifica) - e uma *A-BOX (Assertional Box)* - componente extensional (depende de um conjunto de circustâncias). A diferença entre as duas é que a *A-BOX* refere-se ao conhecimento específico sobre indivíduos do domínio enquanto a *T-BOX* refere-se ao conhecimento na forma de uma terminologia.

Existem várias linguagens da família da Lógica Descritiva, onde cada uma tem sua particularidade e abrange uma certa quantidade de regras sintáticas. A linguagem utilizada aqui

foi a *OWL-DL* na qual é composta pela junção das regras gramaticais das linguagens S, H, I e Q, também da família da Lógica Descritiva.

4.1.3 Sub-Ontologia Engenharia de Produto da *m*PRIME *Ontology*

O início da redefinição da *Onto*PRIME se deu com o trabalho de Lins [Lins F., 2007], onde foi definida a sub-Ontologia de Engenharia de Produtos, da *m*PRIME *Ontology,* também utilizando a linguagem *OWL-DL* na ferramenta *Protégé.*

A classe Engenharia de Produtos é composta pelas atividades da Engenharia de Sistema e da Engenharia de Software envolvidas na criação de um sistema que satisfaça requisitos específicos e expectativas do cliente. Essas atividades envolvem a análise e especificação dos requisitos de sistema e de software, modelagem de software e implementação, integração de componentes de hardware e de software e testes de software e sistema.

4.2 Construção da Ontologia

Não existe uma única metodologia correta para o desenvolvimento de Ontologias, nem um único resultado correto. Desenvolver uma ontologia é, em geral, um processo interativo. Começa-se com um primeiro rascunho da ontologia. Então, essa primeira versão é refinada e detalhes são inseridos. Logo a metodologia utilizada neste trabalho utilizada para a construção da sub-Ontologia Ambiente de Desenvolvimento da *m*PRIME *Ontology* consistiu na repetição exaustiva dos seguintes passos:

- Listar possíveis classes, atributos e relações;
- Estudo detalhado dos termos e relações selecionadas, verificando o que pode ser classe ou pode fazer parte diretamente da relação transformando-se em um atributo;
- Construção da Ontologia.

A *Onto*PRIME, primeira versão da Ontologia de Riscos aqui definida, é baseada na taxonomia do *SEI* e organizada em três categorias: Engenharia de Produto, Ambiente de Desenvolvimento e Restrições de Programa.

A *m*PRIME *Ontology* é a redefinição, em Lógica Descritiva, da categoria de Ambiente de Desenvolvimento da *Onto*PRIME, que diz respeito ao ambiente do projeto onde o software será fabricado.

Para um melhor entendimento da *m*PRIME *Ontology*, algumas alterações foram realizadas na Ontologia de projetos de softwares auxiliar proposta por Lins [Lins F., 2007] em seu trabalho "Modelagem da *m*PRIME *Ontology* em Lógica Descritiva e Implementação", na qual relata a redefinição da sub-Ontologia Engenharia de Produto da *Onto*PRIME. A Figura 4-2 mostra o resultado da Ontologia auxilar de projetos de softwares:

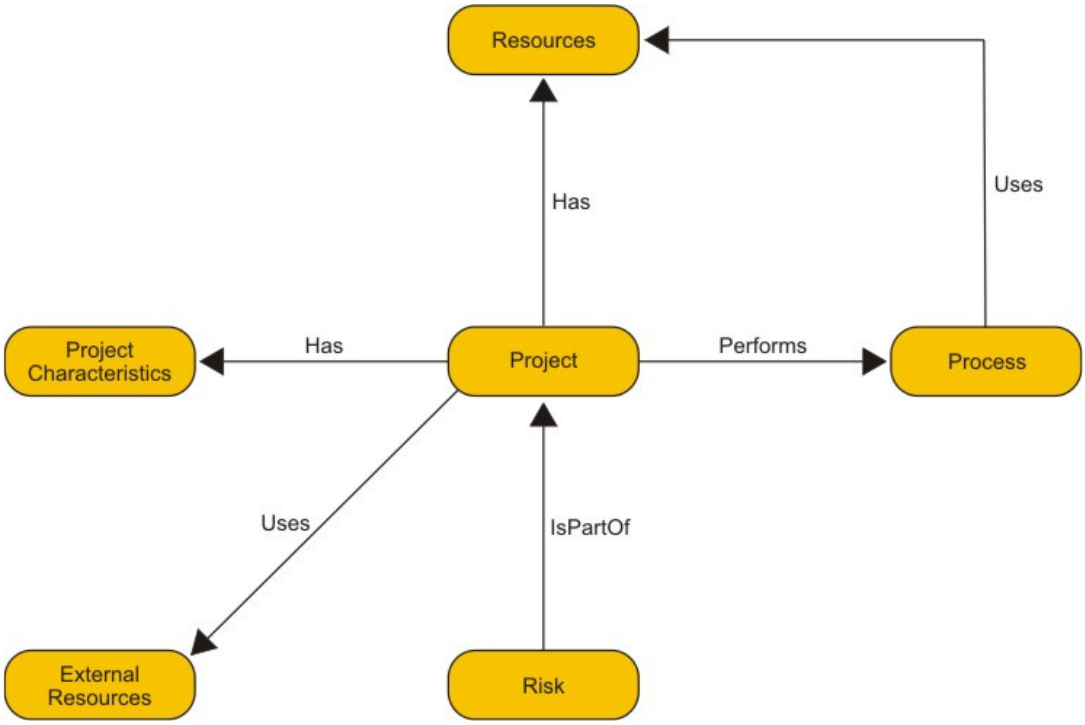

Figura 4-2. Ontologia Auxiliar de Projetos de Software Alterada de Lins [Lins F., 2007]

A Figura 4-2 mostra que os riscos fazem parte de projetos. Além disso projetos possuem recursos alocados assim com características próprias. A figura ainda mostra que o projeto utiliza recursos externos e executam processos. Entenda-se por processos como sendo as atividades organizadas em conjuntos de disciplinas nas quais são responsáveis pelos os fluxos de trabalho *(Workflows)*. O processos possuem atores (responsáveis pela interação com as atividades), utilizam técnicas e geram e consomem artefatos através dos *workflows*.

4.2.1 Sub-Ontologia Ambiente de Desenvolvimento

Este trabalho procurou definir a sub-Ontologia de Ambiente de Desenvolvimento, dando continuidade à redefinição da *Onto*PRIME, já que a sub-Ontologia Engenharia de Produto foi redefinida no trabalho de Lins.

A classe Ambiente de Desenvolvimento trata do ambiente do projeto em que o software será contruído. De acordo com a taxonomia do SEI, ela conta com os seguinte elementos:

- ***Development Process*** **–** Trata da definição, planejamento, documentação e comunicação dos métodos e procedimentos utilizados para o desenvolver o produto;

- ***Development System*** **–** Engloba as ferramentas e os equipamentos de suporte para o desenvolvimento do produto;

- ***Management Process*** **–** Planejamento, monitoramento e controle dos orçamentos e cronograma. Controla fatores envolvidos na definição, implementação e teste do produto. Engloba também a experiência do gerente em desenvolvimento de software e domínio do produto;

- ***Management Methods*** **–** Métodos**,** ferramentas e equipamentos de suporte usados para gerenciar e controlar o desenvolvimento do produto;

- ***Work Environment*** **–** Ambiente no qual são desenvolvidas as atividades de desenvolvimento. Inclui atitudes das pessoas, níveis de cooperação, comunicação e moral.

A Figura 4-3 mostra a hierarquia contruída no plugin *Owlviz* [Owlviz, 2005] do *Protégé*:

Figura 4-3. Hierarquia da Classe da Categoria Ambiente de Desenvolvimento

A Figura 4-3 mostra que a classe *Risk* é uma subclasse de *Thing*, classe padrão criada pelo Protégé, onde as classes dos usuários devem fazer parte dela. Logo depois são mostrados os elementos que constituem a classe Ambiente de Desenvolvimento, chamada de *DevelopmentEnvironmentRisk* e seus elementos e respectivos atributos conforme mostrado na Figura 2-2 - Taxonomia.

4.2.2 Redefinição dos Axiomas

Na elaboração da *m*PRIME *Ontology* alguns axiomas foram mudados, para os predicados pudessem ser definidos, pois a *Onto*PRIME apenas utiliza uma nomenclatura intuitiva com predicados indefinidos.

Para uma melhor visualização do que foi realizado veremos a redefinição dos alguns axiomas citados na seção 4.1.1 além de outros dois novos que foram introduzidos na Ontologia. A explicação de todos os axiomas tornaria o trabalho redundante e exaustivo para o leitor. A redefinição de todos os axiomas pode ser encontrada no Apêndice A, disponibilizado no final desse documento.

Redefinição do Axioma da subclasse Familiariade da classe Risco de Processo de Desenvolvimento

O início da redefinição do axioma se deu pela mudança no texto no qual o axioma trazia. O resultado pode ser verificado abaixo:

Definição: Para todo projeto P que não existe experiência, conhecimento e conforto com o processo utilizado, então existe um risco de familiaridade com processo de desenvolvimento para P.

Posteriormente foi feita a definição do axioma em *OWL-DL*, sendo necessária a definição da classe *Project* e das propriedades *experience, knowledge* e *comfort* onde representam, no domínio, experiência, conhecimento e conforto de toda equipe do projeto com o processo, respectivamente. Assim foi possível definir as seguintes subclasses de *Members*:

$TeamNotExperienceWithProcess \equiv \neg (\exists\ experience.Process)$
$TeamNotKnowledgeWithProcess \equiv \neg (\exists\ knowledge.Process)$
$TeamNotComfortWithProcess \equiv \neg (\exists\ comfort.Process)$

Logo após tais definições as classes *ProjectTeamNotExperienceWithProcess, ProjectTeamNotKnowledgeWithProcess* e *ProjectTeamNotComfortWithProcess* como subclasses de *Project* que indicam que a equipe do projeto não possue, respectivamente, experiência, conhecimento e conforto com o processo. Ainda foi necessária a criação da propriedade *holds* na qual indica que o projeto possui certa característica. Essa subclasses foram definidas da seguinte forma:

$ProjectTeamNotExperienceWithProcess \equiv \exists\ holds.TeamNotExperienceWithProcess$
$ProjectTeamNotKnowledgeWithProcess \equiv \exists\ holds.TeamNotKnowledgeWithProcess$
$ProjectTeamNotComfortWithProcess \equiv \exists\ holds.TeamNotComfortWithProcess$

Agora já é possível definir a classe *DevelopmentProcessFamiliarityRisk* na qual é subclasse de *DevelopmentProcessRisk,* e representa os riscos de familiaridade no processo de desenvolvimento, sendo necessário para essa representação a criação da propriedade *isPartOf*. Segue a definição de *DevelopmentProcessFamiliarityRisk*:

$DevelopmentProcessFamiliarityRisk \sqsupseteq (\forall isPartOf.ProjectTeamNotExperienceWithProcess) \sqcap (\forall isPartOf.ProjectTeamNotExperienceWithProcess) \sqcap (\forall isPartOf.ProjectTeamNotExperienceWithProcess)$

Redefinição do Axioma da subclasse Adequabilidade da classe Risco de Sistema de Desenvolvimento

O texto a definição do axioma foi modificado tendo seu resultado apresentado abaixo:

Definição: Para todo projeto P que possui um baixo grau de auxílio por parte dos modelos de desenvolvimento, ou processos, ou procedimentos, ou atividades requeridas e selecionadas para o programa, então existe um risco de adequabilidade de sistema de desenvolvimento para P.

Foi necessária a criação das classes *DevelopmentModel, Process, Procedures e RequiredActivities* que representam, respectivamente, os modelos de desenvolvimento, processos, procedimentos e atividades requeridas para o programa. Foi definida ainda a propriedade *lowSupportiveLevel* para representar o baixo grau de auxílio.

Após tais definições foi possível a contrução da classe *ProjectLowSupportiveLevel* como segue:

$ProjectLowSupportiveLevel \equiv (\exists lowSupportiveLevel.DevelopmentModel) \sqcup (\exists lowSupportiveLevel.Process) \sqcup (\exists lowSupportiveLevel.Procedures) \sqcup (\exists lowSupportiveLevel.RequiredActivities)$

Logo em seguida pudemos definir a classe *DevelopmentSystemSuitabilityRisk* que representa os riscos de adequabilidade do sistema de desenvolvimento do projeto e é subclasse de *DevelopmentSystemRisk*. Segue sua definição:

$DevelopmentSystemSuitabilityRisk \sqsupseteq \forall isPartOf.ProjectLowSupportiveLevel$

Redefinição do Axioma da subclasse Gerenciamento de Pessoal da classe Risco de Métodos de Gerenciamento

O axioma teve um novo texto conforme descrito abaixo:

Definição: Para todo projeto P no qual não é fácil para os membros do projeto serem gerenciados, então existe risco de gerenciamento de pessoal.

A classe *Members* foi definida para representar os membros da equipe de desenvolvimeto. Nesse ponto é importante fazer uma observação no que diz respeito a existência de outra classe, chamada *Team,* onde ela inclui toda equipe do projeto incluisive o gerente. A classe *Manager* representa os gerentes de projetos. Logo as classes *Members* e *Manager*, são atores do que

interagem com o projeto, assim como outras classes que podem ser visualizadas na hierarquia mostrada na Figura 4-4:

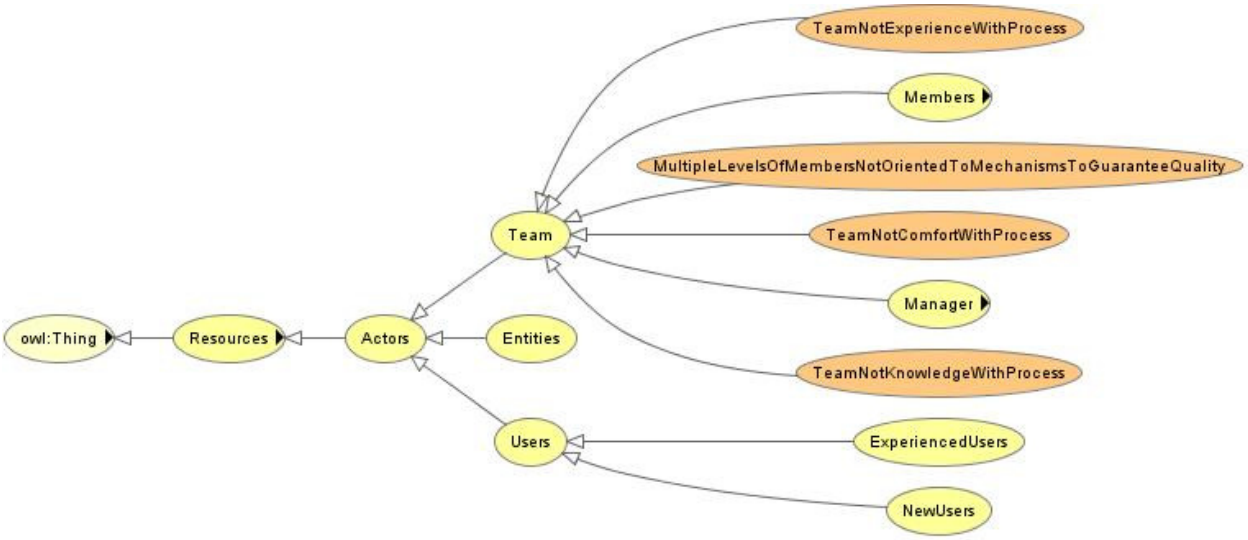

Figura 4-4. Hierarquia da Classe *Actors*

A classe *ProjectMembersNotEasyManaged* foi contruída como subclasse de *Project* e utiliza a propriedade *easyManaged* para representar o fácil gerenciamento por parte dos membros, nesse caso. A seguir temos a definição da classe *ProjectMembersNotEasyManaged*:

ProjectMembersNotEasyManaged ≡ ¬ *(∃easyManaged.Members)*

A seguir foi possível a definição da classe *PersonnelManagementRisk* que representa os riscos de gerenciamento de pessoal e que é subclasse de *ManagementMethodsRisk*. Lembramos que algumas classes, como *PersonnelManagementRisk*, possuem outros axiomas em que as define. As definições das classes com todos os axiomas podem ser vistos no final deste documento no Apêndice A. A classe *PersonnelManagementRisk* foi definida como segue:

PersonnelManagementRisk ⊒ *∀isPartOf.ProjectMembersNotEasyManaged*

Redefinição do Axioma da subclasse Experiência da classe Risco de Processo de Gerenciamento

O texto foi redefinido sendo mostrado abaixo:

Definição: Para todo projeto P em que os gerentes não sejam experientes, então existe risco de experiência de gerenciamento.

A definição do axioma se deu início com a classe *ProjectManagerNotHasExperience*, onde modela a característica do projeto de possuir gerentes que não são experientes e é subclasse de *Project*. Para definição dessa classe foi necessária a utilização da classe *Manager* e a propriedade *experience*, ambos definidos anteriormente. A definição da classe *ProjectManagerNotHasExperience* pode ser vista a seguir:

ProjectManagerNotHasExperience ≡ ¬ *(∃experience.Manager)*

Agora já é possível definir a classe *ExperienceRisk*, que é subclasse de *ManagementProcessRisk*, visualizada a seguir:

ExperienceRisk ⊒ ∀*isPartOf.ProjectManagerNotHasExperience*

Redefinição do Axioma da subclasse Comunicação da classe Risco de Ambiente de Trabalho

A redefinição do texto do axioma ficou como podemos verificar abaixo:

Definição: Para todo projeto P em que não existe boa comunicação entre seus membros, então existe risco de comunicação.

A classe *ProjectNotExistsGoodCommunicationBetweenMembers* foi criada com o objetivo de modelar projetos que não possuem boa comunicação entre os membros. Para isso foi necessário a criação da classe *Communication* que representa a comunicação do projeto e é subclasse *ManagementOfCommunicationsProject*. Essa última classe é responsável por todas atividades de comunicação do projeto envolvidas nos processos de desenvolvimento.

Além da classe *Communication*, foi necessário a representação do adjetivo "bom" ou "boa" através da propriedade *good*. A classe *ProjectNotExistsGoodCommunicationBetweenMembers* está definida no *Protégé* da seguinte forma:

ProjectNotExistsGoodCommunicationBetweenMembers ≡ ¬ *(∃good.Communication)*

Posteriormente foi definida a classe CommunicationRisk, subclasse de WorkEnvironmentRisk e que representa os riscos de comunicação do projeto. A definição ficou da seguinte forma:

WorkEnvironmentRisk ⊒ ∀ *isPartOf.ProjectNotExistsGoodCommunicationBetweenMembers*

Como dito anteriormente, dois axiomas novos foram introduzidos na Ontologia com o objetivo de dar maior poder de identificação de riscos. A definição dos mesmos pode ser vista a seguir.

Definição do Axioma da subclasse Formalidade da classe Risco de Processo de Desenvolvimento

Definição: Para todo projeto P em que os atores não estão definidos, então existe risco de formalidade de processo de desenvolvimento.

Inicialmente foi criada a classe ProjectActorsNotDefined que define um projeto que possui como característica a não-definição dos atores do processo. Assim foi necessário a criação de uma classe maior que englobasse as classes Team, Members e Manager, e obedecesse a hierarquia idealizada com essas classes.

Dessa forma criou-se as classes Entities e Users, onde representam sistemas ou módulos de sistemas, dentre outros fatores não-humanos, que podem interagir de alguma forma com o

processo de desenvolvimento e usuários do sistema de desenvolvimento. A hierarquia completa pode ser consultada na Figura 4-4.

A propriedade defined foi criada para representar a definição de atores, mecanismos de qualidade ou processos. Aqui a utilizaremos aplicando a atores.

Voltando a nossa classe ProjectActorsNotDefined, a mesma ficou definida da seguinte forma:

$ProjectActorsNotDefined \equiv \neg (\exists defined.Actors)$

Posteriormente foi possível definirmos a classe DevelopmentProcessFormalityRisk que é subclasse de DevelopmentEnvironmentRisk e representa os riscos de formalidade do processo de desenvolvimento da seguinte forma:

$DevelopmentProcessFormalityRisk \sqsupseteq \forall isPartOf.\ ProjectActorsNotDefined$

Definição do Axioma da subclasse Formalidade da classe Risco de Processo de Desenvolvimento

Definição: Para todo projeto P que não possui métricas de qualidade não definidas, existe risco de controle de qualidade.

Partimos da definição da classe ProjectNotDefinedMetricsOfQuality, onde a mesma é subclasse de Project e utiliza a classe MetricsOfQuality, que representa um mecanismo para garantir a qualidade. Ela também utiliza a propriedade defined, explicada na definição do axioma anterior.

$ProjectNotDefinedMetricsOfQuality \equiv \neg (\exists defined.\ MetricsOfQuality)$

Após a definição da classe acima, definimos a classe QualityAssuranceRisk, outra sibclasse de ManagementMethodsRisk:

$QualityAssuranceRisk \sqsupseteq \forall isPartOf.ProjectNotDefinedMetricsOfQuality$

4.3 Exemplo Prático

Como um exemplo prático de melhoria da utilização da Ontologia de riscos aqui proposta, poderíamos citar a aplicação dos conceitos definidos na Ontologia em relação às atividades treinamento. Dessa forma os riscos correspondentes a tais atividades poderão ser identificados mais rapidamente em relação à outros métodos, já que existe uma definição formal dos fatores que poderão gerar esses riscos.

Consideremos os riscos relacionados ao sistema de desenvolvimento, mais especificamente os riscos de familiaridade com o sistema de desenvolvimento e *ProjectNotExistsTrainingAdequateForNewUsersOnDevelopmentSystem* sendo a característica de algum projeto não possuir um treinamento adequado para novos usuários do sistema de desenvolvimento e que foi definido da seguinte forma em OWL-DL:

ProjectNotExistsTrainingAdequateForNewUsersOnDevelopmentSystem ≡ ¬ (∃ *training.NewUsers)*

Agora é possível definirmos um dos fatores que identifica o risco de familiaridade com o sistema de desenvolvimento da seguinte forma:

DevelopmentSystemFamiliarityRisk ⊒ ∀ *isPartOf.ProjectNotExistsTrainingAdequateForNewUsersOnDevelopmentSystem*

Dessa forma para que tal risco seja identificado é necessário aplicamos o conceito definido acima ao projeto em que se deseja saber se existe a possibilidade do risco ocorrer. Assim, podemos propor a seguinte pergunta: O projeto possui treinamento adequado para novos usuários do sistema de desenvolvimento? Em caso afirmativo o risco pode ocorrer.

Lembramos que tal facilidade de identificação só é possível por conta de uma prévia conceitualização de um vocabulário e a construção de uma hierarquia, relacionados ao Ambiente de Desenvolvimento de Múltiplos Projetos, que foram realizadas através da implementação de uma Ontologia, a *m*PRIME *Ontology*.

4.4 Resumo do Capítulo

No início do capítulo foi realizado um tratamento das premissas necessárias para a realização deste trabalho. Dentre elas destacamos a *Onto*PRIME (Ontologia implementada em Lógica de Primeira Ordem), Lógica Descritiva (compostas basicamente por conceitos que representam um conjunto de indivíduos e propriedades que representam uma relação binária entre dois indivíduos) e a sub-Ontologia Engenharia de Produto da *m*PRIME *Ontology*, onde se deu o início da redefinição da *Onto*PRIME com o trabalho de Lins.

A seguir, na seção 4.2, pelo fato de não existir uma única metodologia correta para o desenvolvimento de Ontologias, foi apresentada a metodologia realizada para a construção da sub-Ontologia de Ambiente de Desenvolvimento, redefinida neste trabalho em Lógica Descritiva.

Por fim, na seção 4.2.2, mostramos algumas redefinições dos axiomas apresentados na seção 4.1.1 assim como dois novos axiomas. Assim, para um entendimento de todo traballho, a redefinição completa dos axiomas da sub-Ontologia de Ambiente de Desenvolvimento pode ser consultada no final deste documento no Apêndice A.

Capítulo 5

Conclusão

O trabalho realizado procurou redefinir uma Ontologia de riscos em um Ambiente de Desenvolvimento de Múltiplos Projetos, chamada *Onto*PRIME, onde passou a se chamar, em sua nova versão, de *m*PRIME *Ontology*. A *Onto*PRIME foi definida em Lógica de Primeira Ordem e a *m*PRIME *Ontology* em Lógica Descritiva, através da linguagem *OWL-DL*.

O motivo de tal redefinição se deve ao fato da *OWL-DL* ser uma linguagem que vem se destacando na elaboração de Ontologias. A *m*PRIME *Ontology* tem por objetivo fornecer um vocabulário comum que poderá ser utilizado para representar conhecimento útil para os desenvolvedores de software sobre os riscos e oportunidades que podem afetar um projeto de software em um ambiente organizacional. Um outro motivo relevante, pode ser atribuído à expressividade na qual a *OWL-DL* pôde oferecer à *m*PRIME *Ontology*.

A taxonomia de riscos do SEI é contituída de três categorias: Engenharia de Produto, Ambiente de Desenvolvimento e Restrições de Programa. Este trabalho se limitou a redefinir a categoria Ambiente de Desenvolvimento, pois a categoria Engenharia de Produto encontra-se já definida no trabalho de Lins e além disso o escopo do trabalho aumentaria de forma considerável.

A ferramenta utilizada no trabalho foi o *Protégé v3.3.1*, um ambiente para criação e edição de ontologias e bases de conhecimento e que fornece suporte à linguagem *OWL-DL* sendo possível a definição de Ontologias de forma rápida e intuitiva. No trabalho também foi necessário a utilização do *Graphviz v2.17* [Graphviz, 2007] e do plugin do *Protégé*, o *Owlviz*, onde intergrados deram suporte à geração de alguns gráficos utilizados neste trabalho.

Esperamos que mais projetos de software tenham sucesso através da execução efetiva da atividade de Gerência de Riscos com auxílio da proposta da *m*PRIME *Ontology* de identificação de riscos em Ambientes de Desenvolvimento de Múltiplos Projetos.

5.1 Trabalhos Relacionados

Existem alguns trabalhos relacionados com o presente trabalho. Podemos citar dois trabalhos de Fabiano B. O primeiro, chamado de *"Learning How to Manage Risks Using Organizational Knowledge",* apresenta uma ferramenta chamada GeRis [Fabiano B. et al, 2004a], que traz uma abordagem sobre o gerenciamento de risco através do conhecimento.

O segundo trabalho de Fabiano B., chamado de "Ontologias e Ambientes de Desenvolvimento de Software Semânticos" [Fabiano B. et al, 2004b], discute como ontologias têm sido utilizadas no ambiente de desenvolvimento de software, já que, assim como na *Web,* o excesso de recursos de informação, associado à falta de semântica para guiar uma busca por recursos realmente relevantes para o contexto em mãos tendo em vista a que o desenvolvimento de software é uma tarefa de conhecimento intenso, na qual muitos recursos de informação são produzidos e utilizados.

O trabalho mais importante para a realização deste foi o desenvolvido por Gusmão [Gusmão et al, 2005], onde traz a definição da *Onto*PRIME, uma Ontologia de riscos para Ambientes de Desenvolvimento de Múltiplos Projetos, que foi desenvolvida em Lógica de Primeira Ordem. O presente trabalho trata da redefinição de da sub-Ontologia Ambiente de Desenvolvimento da *Onto*PRIME, dessa vez em Lógica Descritiva, na qual foi chamada nessa nova versão de *m*PRIME *Ontology*.

Um outro trabalho importante relacionado com este é o de Lins [Lins F., 2007]. Ele tratou de redefinir uma outra sub-Ontologia da *Onto*PRIME, a de Engenharia de Produto, também usando Lógica Descritiva e a ferramenta Protégé [Protégé, 2007]. Portanto o presente trabalho tratou de dar continuidade ao trabalho de Lins, completando mais uma parte da redefinição da *Onto*PRIME.

5.2 Dificuldades Encontradas

Para a realização desde trabalho algumas dificuldades foram encontradas. Dentre elas podemos destacar a falta de Ontologias na literatura referente a riscos em ambientes de desenvolvimento. Desta forma *m*PRIME *Ontology*, se baseou exclusivamente na *Onto*PRIME.

Outro problema encontrado foi a falta de material em português sobre as ferramentas utilizadas neste trabalho, o *Protégé*, *Owlviz* e *Graphviz*.

Podemos ainda citar a ocorrêcia de um *"Bug"*na ferramenta *Protégé*, onde, algumas vezes, todo trabalho poderia ser perdido, caso não estivesse sido salvo, ao utilizarmos a opção *Search*, pois a ferramenta travava sendo necessária a reinicialização na mesma.

5.3 Trabalhos Futuros

Algumas propostas de trabalhos futuros podem ser indicadas aqui com intuito de um maior proveito para a melhoria da Ontologia. A primeira dela trata da continuidade da redefinição da *Onto*PRIME, através da implementação da terceira e última sub-Ontologia, segundo a taxonomia

do *SEI*, a de Restrições de Programa, seguindo o mesmo padrão utilizado nas redefinição das outras sub-Ontologias.

Um outro futuro trabalho se baseia em uma análise mais profunda da sub-Ontologia de Ambiente de Desenvolvimento descrita no Apêndice A com o objetivo de dar mais consistência e abordar de forma mais precisa os riscos em Ambientes de Desenvolvimento de Múltiplos Projetos.

Por fim, após a redefinição da terceira sub-Ontologia de Restrições de Programa, pode ser realizada a junção das três sub-Ontologias. Para tal atividade é necessário ser utilizado um *Reason*, uma máquina de raciocínio, capaz de verificar a consistência da hierarquia construída na Ontologia.

Referências Bibliográficas

[Alex Osborn, 1930] – Alez Orborn & Brainstorming – Disponível em: http://www.ciadvertising.org/studies/student/97_fall/practitioner/osborn/afosborn.htm Acesso em: 16.10.2007.

[Bernstein, 1997] Bernstein, P. Desafio ao Deuses: a fascinante história do risco. Rio de Janeiro: Campus. 1997.

[Boehm, 1991]Boehm, B. W. (1991) Software Risk Management: Principles and Practices

[Borst, 1997] Borst, W. N. Construction of engineering ontologies. 1997. Tese (Doutorado). Disponível em: http://www.ub.utwente.nl/webdocs/inf/1/t0000004.pdf. Acesso em: 10/11/2007

[Carr M. et al, 1993] Carr M. et al (1993) Taxonomy Based Risk Identification. Technical Report CMU/SEI-93-TR-6. Pittsburgh, PA: Software Ebgineering Institute, Carnegie Mellon University, USA.

[Fabiano B. et al, 2004a] Fabiano B. Ruy, Ricardo A. Falbo, Gleidson Bertollo, Denise F. Togneri "Learning How to Manage Risks Using Organizational Knowledge" - Universidade Federal do Espírito Santo, Vitória - ES – Brasil.

[Fabiano B. et al, 2004b] Fabiano B. Ruy, Ricardo A. Falbo, Juliana Pezzin, Rodrigo Dal Moro "Ontologias e Ambientes de Desenvolvimento de Software Semânticos" - Universidade Federal do Espírito Santo, Vitória - ES – Brasil.

[Gomez & Perez, 1999] Gómez-Pérez, A. and V.R. Benjamins, Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods in International Joint Conference on Artificial Intelligence(IJCAI-99), Workshop on Ontologies and Problem-Solving Methods (KRR5), V.R. Benjamins, et al., Editors. 1999: Stockolm, Sweden.

[Graphviz, 2007] Disponível em: http://www.graphviz.org/ Acesso em: 12.10.2007

[Gruber, 1996] GRUBER, T. What Is An Ontology? 1996. Disponível em: http://www.ksl.stanford.edu/kst/what-is-an-ontology.html Acesso em: 20/11/2007

[Guarino, 2001] Guarino, N. Proceedings of the International Conference on Formal Ontology in Information Systems, 2001, Buffalo. Nova York: Barry Smith University at Buffalo, 2001.

[Gusmão, 2007] Gusmão, C. (2007) Um Modelo de Processo de Gestão de Riscos para Ambientes de Múltiplos Projetos de Desenvolvimento de Software. Tese de Doutorado. Universidade Federal de Pernambuco. Recife – PE, Brasil.

[Gusmão & Moura, 2005] Gusmão, C.M.G.; Moura, H. P. (2005b) Gestão de Riscos para Ambientes de Múltiplos Projetos de Software: Teoria e Prática. In: IV ERI MG – Escola Regional de Informática de Minas Gerais. ISBN 85-7669-048-9

[Gusmão et al, 2004] Campello, A.; Gusmão, C.; Amorim, L. ; Guedes, M.; Monteiro, M.; OntoPRIME: Ontologia de Riscos para Ambientes de Desenvolvimento de Software Multiprojetos, Universidade Federal de Pernambuco, Recife, Brasil. (2004).

[J. Davies et al, 2003] J. Davies, D. Fensel, F. van Harmelen, "Towards The Semantic Web: Ontology-Driven Knowledge Management", John Wiley & Sons Ltd, 2003.

[Knight, 1921] Knight, F.H. (1921) Risk, Uncertainty and Profit. Houghton Miffin, Boston.

[Larousse, 1998] Le Petit Larousse Compact. (1998) Édition Entièrement Nouvelle

[Lins F., 2007] Lins F. (2007) Modelagem da Mprime Ontology em Lógica Descritiva e Implementação, Trabalho de Graduação - Universidade Federal de Pernambuco. Recife – PE, Brasil.

[Owlviz, 2005] Disponível em: http://www.co-ode.org/downloads/owlviz/ Acesso em: 10.10.2007

[PMBOK, 2004] Guia PMBOK®, Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos, Terceira edição.

[PMI, 2004] PMI - Project Management Institute. (2004) A Guide to the Project Management Body of Knowledge. – ANSI/PMI 99-01-2004. Project Management Institute. Four Campus Boulevard. Newtown Square. USA.

[Pressman, 1995] Pressman, R. S. (1995) Engenharia de Software. São Paulo: Ed. Makron Books.

[Protégé, 2007] Protégé. Disponível em: http://protege.stanford.edu/ Acesso em: 20.09.2007

[SEI, 2003] SEI - Software Engineering Institute (2003) Taxonomy - Based Risk Identification, Carnegie Mellon University - Pittsburgh, Pennsylvania.

[T. Berners-Lee et al, 2001] T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web", Scientific American, May 2001.
[Turoff and Linstone, 1975] Turoff, Murray; Linstone, Harold A. The Delphi method. New York: Addison Wesley Publishing Company Inc., 1975.

[Victoria et al, 2000] Victoria, C. G. et al. (2000) Pesquisa Qualitativa em Saúde: Uma Introdução ao Tema. Porto Alegre: Tomo Editorial.

[W3C, 2004] Web Ontology Language Guide. Disponível em: http://www.w3.org/TR/owl-guide/ Acesso em: 21.11.2007.

Apêndice A

Definição dos Axiomas da Sub-Ontologia Ambientes de Desenvolvimento da mPRIME Ontology

1. Definição dos Axiomas da Classe Riscos de Processo de Desenvolvimento – *DevelopmentProcessRisk*

A hierarquia da classe *DevelopmentProcessRisk*, subclasse de *DevelopmentEnvironmentRisk*, pode ser vista a seguir na Figura A-1:

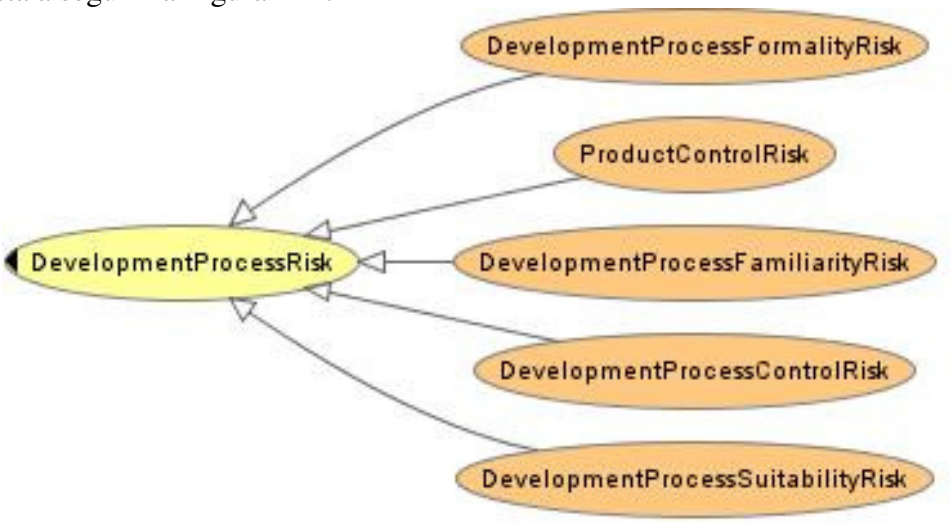

Figura A-1. Hierarquia da Classe *DevelopmentProcessRisk*

1.1. Axiomas da subclasse Controle do Processo – ***DevelopmentProcessControlRisk***

1.1.1. Para todo projeto P em que não há garantia dos processos definidos então existe um risco de controle do processo de desenvolvimento.

$DevelopmentProcessControlRisk \sqsupseteq \forall isPartOf.\ ProjectProcessNotGuaranteed$
$ProjectProcessNotGuaranteed \equiv \neg\ (\exists guaranteed.\ Process)$

1.1.2. Para todo projeto P no qual existe um processo que não é dimensionado e aperfeiçoado, então existe um risco de controle do processo de desenvolvimento.

$DevelopmentProcessControlRisk \sqsupseteq (\forall isPartOf.\ ProjectProcessNotImprovement) \sqcap (\forall isPartOf.ProjectProcessNotMeasurement)$
$ProjectProcessNotImprovement \equiv \neg(\exists improvement.Process)$
$ProjectProcessNotMeasurement \equiv \neg(\exists measurement.Process)$

1.2. Axiomas da subclasse Familiaridade – ***DevelopmentProcessFamiliarityRisk***

1.2.1. Para todo projeto P em que não há experiência, conhecimento e conforto com o processo utilizado, então existe um risco de familiaridade com processo de desenvolvimento.

$DevelopmentProcessFamiliarityRisk \sqsupseteq (\forall isPartOf.\ ProjectTeamNotExperienceWithProcess) \sqcap (\forall isPartOf.\ ProjectTeamNotKnowledgeWithProcess) \sqcap (\forall isPartOf.\ ProjectTeamNotComfortWithProcess)$
$ProjectTeamNotExperienceWithProcess \equiv \exists holds.TeamNotExperienceWithProcess$
$TeamNotExperienceWithProcess \equiv \neg(\exists experience.Process)$
$ProjectTeamNotKnowledgeWithProcess \equiv \exists holds.TeamNotKnowledgeWithProcess$
$TeamNotKnowledgeWithProcess \equiv \neg(\exists knowledge.Process)$
$ProjectTeamNotComfortWithProcess \equiv \exists holds.TeamNotComfortWithProcess$
$TeamNotComfortWithProcess \equiv \neg(\exists comfort.Process)$

1.3. Axiomas da subclasse Formalidade – ***DevelopmentProcessFormalityRisk***

1.3.1. Para todo projeto P no qual um processo não é bem definido, então existe um risco de formalidade de processo de desenvolvimento.

$DevelopmentProcessFormalityRisk \sqsupseteq \forall isPartOf.ProjectProcessNotCommunicatedForAllAspectsAndPhase$
$ProjectProcessNotCommunicatedForAllAspectsAndPhases \equiv \exists communicated.Process$

1.3.2. Para todo projeto P no qual um processo não é bem documentado, então existe um risco de formalidade de processo de desenvolvimento.

$DevelopmentProcessFormalityRisk \sqsupseteq \forall isPartOf.\ ProjectProcessNotDocumented$
$ProjectProcessNotDocumented \equiv \neg(\exists documented.Process)$

1.3.3. Para todo projeto P no qual um processo não é comunicado para todos os aspectos e fases do desenvolvimento, então existe um risco de formalidade de processo de desenvolvimento.

$DevelopmentProcessFormalityRisk \sqsupseteq \forall isPartOf.\ ProjectProcessNotDefined$
$ProjectProcessNotDefined \equiv \neg(\exists defined.Process)$

1.3.4. Para todo pojeto P em que os atores do não estão definidos, então existe risco de formalidade de processo de desenvolvimento.

$DevelopmentProcessFormalityRisk \sqsupseteq \forall isPartOf.ProjectActorsNotDefined$
$ProjectActorsNotDefined \equiv \neg(\exists defined.Actors)$

1.4. Axiomas da subclasse Adequabilidade – ***DevelopmentProcessSuitabilityRisk***

1.4.1. Para todo projeto P no qual o modelo de desenvolvimento, processo, métodos, e ferramentas selecionadas não suportam o escopo e tipo de atividades requeridas, então existe um risco de adequabilidade de processo de desenvolvimento.

$DevelopmentProcessSuitabilityRisk \sqsupseteq (\forall isPartOf.ProjectDevelopmentModelProcessMethodsDevelopmentToolsNotSupportsScope) \sqcap (\forall isPartOf.ProjectDevelopmentModelProcessMethodsDevelopmentToolsNotSupportsTypesOfRequiredActivities)$
$ProjectDevelopmentModelProcessMethodsDevelopmentToolsNotSupportsScope \equiv \neg(\exists supportsScope.DevelopmentModel) \sqcap \neg(\exists supportsScope.Process) \sqcap \neg(\exists supportsScope.Methods) \sqcap \neg(\exists supportsScope.DevelopmentTools)$
$ProjectDevelopmentModelProcessMethodsDevelopmentToolsNotSupportsTypesOfRequiredActivities \equiv \neg(\exists supportsRequiredActivities.DevelopmentModel) \sqcap \neg(\exists supportsRequiredActivities.Process) \sqcap \neg(\exists supportsRequiredActivities.Methods) \sqcap \neg(\exists supportsRequiredActivities.DevelopmentTools)$

1.5. Axiomas da subclasse Controle do Produto – ***ProductControlRisk***

1.5.1. Para todo projeto P no qual não é possível rastrear os requisitos desde sua especificação, passando pela implementação dos mesmos até os testes, então existe um risco de controle do produto

$ProductControlRisk \sqsupseteq \forall isPartOf.\ ProjectRequirementsNotTraced$
$ProjectRequirementsNotTraced \equiv \neg(\exists traced.Requirements)$

2. **<u>Definição dos Axiomas da Classe Riscos de Sistema de Desenvolvimento – *DevelopmentSystemRisk*</u>**

A hierarquia da classe *DevelopmentSystemRisk*, subclasse de *DevelopmentEnvironmentRisk*, pode ser vista a seguir na Figura A-2:

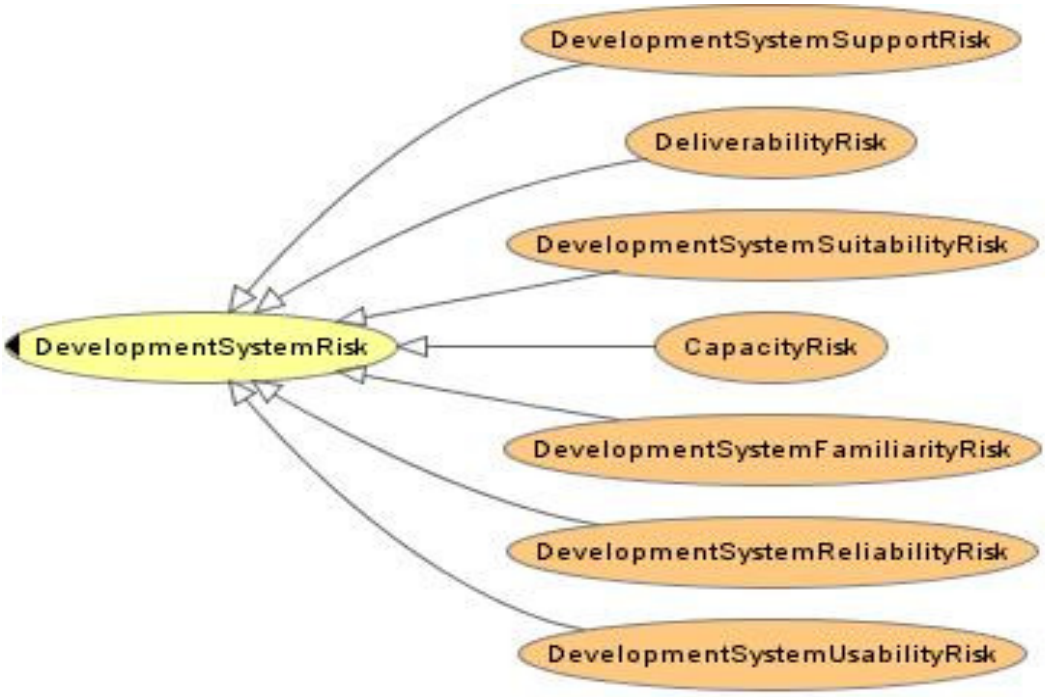

Figura A-2. Hierarquia da Classe *DevelopmentSystemRisk*

2.1. Axiomas da subclasse Capacidade – ***CapacityRisk***

2.1.1. Para todo projeto P que possui poucas estações de desenvolvimento, então existe um risco de capacidade de sistema de desenvolvimento.

$CapacityRisk \sqsupseteq \forall isPartOf.\ ProjectFewWorkstations$
$ProjectFewWorkstations \equiv \exists few.Computer$

2.1.2. Para todo projeto P que possui máquinas com pouco poder de processamento, então existe um risco de capacidade de sistema de desenvolvimento.

$CapacityRisk \sqsupseteq \forall isPartOf.\ ProjectComputerLowPower$
$ProjectComputerLowPower \equiv \exists lowPower.Computer$

2.1.3. Para todo projeto P no qual o banco de dados utilizado possui armazenamento insuficiente, então existe um risco de capacidade de sistema de desenvolvimento.

$CapacityRisk \sqsupseteq \forall isPartOf.\ ProjectDataBaseInsufficientStorage$
$ProjectDataBaseInsufficientStorage \equiv \exists insufficient.DataBaseSize$

2.1.4. Para todo projeto P que possui equipamentos inadequados para suportar atividades paralelas para desenvolvimento, testes e atividades de suporte, então existe um risco de capacidade de sistema de desenvolvimento.

CapacityRisk ⊒ ∀isPartOf. ProjectEquipmentInadequateForParallelActivities
ProjectEquipmentInadequateForParallelActivities ≡ ¬(∃ adequate.EquipmentToSuppotParallelActivities)

2.2. Axiomas da subclasse Entrega – ***DeliverabilityRisk***

2.2.1. Para todo projeto P no qual não são alocados os recursos necessários para satisfazer os requisitos de entrega, então existe um risco de entrega.

DeliverabilityRisk ⊒ ∀isPartOf.ProjectResourcesNotAllocatedToSatisfyRequirementsDelivery
ProjectResourcesNotAllocatedToSatisfyRequirementsDelivery ≡ ∃holds. DeliveryRequirementsNotSatisfied
DeliveryRequirementsNotSatisfied ≡ ¬(∃allocatedAdequate.Resources)

2.3. Axiomas da subclasse Familiaridade – ***DevelopmentSystemFamiliarityRisk***

2.3.1. Para todo projeto P no qual o sistema de desenvolvimentonão foi previamente utilizado pela companhia ou pelo pessoal do projeto, então existe um risco de familiaridade de sistema de desenvolvimento.

DevelopmentSystemFamiliarityRisk ⊒(∀isPartOf. ProjectHoldsDevelopmentSystemNeverUserByCompany) ⨆(∀isPartOf. ProjectHoldsDevelopmentSystemNeverUserByMembers)
ProjectHoldsDevelopmentSystemNeverUserByCompany ≡ ∃holds. DevelopmentSystemNeverUserByCompany
DevelopmentSystemNeverUserByCompany ≡ ∃neverUsed.Company
ProjectHoldsDevelopmentSystemNeverUserByMembers ≡ ∃holds. DevelopmentSystemNeverUserByMembers
DevelopmentSystemNeverUserByMembers≡ ∃neverUsed.Members

2.3.2. Para todo projeto P no qual não foi realizado um treinamento adequado para os novos usuários sobre o sistema de desenvolvimento a ser utilizado, então existe um risco de familiaridade de sistema de desenvolvimento.

DevelopmentSystemFamiliarityRisk ⊒ ∀isPartOf. ProjectNotExistsTrainingAdequateForNewUsersOnDevelopmentSystem
ProjectNotExistsTrainingAdequateForNewUsersOnDevelopmentSystem ≡ ¬(∃ *training.NewUsers)*

2.4. Axioma da subclasse Confiabilidade – ***DevelopmentSystemReliabilityRisk***

2.4.1. Para todo projeto P onde os componentes do sistema de desenvolvimento não estejam disponíveis, então existe um risco de confiabilidade de sistema de desenvolvimento.

DevelopmentSystemReliabilityRisk ⊒ ∀ isPartOf.ProjectDevelopmentSystemComponentsNotAvailable
ProjectDevelopmentSystemComponentsNotAvailable ≡ ∃ available.DevelopmentSystemComponents

2.4.2. Para todo projeto P onde os componentes do sistema de desenvolvimento não estejam funcionando corretamente, então existe um risco de confiabilidade de istema de desenvolvimento.

DevelopmentSystemReliabilityRisk ⊒ ∀ isPartOf.ProjectDevelopmentSystemComponentsNotWorkingCorrectly
ProjectDevelopmentSystemComponentsNotWorkingCorrectly ≡ ∃ *workCorrectly.DevelopmentSystemComponents*

2.5. Axiomas da subclasse Adequabilidade – ***DevelopmentSystemSuitabilityRisk***

2.5.1. Para todo projeto P que possui um baixo grau de auxílio por parte dos modelos de desenvolvimento, ou processos, ou procedimentos, ou atividades requeridas e selecionadas para o programa, então existe um risco de adequabilidade de sistema e desenvolvimento.

DevelopmentSystemSuitabilityRisk ⊒ ∀isPartOf.ProjectLowSupportiveLevel
ProjectLowSupportiveLevel ≡(∃lowSupportiveLevel.DevelopmentModel) ⨅ (∃ lowSupportiveLevel.Process) ⨅(∃lowSupportiveLevel.Procedures) ⨅(∃lowSupportiveLevel. RequiredActivities)

2.6. Axiomas da subclasse Suporte do Sistema – ***DevelopmentSystemSupportRisk***

2.6.1. Para todo projeto P que não envolveu um treinamento na utilização do sistema de desenvolvimento, então existe um risco de suporte do sistema de desenvolvimento.

DevelopmentSystemSupportRisk ⊒ ∀ isPartOf.ProjectMembersNotTrainingOnDevelopmentSystem
ProjectMembersNotTrainingOnDevelopmentSystem ≡ ∃ holds.MembersNotTrainingOnDevelopmentSystem
MembersNotTrainingOnDevelopmentSystem ≡ ¬(∃training.DevelopmentSystem)

2.6.2. Para todo projeto P com algum sistema de desenvolvimento que não oferece acesso a usuários experientes, então existe um risco de suporte do sistema de desenvolvimento.

DevelopmentSystemSupportRisk ⊒ ∀ isPartOf.ProjectHoldsDevelopmentSystemNotOffersAccessForExperiencedUsers
ProjectHoldsDevelopmentSystemNotOffersAccessForExperiencedUsers ≡ ∃ holds.DevelopmentSystemNotOffersAccessForExperiencedUsers
DevelopmentSystemNotOffersAccessForExperiencedUsers ≡ ¬(∃accessible.ExperiencedUsers)

2.6.3. Para todo projeto P com algum sistema de desenvolvimento que não possui uma resolução adequada dos problemas, então existe um risco de suporte do sistema de desenvolvimento.

DevelopmentSystemSupportRisk $\sqsupseteq \forall$ *isPartOf.ProjectDevelopmentSystemNotAdequateResolution*
ProjectDevelopmentSystemNotAdequateResolution $\equiv \neg$ ($\exists$ *adequateResolution.DevelopmentSystem)*

2.7. Axiomas da subclasse Usabilidade – ***DevelopmentSystemUsabilityRisk***

2.7.1. Para todo projeto P cujo sistema de desenvolvimento não possui documentação ou não é acessível, então existe um risco de usabilidade de sistema de desenvolvimento.

DevelopmentSystemUsabilityRisk $\sqsupseteq$ ($\forall$*isPartOf.ProjectDevelopmentSystemNotDocumented)* $\sqcup$ ($\forall$*isPartOf.ProjectDevelopmentSystemNotAccessible)*
ProjectDevelopmentSystemNotDocumented $\equiv \neg$ ($\exists$*documented.DevelopmentSystem)*
ProjectDevelopmentSystemNotAccessible $\equiv \neg$ ($\exists$*accecible.DevelopmentSystem)*

2.7.2. Para todo projeto P cujo sistema de desenvolvimento não é fácil de usar, então existe um risco de usabilidade de sistema de desenvolvimento.

DevelopmentSystemUsabilityRisk $\sqsupseteq \forall$*isPartOf. ProjectDevelopmentSystemNotEasyUse*
ProjectDevelopmentSystemNotEasyUse $\equiv \neg$ ($\exists$*easyUse.DevelopmentSystem)*

3. Definição dos Axiomas da Classe Riscos de Métodos de Gerenciamento – *ManagementMethodsRisk*

A hierarquia da classe *ManagementMethodsRisk*, subclasse de *DevelopmentEnvironmentRisk*, pode ser vista a seguir na Figura A-3:

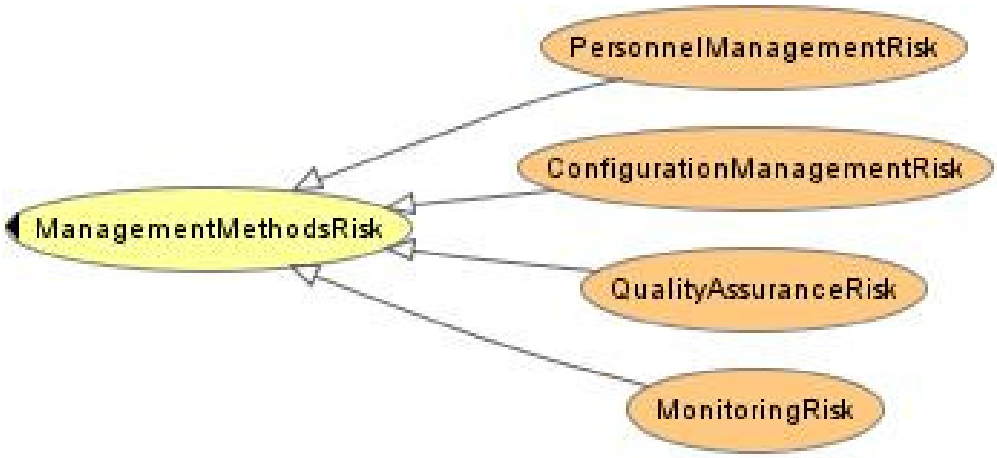

Figura A-3. Hierarquia da Classe *ManagementMethodsRisk*

3.1. Axiomas da subclasse Gerência de Configuração – ***ConfigurationManagementRisk***

3.1.1. Para todo projeto P que esta em múltiplas localidades e o sistema de gerenciamento de configuração não prove suporte a múltiplas localidades, então existe risco de gerência de configuração.

ConfigurationManagementRisk ⊒(∀isPartOf.ProjectExistsInMultipleSites) ⊓(∀ isPartOf.ProjectConfigurationManagementSystemNotSupportsMultipleSites)
ProjectExistsInMultipleSites ≡ ∃exists.MultipleSites
ProjectConfigurationManagementSystemNotSupportsMultipleSites ≡ ∃holds. ConfigurationManagementSystemNotSupportsMultipleSites
ConfigurationManagementSystemNotSupportsMultipleSites ≡ ¬(∃supports.MultipleSites)

3.1.2. Para todo projeto P no qual existe a necessidade de coordenação com um sistema já instalado e o sistema não possui a gerencia de configuração adequada, então existe risco de gerência de configuração.

ConfigurationManagementRisk ⊒(∀isPartOf.ProjectNeedsCoordinationWithInstalledSystem) ⊓ (∀isPartOf.ProjectConfigurationManagerSystemNotAdequated)
ProjectNeedsCoordinationWithInstalledSystem ≡ ∃needsCoordination.InstalledSystem
ProjectConfigurationManagerSystemNotAdequated ≡ ¬(∃ adequate.ConfigurationManagementSystem)

3.1.3. Para todo projeto P no qual não existe um sistema de gerenciamento de configuração adequado, então existe risco de gerência de configuração.

ConfigurationManagementRisk ⊒ ∀ isPartOf.ProjectConfigurationManagerSystemNotAdequated
ProjectConfigurationManagerSystemNotAdequated ≡ ¬(∃ adequate.ConfigurationManagerSystem)

3.1.4. Para todo projeto P no qual as funções de gerenciamento de configuração não estão adequadamente alocadas a membros qualificados, então existe risco de gerência de configuração.

ConfigurationManagementRisk ⊒ ∀ isPartOf.ProjectFunctionsOfConfigurationManagementSystemNotAllocatedAdequate
ProjectFunctionsOfConfigurationManagementSystemNotAllocatedAdequate ≡ ¬(∃ allocatedAdequate.ConfigurationManagementSystem)

3.2. Axiomas da subclasse Monitoramento – ***MonitoringRisk***

3.2.1. Para todo projeto P no qual não existem relatórios de acompanhamento estruturados periódicos, então existe risco de monitoramento.

MonitoringRisk ⊒ ∀isPartOf.ProjectNotExistsReportsOfAccompanimentStructuralized
ProjectNotExistsReportsOfAccompanimentStructuralized ≡ ¬(∃ exists.ReportsOfAccompanimentStructuralized)

3.2.2. Para todo projeto P no qual as informações apropriadas não são relatadas aos níveis organizacionais corretos, então existe risco de monitoramento.

MonitoringRisk ⊒ ∀ isPartOf.ProjectAppropriateInformationNotReportedForCorrectOrganizationalLevels
ProjectAppropriateInformationNotReportedForCorrectOrganizationalLevels ≡ ∃ exists.AppropriateInformationNotRelatedForCorrectOrganizationalLevels
AppropriateInformationNotRelatedForCorrectOrganizationalLevels ≡ ¬(∃ related.UpperLowerHierarchicalChain)

3.2.3. Para todo projeto P no qual o progresso não é comparado com o planejado, então existe o risco de monitoramento.

MonitoringRisk ⊒ ∀isPartOf. ProjectHoldsProgressNotComparedWithPlan
ProjectHoldsProgressNotComparedWithPlan *≡ ∃holds.ProgressNotComparedWithPlan*
ProgressNotComparedWithPlan *≡ ¬(∃compared.Plan)*

3.3. Axiomas da subclasse Gerenciamento de Pessoal – ***PersonnelManagementRisk***

3.3.1. Para todo projeto P no qual as pessoas não foram treinadas nas qualificações necessárias para este programa, então existe risco de gerenciamento de pessoal.

PersonnelManagementRisk ⊒ ∀isPartOf. ProjectNotTrainingForTeam
ProjectNotTrainingForTeam ≡ ¬(∃training.Team)

3.3.2. Para todo projeto P no qual as pessoas envolvidas não possuem experiências em suas áreas, então existe risco de gerenciamento de pessoal.

PersonnelManagementRisk ⊒ ∀isPartOf. ProjectTeamNotExperienceInTheirAreas
ProjectTeamNotExperienceInTheirAreas ≡ ¬(∃experience.Team)

3.3.3. Para todo projeto P no qual não é fácil para os membros do projeto serem gerenciados, então existe risco de gerenciamento de pessoal.

PersonnelManagementRisk ⊒ ∀isPartOf.ProjectMembersNotEasyManaged
ProjectMembersNotEasyManaged ≡ ¬(∃easyManaged.Members)

3.3.4. Para todo projeto P no qual os membros do projeto não estão cientes de seu status em relação ao planejado, então existe risco de gerenciamento de pessoal.

PersonnelManagementRisk ⊒ ∀*isPartOf.ProjectTeamNotAwareOfStatus*
ProjectTeamNotAwareOfStatus ≡ ¬(∃*awareOfStatus.Team)*

3.3.5. Para todo projeto P no qual os membros do projeto não sentem que é importante seguir o planejamento, então existe risco de gerenciamento de pessoal.

PersonnelManagementRisk ⊒ ∀*isPartOf.ProjectTeamNotFeeImportantToFollowPlan*
ProjectTeamNotFeeImportantToFollowPlan ≡ ¬(∃*fellImportantToFollowPlan.Team)*

3.3.6. Para todo projeto P no qual os membros do projeto não são consultados sobre as decisões gerenciais que afetam seu trabalho, então existe risco de gerenciamento de pessoal.

PersonnelManagementRisk ⊒ ∀
isPartOf.ProjectMembersNotConsultedManagementDecisionsAffectingTheirWork
ProjectMembersNotConsultedManagementDecisionsAffectingTheirWork ≡ ¬(∃
consulted.Members)

3.3.7. Para todo projeto P no qual as pessoas apropriadas não são envolvidas nas reuniões com o cliente, então existe risco de gerenciamento de pessoal.

PersonnelManagementRisk ⊒ ∀
isPartOf.ProjectMeetingTeamNotInvolvedMeetingsWithCustomers
ProjectMeetingTeamNotInvolvedMeetingsWithCustomers ≡ ¬(∃*holds.*
MeetingTeamNotInvolvedMeetingsWithCustomers)
MeetingTeamNotInvolvedMeetingsWithCustomers ≡ ¬(∃*involved.MeetingsWithCustomers)*

3.4. Axiomas da subclasse Controle de Qualidade – ***QualityAssuranceRisk***

3.4.1. Para todo projeto P no qual as funções de controle de qualidade não estão adequadamente alocadas a membros com as qualificações necessárias, então existe risco de controle de qualidade.

QualityAssuranceRisk ⊒ ∀*isPartOf.ProjectFunctionsOfQualityControlNotAllocatedAdequate*
ProjectFunctionsOfQualityControlNotAllocatedAdequate ≡ ¬(∃
allocatedAdequate.FunctionsOfQualityControl)

3.4.2. Para todo projeto P no qual não possui mecanismos definidos para garantir qualidade então existe risco de controle de qualidade.

$QualityAssuranceRisk \sqsupseteq \forall isPartOf.ProjectNotDefinedMechanismsToGuaranteeQuality$
$ProjectNotDefinedMechanismsToGuaranteeQuality \equiv \neg(\exists$
$defined .MechanismsToGuaranteeQuality)$

3.4.3. Para todo projeto P em que não possui métricas de qualidades definidas, então existe risco de controle de qualidade.

$QualityAssuranceRisk \sqsupseteq \forall isPartOf.ProjectNotDefinedMetricsOfQuality$
$ProjectNotDefinedMetricsOfQuality \equiv \neg(\exists defined .MetricsOfQuality)$

4. Definição dos Axiomas da Classe Riscos de Processo de Gerenciamento – *ManagementProcessRisk*

A hierarquia da classe *ManagementProcessRisk*, subclasse de *DevelopmentEnvironmentRisk*, pode ser vista a seguir na Figura A-4:

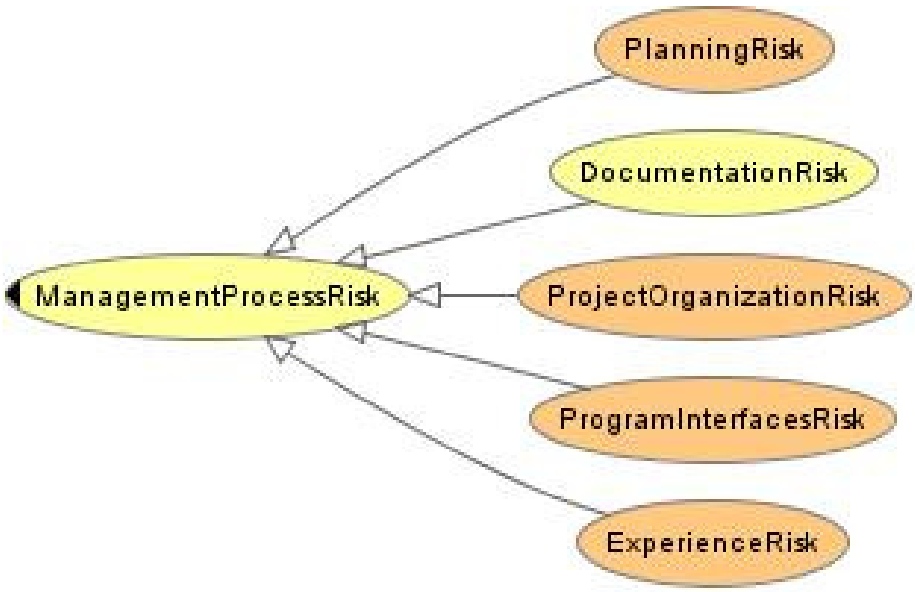

Figura A-4. Hierarquia da Classe *ManagementProcessRisk*

4.1. Axiomas da subclasse Experiência de Gerenciamento – ***ExperienceRisk***

4.1.1. Para todo projeto P para o qual os gerentes não sejam experientes, então existe risco de experiência de gerenciamento.

$ExperienceRisk \sqsupseteq \forall isPartOf.ProjectManagerNotHasExperience$
$ProjectManagerNotHasExperience \equiv \neg(\exists esperience.Manager)$

4.2. Axiomas da subclasse Planejamento – ***PlanningRisk***

4.2.1. Para todo algum projeto P para o qual existe um planejamento e o projeto não esta sendo gerenciado de acordo com o plano, então existe risco de planejamento.

$PlanningRisk \sqsupseteq (\forall isPartOf.ProjectExistsPlan) \sqcap (\forall isPartOf.ProjectProgressNotAccordanceWithPlan)$
$ProjectExistsPlan \equiv \exists exists.Plan$
$ProjectProgressNotAccordanceWithPlan \equiv \neg(\exists accordance.Plan)$

4.2.2. Para todo projeto P para o qual ocorre alguma interrupção e não existe o replanejamento, então existe risco de planejamento.

$PlanningRisk \sqsupseteq \forall isPartOf.ProjectInterruptionWithoutReplan$
$ProjectInterruptionWithoutReplan \equiv (\exists interruption.Progress) \sqcap \neg(\exists exists.Replan)$

4.2.3. Para todo projeto P no qual não existem planos de contingência para todos os riscos conhecidos, então existe risco de planejamento.

$PlanningRisk \sqsupseteq \forall isPartOf.ProjectNotExistsPlanContingencyForKnownRisks$
$ProjectNotExistsPlanContingencyForKnownRisks \equiv \neg(\exists exists.$
$PlanContingencyForKnownRisks)$

4.2.4. Para todo projeto P para o qual as tarefas de longo prazo não estão adequadamente sendo tratadas, então existe risco de planejamento.

$PlanningRisk \sqsupseteq \forall isPartOf.ProjectActivitiesOfLongTimeNotTreated$
$ProjectActivitiesOfLongTimeNotTreated \equiv \neg(\exists treated.ActivitiesOfLongTime)$

4.2.5. Para todo projeto P no qual as pessoas em todos os níveis não são incluídas no planejamento de suas atividades, então existe risco de planejamento.

$PlanningRisk \sqsupseteq \forall isPartOf.ProjectTeamNotIncludedInPlan$
$ProjectTeamNotIncludedInPlan \equiv \neg(\exists included.Team)$

4.3. Axiomas da subclasse Interfaces de Comunicação de Programa – ***ProgramInterfacesRisk***

4.3.1. Para todo projeto P no qual existem conflitos não documentados e resolvidos em tempo apropriados, então existe risco de interface de comunicação.

ProgramInterfacesRisk ⊒(∀isPartOf.ProjectExistsConflictsNotDocumented) ⊓(∀ isPartOf.ProjectExistsConflictsNotResolvedInAdequateTime)
ProjectExistsConflictsNotDocumented ≡ ¬(∃documented.Conflicts)
ProjectExistsConflictsNotResolvedInAdequateTime ≡ ¬(∃resolvedInAdequateTim.Conflicts)

4.3.2. Para todo projeto P no qual os representantes de todas as partes interessadas do cliente estão representados nas decisões acerca de funcionalidades e operações, então existe risco de interface de comunicação.

ProgramInterfacesRisk ⊒(∀ isPartOf.ProjectHoldsAllFactionCustomerRepresentantInvolvedDecisionsAboutFunctionality) ⊓ (∀isPartOf.ProjectHoldsAllFactionCustomerRepresentantInvolvedDecisionsAboutOperations)
ProjectHoldsAllFactionCustomerRepresentantInvolvedDecisionsAboutFunctionality ≡ ∃ holds.AllFactionCustomerRepresentantInvolvedDecisionsAboutFunctionality
AllFactionCustomerRepresentantInvolvedDecisionsAboutFunctionality ≡ ∃ involved.DecisionsAboutFunctionality
ProjectHoldsAllFactionCustomerRepresentantInvolvedDecisionsAboutOperations ≡ ∃ holds.AllFactionCustomerRepresentantInvolvedDecisionsAboutOperations
AllFactionCustomerRepresentantInvolvedDecisionsAboutOperations ≡ ∃ involved.DecisionsAboutOperations

4.3.3. Para todo projeto P no qual existe uma cadeia hierárquica superior e inferior e existem problemas no projeto e não é feita a comunicação destes problemas, então existe risco de interface de comunicação.

ProgramInterfacesRisk ⊒ ∀ isPartOf.ProjectProblemsNotCommunicatedForUpperLowerHierarchicalChain
ProjectProblemsNotCommunicatedForUpperLowerHierarchicalChain ≡ (∃ exists.UpperLowerHierarchicalChain) ⊓(∃ communicatedProblem.UpperLowerHierarchicalChain)

4.3.4. Para todo projeto P no qual os membros apropriados não são envolvidos em reuniões com os clients, então existe risco de interface de comunicação.

ProgramInterfacesRisk ⊒ ∀isPartOf.ProjectMeetingTeamNotInvolvedMeetingsWithCustomers
ProjectMeetingTeamNotInvolvedMeetingsWithCustomers ≡ ¬(∃ holds.MeetingTeamInvolvedMeetingsWithCustomers)
MeetingTeamInvolvedMeetingsWithCustomers ≡ ∃involved. MeetingsWithCustomers

4.4. Axiomas da subclasse Organização do Projeto – ***ProjectOrganizationRisk***

4.4.1. Para todo projeto P no qual não existe uma organização do projeto, então existe risco de organização do projeto.

$ProjectOrganizationRisk \sqsupseteq \forall isPartOf.ProjectNotExistsOrganizationOfProject$
$ProjectNotExistsOrganizationOfProject \equiv \neg(\exists exists.OrganizationOfProject)$

4.4.2. Para todo projeto P no qual as pessoas não entendem seus papeis ou os papeis dos demais no projeto, então existe risco de organização do projeto.

$ProjectOrganizationRisk \sqsupseteq \forall isPartOf.ProjectTeamNotUnderstandTheirRolesOrOtherRoles$
$ProjectTeamNotUnderstandTheirRolesOrOtherRoles \equiv \neg(\exists understandTheirRoles.Team) \sqcup \neg(\exists understandOtherRoles.Team)$

4.4.3. Para todo projeto P no qual as pessoas não sabem quem tem autoridade para o que, então existe risco de organização do projeto.

$ProjectOrganizationRisk \sqsupseteq \forall isPartOf.ProjectMembersNotKnowWhoAutority$
$ProjectMembersNotKnowWhoAutority \equiv \neg(\exists knowWhoHasAuthority.Team)$

5. Definição dos Axiomas da Classe Riscos de Ambiente de Trabalho – *WorkEnvironmentRisk*

A hierarquia da classe *WorkEnvironmentRisk*, subclasse de *DevelopmentEnvironmentRisk*, pode ser vista a seguir na Figura A-5:

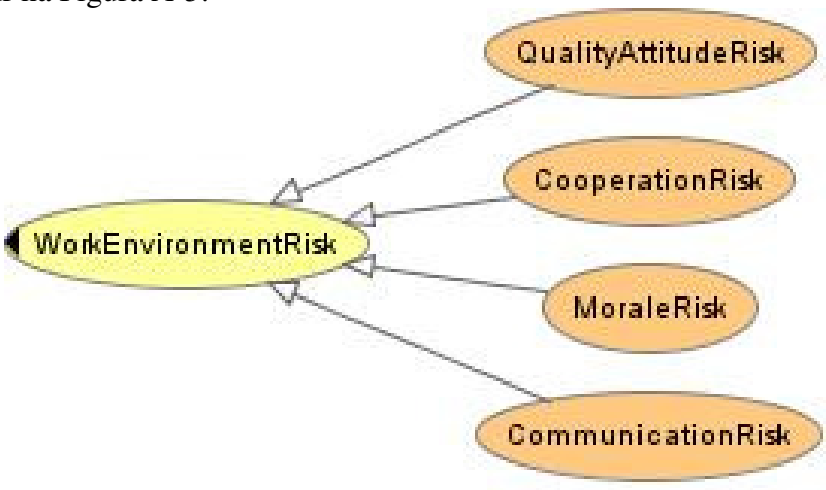

Figura A-5. Hierarquia da Classe *WorkEnvironmentRisk*

5.1. Axiomas da subclasse Comunicação – ***CommunicationRisk***

5.1.1. Para todo projeto P no qual não existe boa comunicação entre seus membros, então existe risco de comunicação.

CommunicationRisk ⊒ ∀isPartOf.ProjectNotExistsGoodCommunicationBetweenMembers
ProjectNotExistsGoodCommunicationBetweenMembers ≡ ¬(∃good.Communication)

5.1.2. Para todo projeto P no qual os gerentes não estão receptivos a comunicação dos membros do projeto, então existe risco de comunicação.

CommunicationRisk ⊒ ∀
isPartOf.ProjectHoldsManagerNotReceptiveCommunicationOfMembers
ProjectHoldsManagerNotReceptiveCommunicationOfMembers ≡ ∃
holds.ManagerNotReceptiveCommunicationOfMembers
ManagerNotReceptiveCommunicationOfMembers ≡ ¬(∃receptiveCommunication.Members)

5.1.3. Para todo projeto P no qual os membros não recebem notificações acerca de suas atividades em tempo adequado, então existe risco de comunicação.

CommunicationRisk ⊒ ∀
isPartOf.ProjectExistsMembersNotReceivedNotificationsInAppropriateTime
ProjectExistsMembersNotReceivedNotificationsInAppropriateTime ≡ ∃
exists.MembersNotReceivedNotificationsInAppropriateTime
MembersNotReceivedNotificationsInAppropriateTime ≡ ¬(∃
receive.NotificationsInAppropriateTime)

5.2. Axiomas da subclasse Risco de Cooperação – ***CooperationRisk***

5.2.1. Para todo projeto P em que as pessoas não trabalham efetivamente em prol dos objetivos em comum, então existe risco de cooperação.

CooperationRisk ⊒ ∀isPartOf.ProjectTeamNotWork CommonGoals
ProjectTeamNotWork CommonGoals ≡ ¬(∃work CommonGoals.Team)

5.2.2. Para todo projeto P em que as pessoas não trabalham cooperativamente além de seus limites funcionais, então existe risco de cooperação

CooperationRisk ⊒ ∀isPartOf.ProjectTeamNotWorkCooperatively.
ProjectTeamNotWorkCooperatively ≡ ¬(∃workCooperatively.Team)

5.2.3. Para todo projeto P em que são necessárias intervenções gerenciais para que as pessoas possam trabalham conjuntamente, então existe risco de cooperação.

CooperationRisk ⊒ ∀isPartOf.ProjectMembersNeedsInterventionManagementForWorkJointly
ProjectMembersNeedsInterventionManagementForWorkJointly ≡ ∃
needsInterventionManagement.Members

5.3. Axiomas da subclasse Risco de Moral – ***MoraleRisk***

5.3.1. Para todo projeto P no qual a moral está baixa, então existe risco de moral.

$MoraleRisk \sqsupseteq \forall isPartOf.ProjectLowMoraleOfTeam$
$ProjectLowMoraleOfTeam \equiv \exists lowMorale.Team$

5.3.2. Para todo projeto P no qual os membros estão sendo afetados por problemas, então existe risco de moral.

$MoraleRisk \sqsupseteq \forall isPartOf.ProjectTeamAffectedByProblems$
$ProjectTeamAffectedForProblems \equiv \exists affectedByProblems.Team$

5.4. Axiomas da subclasse Atitude Para Qualidade – ***QualityAttitudeRisk***

5.4.1. Para todo projeto P no qual os múltiplos níveis dos membros não estão orientados a procedimento de qualidade, então existe risco de atitude para qualidade.

$QualityAttitudeRisk \sqsupseteq \forall$
$isPartOf.ProjectHoldsMultipleLevelsOfMembersNotOrientedToMechanismsToGuaranteeQuality$
$ProjectHoldsMultipleLevelsOfMembersNotOrientedToMechanismsToGuaranteeQuality \equiv \exists$
$holds.MultipleLevelsOfMembersNotOrientedToMechanismsToGuaranteeQuality$
$MultipleLevelsOfMembersNotOrientedToMechanismsToGuaranteeQuality \equiv \neg(\exists oriented.$
$MechanismsToGuaranteeQuality)$

5.4.2. Para todo projeto P no qual o cronograma não considera a qualidade, então existe risco de atitude para qualidade.

$QualityAttitudeRisk \sqsupseteq \forall isPartOf.ProjectScheduleNotConsidersQuality$
$ProjectScheduleNotConsidersQuality \equiv \neg(\exists considersQuality.Schedule)$