Universidade Federal de Pernambuco
Centro de Informática
Pós-Graduação em Ciência da Computação

*Onto*PRIME: Ontologia de Riscos para Ambientes de Desenvolvimento de Software Multiprojetos

Por
Antonio Campello
Cristine Gusmão
Leonardo Amorim
Marcelo Guedes
Monique Monteiro

Profa. Dra. Patrícia Tedesco
Orientadora

Prof. Dr. Hermano Perrelli de Moura
Co-orientador

Recife, setembro de 2004

"A gerência de risco é a gerência de projetos para adultos ."
(DeMARCO, Tom)

SUMÁRIO

LISTA DE FIGURAS

CAPÍTULO 1

INTRODUÇÃO

Este capítulo apresenta as questões que motivaram a realização deste projeto, objetivos geral e específicos, contribuições esperadas, escopo, metodologia de pesquisa e por fim, a estrutura adotada no documento.

As incertezas são parte do cotidiano humano. Desde os primórdios até os dias atuais o homem procura defender-se dos riscos que o cercam, galgando níveis de satisfação das necessidades básicas, de segurança e culminando nas de cunho puramente profissional. As pessoas em sua grande maioria diariamente fazem escolhas, com graus diferenciados de riscos, mas também com um alto grau de oportunidade e benefícios associados.

Deste modo, tem-se que as incertezas incutem, geram e implicam em riscos. E estes, são definidos como a probabilidade ou possibilidade da ocorrência de valores para determinados eventos e fenômenos, não desejáveis e, ou, adversos. Isto leva a estabelecer a convivência contínua e inevitável com inúmeros tipos de riscos.

Porém, entende-se que esta convivência precisa ser explicitada, ou mesmo, elucidada, favorecendo a identificação, análise e quantificação da intensidade dos riscos. Proporcionando o estabelecimento de estratégias que visem a ações de prevenção, minimização e, ou, mitigação dos efeitos associados aos riscos.

Os riscos possuem diferentes significados, como os de ordem física, estrutural, econômica, social e ambiental. Podendo estes, desdobrar-se em diversos componentes e sucessivos níveis de detalhamento. A exemplos destes casos, podem-se citar os riscos associados a atividades como a: (a) elaboração de projetos de desenvolvimento de software, (b) formulação de análises de viabilidade econômica e financeira destes projetos, (c) realização de análises de projetos similares e (d) condução do projeto de Melhoria de Processo do Software Brasileiro (mps BR).

Atualmente todos os ramos da atividade humana dependem de alguma forma da utilização de software para operar, dar suporte, controlar equipamentos e fluxos de informações, gravar ou

processar atividades. A área de Engenharia de Software tem promovido vários estudos com a finalidade de produzir modelos de melhoria, processos, métodos e ferramentas para aumentar a probabilidade de sucesso na execução de projetos de software, garantindo a qualidade de seus produtos [PRESSMAN, 1995]. Portanto, na capacidade de prevenir e controlar riscos pode estar o diferencial no gerenciamento de projetos em organizações desenvolvedoras de software.

1.1. MOTIVAÇÃO

Atualmente, os sistemas de computação estão difundidos em todos os setores da vida moderna e apesar dos avanços nas tecnologias de software, a maioria dos sistemas de software continua difícil de entender, manter e evoluir [HALL, 1998]. Desta forma, à medida que o tamanho e a complexidade dos sistemas de software crescem, aumenta a necessidade da utilização de metodologias para o gerenciamento de riscos, apoiando os projetistas e gerentes de projetos de tecnologia da informação (TI) - no desenvolvimento e efetivação de suas atividades, garantindo o cumprimento das metas do projeto, custos e prazos, e por fim, a qualidade do processo e do produto gerado.

Existem várias definições e usos para o termo risco. Knight definiu risco com sendo a exposição a eventos incertos com probabilidades conhecidas [KNIGHT, 1921]. No dicionário Houaiss, o termo risco é definido como: “probabilidade de perigo” [HOUAISS, 2001]. Como também no dicionário Webster’s que diz: “risco é a probabilidade de perda, injúria, desvantagem ou destruição” [WEBSTERS, 1993]. No entanto, segundo Bernstein [BERNSTEIN, 1997], a origem da palavra risco vem do italiano antigo, *risicare* que significa “ousar”, que por sua vez, deriva do latin *risicu*, *riscu*. Desta forma é uma escolha e não o destino. E ainda: “a capacidade de administrar riscos, e com ela, a vontade de correr riscos e fazer opções ousadas são elementos-chaves da energia que impulsiona o mundo.” Logo, sendo o risco uma opção, pode-se medi-lo, avaliar as conseqüências de sua ocorrência e conseqüentemente, geri-lo.

Muitas abordagens para gerenciar riscos em projetos de software vêm sendo propostas e usadas desde que Boehm [BOEHM, 1991] e Charette [CHARETTE, 1990] conseguiram trazer a atenção da comunidade de Engenharia de Software para a necessidade de gerir risco, através de suas propostas de processos de gerenciamento de risco. Embora existam muitos relatórios

individuais de sucessos em desenvolvimento de software, através de utilização de técnicas de gestão de risco, na prática, a indústria de software como um todo, ainda não aplica ativamente e sistematicamente os métodos de um efetivo processo de gerência de risco.

Tom De Marco [DeMARCO, 2003], em seu último livro "*Waltzing with Bears*", fazendo um comparativo com a maturidade organizacional e os vários modelos existentes hoje para suporte a garantia da qualidade dentro das organizações, diz que o processo de gerência de risco ainda está no jardim da infância. Muitas organizações não têm a preocupação de institucionalizar o gerenciamento de risco, talvez porque seus profissionais e equipes já realizam de uma forma ou de outra, atividades de identificação e análise de riscos, ou porque são tantos os detalhes, atividades, marcos e artefatos gerados, que realizar mais atividades tem um preço e custo a considerar. Ou ainda, por não existir nas organizações que desenvolvem software a cultura de ter explicitamente uma estrutura para gerir riscos.

Identificar risco não é tudo. Muitas são as variáveis que influenciam os projetos, não só as de caráter técnico, mas financeiras, negócio, pessoais entre tantas outras. Os Ambientes de Desenvolvimento de Software, principalmente os que desenvolvem vários projetos em paralelo, necessitam de um 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. É a memória organizacional de projetos. Saber o que se pode fazer ou não fazer, em determinadas situações, pode representar a continuidade da organização. Não esquecendo que estas organizações, na grande maioria das situações tratadas, precisam de decisões rápidas e eficazes.

Desta forma, muito se tem a fazer para permitir que a gerência de risco, como a própria gerência de projeto, seja um processo que flua em todas as fases do desenvolvimento de software de forma que:

- Os *Stakeholders* precisam entender o que é "Gerência de Risco" e como o projeto pode usufruir os benefícios de sua aplicação.
- A implementação das atividades do processo de Gerência de Risco deve ser adaptada para cada projeto de forma apropriada.

- O processo de Gerência de Risco deve ser avaliado periodicamente como forma de melhoria do processo de desenvolvimento de software e garantia da qualidade dos produtos gerados.
- Deve existir um Plano Diretor para a Gerência de Risco na organização.
- A Gerência de Risco deve estar presente em todas as áreas e atividades necessárias para o desenvolvimento de um produto de software.
- A correta elicitação de requisitos é um dos fatores de sucesso dentro de um projeto de desenvolvimento de software.
- Gerenciar riscos não é apenas realizar testes.
- É vital definir os indicadores e métricas associadas para o acompanhamento e controle dos riscos.
- É preciso ter um processo cíclico e sistemático de Gerência de Risco.
- A Gerência de Risco deve ter uma independência dentro do desenvolvimento, assim com a Gerência de Qualidade, embora esteja diretamente ligada a esta.
- Gerenciar riscos é um processo obrigatório. É a garantia da realização de projetos e produtos saudáveis.

Gerenciar projetos de software, visando a aumentar a competência da organização no planejamento, identificação, avaliação e execução de projetos, envolve um conjunto bastante variado de aspectos que influenciam no desenvolvimento de um produto de software, dentre os quais temos: aspectos cognitivos, que estão relacionados aos fatores humanos envolvidos no projeto (equipe de desenvolvimento, *stakeholders*, etc.), aspectos econômicos, que são mais influenciados pelos interesses comerciais da organização, e aspectos técnicos, que se relacionam às etapas e atividades relacionadas ao desenvolvimento do projeto (análise, projeto, implementação, teste, etc.).

O gerenciamento de risco é um processo através do qual os gerentes de projeto satisfazem as necessidades de identificação de fatores de risco chaves para os projetos em andamento,

obtendo consistência e entendimento, escolhendo ou priorizando que riscos devem ser mitigados e eliminados, como também potencializando os resultados das oportunidades encontradas.

1.1.1. Relevância

De uma forma geral, gerentes de projetos têm que alocar, ratear recursos entre projetos, gerenciá-los dentro do orçamento e tempo disponíveis, e por fim, garantir que estes recursos limitados sejam implementados de acordo com o planejado para alcançar o objetivo definido.

A maioria dos gerentes de projeto de software utiliza a estratégia de gerenciamento de risco reativo, que nada mais é do que reagir ao risco de acordo com a ocorrência. Mais barato e lógico é ter atitude pró-ativa. A estratégia de gerenciamento de risco pró-ativa começa pelo planejamento da gerência de risco e é um processo de âmbito organizacional.

O resultado de todos os projetos desenvolvidos por uma organização tem grande parcela de contribuição no seu sucesso. Projetos individuais influenciam a organização, mas também sofrem a influência de todos os outros projetos que estejam sendo inicializados, ou mesmo, executados no período. Em apenas um projeto já existe muita informação a ser tratada. Nenhum projeto é desenvolvido isoladamente. Priorizá-los e garantir que os projetos mais importantes sejam realizados é um dos grandes, se não vital, objetivos organizacionais.

Este trabalho tem sua relevância e conseqüente contribuição pautada na construção de uma Ontologia de Riscos que dará suporte ao processo de Gerência de Risco 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. Embora algumas partes deste tópico – Gerência de Risco, Gerência de Múltiplos Projetos – tenham merecido alguma atenção da literatura, ainda existe a necessidade de um maior aprofundamento e conseqüente detalhamento, pois as pesquisas existentes, em sua grande maioria, foram conduzidas de forma pontual, dentro da Engenharia de Software.

1.2. OBJETIVOS

O objetivo fundamental deste trabalho é contribuir com a indústria de software através da definição da *Onto*PRIME (*Ontology for Project Risk Management*), que dará suporte ao modelo de Gerenciamento de Risco em Ambientes de Desenvolvimento de Software de Múltiplos Projetos. Para alcançá-lo faz-se necessário atual em nível:

Geral:

A OntoPRIME tem como finalidade fornecer um vocabulário comum que possa ser utilizado para representar conhecimento útil para os desenvolvedores de software sobre os riscos e oportunidades que podem afetar um projeto de software, vários projetos e mesmo, entre projetos, dentro de uma organização desenvolvedora de software.

Específico:

1. A OntoPRIME deve fornecer a estrutura a ser utilizada para organizar o conhecimento sobre riscos e oportunidades, podendo ser utilizada para orientar a aquisição do conhecimento dos riscos dos projetos;

2. A OntoPRIME deverá contemplar as informações sobre riscos, processos de gerência de risco, métodos, ferramentas e técnicas para identificação e controle de riscos.

1.3. CENÁRIOS DE UTILIZAÇÃO

Para melhor fundamentação da importância da *Onto*PRIME, no domínio dos riscos que podem influenciar projetos de software, são apresentados a seguir alguns possíveis cenários para sua utilização:

1. Tendo com base características, e outros parâmetros definidos de projetos, a OntoPRIME pode permitir a identificação dos principais riscos associados e apresentar uma lista priorizando-os, de acordo com a estratégia organizacional;

2. Desenvolver software é um conjunto complexo de atividades. A estrutura definida pela OntoPRIME pode ser utilizada para a definição dos Planos de Gerência de Risco para diversos tipos de projetos, o que favorece o desenvolvimento e/ou adaptação de ferramentas CASE baseadas na ontologia;

3. Uma das grandes dificuldades em Ambientes de Desenvolvimento de Software é a integração entre as várias ferramentas utilizadas. A OntoPRIME, a partir das bases geradas em conformidade com sua estrutura, favorece a integração de dados e a aquisição de conhecimento;

4. Um sistema que manipula o conhecimento sobre riscos que influenciam os projetos em Ambientes de Desenvolvimento de Software Multiprojetos precisa ser desenvolvido.

    a. Desta forma, primeiramente os riscos que envolvem os requisitos podem ser facilmente tratados se um vocabulário comum for utilizado pelas pessoas envolvidas na tarefa;

    b. Em segundo lugar, a estrutura para representação do conhecimento sobre riscos em Ambientes de Desenvolvimento de Software e o conhecimento já existente na organização podem ser utilizados para compor uma primeira visão sobre os requisitos do novo projeto;

    c. Terceiro, é importante conhecer o papel e responsabilidades de cada um dos membros da organização, bem como suas qualificações. Isso favorece o desempenho das atividades;

    d. Quarto e último, é importante saber quais os tipos de riscos que mais incidem em determinados tipos de projetos. A OntoPRIME tem papel essencial em todos os quatro itens.

5. A gerência de projeto está preparando os recursos para um novo projeto. Seleção de pessoas, cotação de equipamentos, compartilhamento de recursos para alocação no novo projeto com características próprias. Por outro lado, os profissionais da equipe de desenvolvimento estão precisando de apóio especializado para a realização de determinada atividade. Em ambos os casos, o conhecimento sobre riscos nos projetos, obtido com base na OntoPRIME, pode auxiliar a identificação dos membros com as características e conhecimentos desejados.

1.4. METODOLOGIA DE DESENVOLVIMENTO

Através da Figura 1 pode-se melhor visualizar todas as etapas necessárias para a formalização da Ontologia de Gerenciamento de Riscos – OntoPRIME, na sua totalidade.

A metodologia adotada para a realização deste projeto foi primeiramente, a revisão bibliográfica para conhecer o estado atual da gerência de risco na Engenharia de Software e as diversas abordagens adotadas para o processo de gerência de risco: desde o planejamento estratégico, identificação de fatores de risco, análise e priorização até o efetivo controle e tratamento de riscos. A exploração do tema baseou-se em leituras nacionais e internacionais. Foram pesquisados livros da área de engenharia de software, gerência de projetos e gerência de risco. Periódicos também foram explorados, como jornais, relatórios técnicos e artigos da área acadêmica e industrial, com foco em modelos, normas e estudos de casos. A partir destes estudos optou-se por utilizar como base a Taxonomia de Riscos proposta pelo *Software Engineering Institute* em 1993 [CARR, 1993] para dar suporte ao seu Processo Contínuo de Gerenciamento de Riscos. A partir desta Taxonomia foram analisados os relacionamentos e axiomas derivados de suas classes e subclasses, conforme fase 2, Figura 1.

As Fases 4, 5, Figura 1, corresponderão a adaptação das características do gerenciamento de portfólio de projetos, validação e melhoramento do modelo proposto, onde será realizado um estudo detalhado, cuja finalidade é validar os componentes identificados para o modelo em ambientes de múltiplos projetos. Durante estas fases serão utilizados métodos quantitativos e qualitativos de levantamento de dados. As técnicas de coleta de informações que serão utilizadas são: entrevista individual estruturada, montagem de grupos focais com gerentes de projeto de software, questionários para a equipe de desenvolvimento, bem como para clientes externos; e por fim levantamento de documentos referentes à metodologia e processo de desenvolvimento.

Após a compilação de uma versão preliminar do modelo sobre processo e técnicas deve ser submetido a discussões com profissionais de empresas locais produtoras de software, no sentido de avaliar sua aplicabilidade em ambiente local de desenvolvimento de “produtos de software” (Fase 4, Figura 1). Com base em tais discussões, serão promovidos os ajustes que forem considerados adequados.

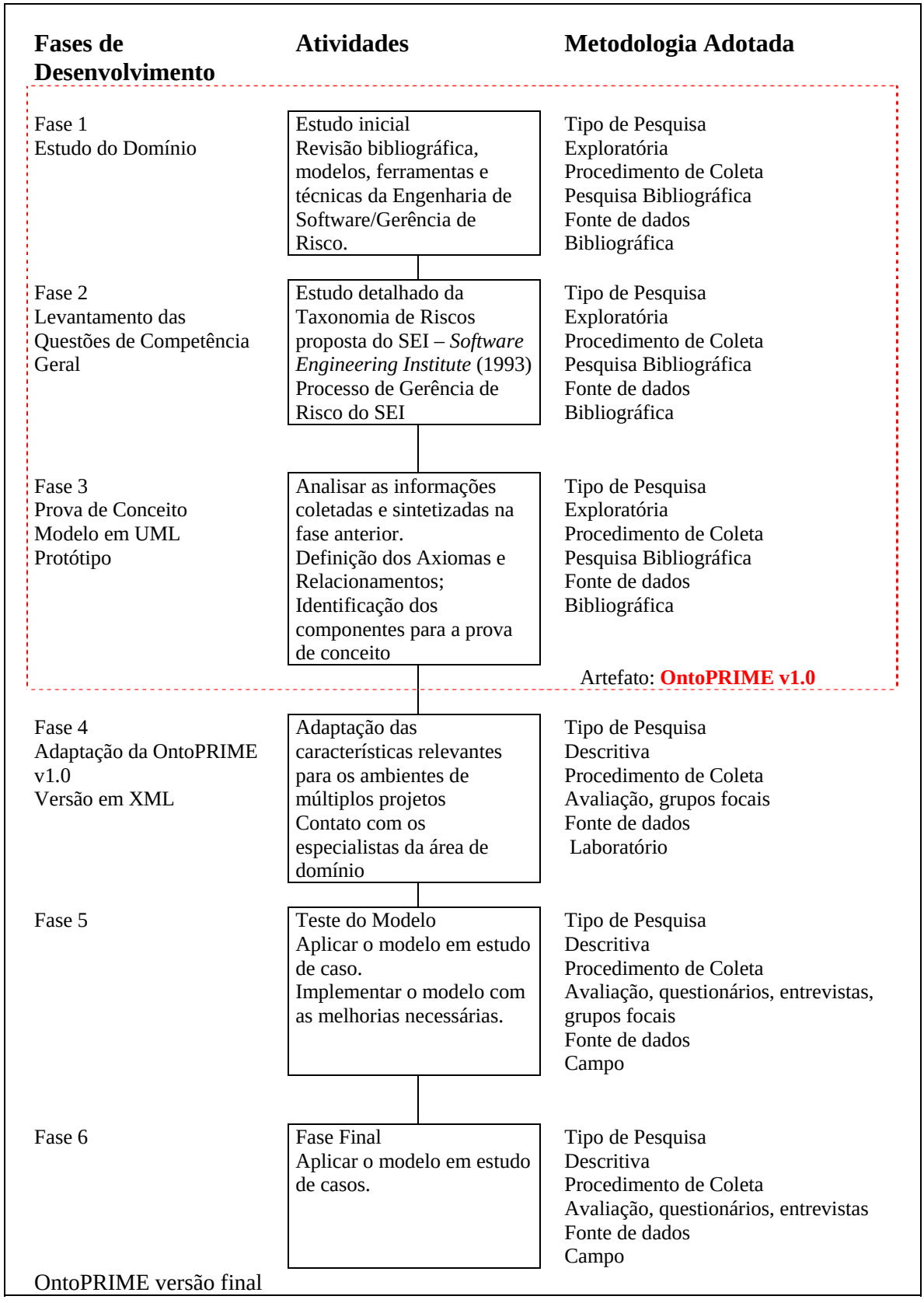

Figura 1 – Metodologia Desenvolvimento - OntoPRIME

O passo seguinte deverá ser a seleção de empresas produtoras de software, que possam fazer parte do estudo de caso, nas quais se aplicarão os processos e técnicas recomendadas, com vistas a que se faça uma avaliação das dificuldades do uso, suas omissões, suas carências ou seus excessos (Fase 5, Figura 1).

As conclusões dos estudos de caso servirão para se proceder aos ajustes finais e a produção da última versão da OntoPRIME, concluindo-se assim o projeto maior com a compilação das conclusões e recomendações.

1.5. ESCOPO

Projeto

Este projeto foi desenvolvido implementando as fases 1 a 3 da Figura 1, com o objetivo de atender ao primeiro cenário de utilização apresentado na seção 1.3. Portanto foram desenvolvidas três sub-ontologias no domínio dos riscos, baseadas na Taxonomia do SEI [CARR, 1993], compondo a versão inicial da OntoPRIME.

As três ontologias definidas, voltadas respectivamente para riscos de Engenharia do Produto, Ambiente de Desenvolvimento e Restrições de Programa, tiveram relacionamentos internos e externos definidos, escritos em linguagem natural e axiomas em Lógica de Primeira Ordem (LPO).

Protótipo

Para análise dos relacionamentos um protótipo foi implementado utilizando apenas os axiomas da classe de Engenharia do Produto, devido à grande e complexa quantidade de informações geradas.

O objetivo do protótipo foi apresentar a efetiva utilização da linguagem proposta pela OntoPRIME. O cenário de validação foi o preenchimento de um questionário onde cada resposta marcada revela uma característica do projeto e pode levar à inferência de um determinado tipo de risco. A lista de riscos identificados é representada por meio de uma árvore de riscos gerada com base nas características do projeto (obtidas a partir das respostas ao questionário) e nos relacionamentos de causa e efeito entre diferentes tipos de risco definidos para a ontologia. Assim, dentro da árvore retornada, cada "nó interno" (diferente de "nó raiz" ou "folha") é um risco cujos "nós filhos" representam suas causas, que tanto podem

ser características do projeto como outros riscos. O "nó raiz" representa o risco do projeto como um todo.

Em um ambiente real, este tipo de aplicação seria utilizado de forma transparente para os usuários, equipes de desenvolvimento, gerentes, ou quaisquer outras pessoas participantes de um projeto, conforme apresentado no item 1.3 – Cenários de Utilização, sem que fosse necessariamente exigido o preenchimento de questionários, mas sim a descrição completa do(s) projeto(s) e de todos os seus artefatos em um formato pré-determinado. Além disso, seria utilizado no lugar do usuário humano um agente monitor responsável por capturar e interpretar tais descrições e analisá-las levando em conta os relacionamentos entre diferentes projetos e o histórico de riscos de projetos passados.

1.6. ESTRUTURA DO DOCUMENTO

Este trabalho está estruturado de forma a tratar o tema gradativamente e de acordo com a abordagem adotada para o desenvolvimento do mesmo. Em conseqüência, os capítulos foram estruturados como segue:

Capítulo 2 – Gerência de Risco na Engenharia de Software: objetiva apresentar os conceitos chaves da área de Gerência de Risco utilizados para a concepção da Ontologia de Riscos.

Capítulo 3 – Ontologia de Riscos: apresenta os requisitos e suas especificações em Lógica de Primeira Ordem (LPO) considerada para a construção da versão 1.0 da OntoPRIME.

Capítulo 4 – Arquitetura e Protótipo: mostra o modelo arquitetural definido para dar suporte a implementação do protótipo da OntoPRIME.

Capítulo 5 – Considerações Finais e Trabalhos Futuros apresenta as lições aprendidas ao longo da implementação do projeto e proposta de extensão deste trabalho.

CAPÍTULO 2

GERÊNCIA DE RISCO EM ENGENHARIA DE SOFTWARE

A finalidade é introduzir os conceitos básicos da área de Gerência de Riscos utilizados para implementar a OntoPRIME.

O desenvolvimento de software pode ser considerado uma atividade de risco e diversos estudos e autores atuais comprovam que muitos dos problemas envolvidos em projetos de grande porte estão muito mais associados a falhas em atividades de gerenciamento do que falhas em atividades técnicas.

Ultimamente, a área que trata riscos na engenharia de software evoluiu, passando de uma análise dentro das fases do modelo de desenvolvimento de software, como era a proposta do modelo em Espiral [HIGUERA, 1996], para se tornar uma gerência que deve permear todos os processos do ciclo de vida do software[1.]

Nestes últimos anos, a Gerência de Riscos, de uma forma geral, tem buscado alcançar o adequado balanceamento entre aproveitar as oportunidades e minimizar os eventos adversos. A Gerência de Risco é parte integrante das boas práticas de gestão empresarial. É desenvolvida como um processo iterativo, composto de etapas seqüenciais, de modo a permitir a melhoria contínua da tomada de decisões e do desempenho organizacional.

Atualmente, gerir riscos envolve o estabelecimento de uma cultura e infra-estrutura adequadas, bem como a aplicação de uma metodologia lógica e sistemática para administrar os riscos "negativos" (de perdas) e os riscos "positivos" (de ganhos), associados a qualquer atividade, função, processo ou projeto.

Para ser eficiente a gerência de risco deve ser apresentada à organização. Todos os seus membros devem ser capacitados e o processo adotado não pode andar sozinho. Vários são os exemplos de métodos e técnicas, novos e consagrados, que podem ser utilizados nesse contexto atual da gerência de riscos. No mundo atual dos negócios medidas e estatísticas de riscos não

[1] O ciclo de vida do software vai desde a concepção de idéias até a descontinuidade do produto de software.

são o suficiente. Os estudos de avaliação subjetiva dos riscos vêm crescendo e ganhando adeptos rapidamente. A razão para esta mudança é a rapidez com que se pode avaliar os riscos através de uma análise qualitativa ao invés de gastar tempo em obter estatísticas e valores de riscos mais realistas.

A gerência de risco na Engenharia de Software tem a finalidade de aumentar a qualidade do produto e do processo de desenvolvimento de software. Observa-se que os projetos de desenvolvimento de software, em geral, apresentam atrasos de cronogramas, custos realizados além do planejado e funcionalidades aquém das expectativas. Esses problemas, embora considerados inerentes ao desenvolvimento de software por muitos autores, podem ser minimizados e controlados pelo contínuo gerenciamento de risco de projetos. Pode-se destacar dois objetivos principais da gerência de risco: prevenção e mitigação de riscos. Ainda que a eliminação completa dos riscos possa parecer um ideal da gerência de risco, é utopia pensar na totalidade desta eliminação. Os objetivos são formulados de maneira mais realista, proporcionando aos gerentes de projeto meios que tornem o projeto mais previsível e controlável.

2.1. CONCEITOS CHAVES

Este trabalho tem seu foco em Ambientes de Desenvolvimento de Software Multiprojetos, mais especificamente na formalização do vocabulário utilizado – OntoPRIME, ontologia que será usada como base para aquisição e manipulação de conhecimento sobre riscos nestes ambientes.

A seguir serão apresentados os conceitos essenciais para a organização da estrutura, relacionamentos e axiomas da OntoPRIME.

Risco de Software

Riscos são inerentes a qualquer atividade de desenvolvimento de software. Assumir riscos é essencial para o progresso e falhas são normalmente partes do aprendizado. Por outro lado, a ocorrência inevitável de riscos não implica na inabilidade de reconhece-los e geri-los, muito pelo contrário, deve-se gerir para minimizar seu impacto negativo enquanto ganha-se a oportunidade de desenvolver novos e melhores produtos de software.

No contexto da Taxonomia de Riscos [CARR, 1993], objeto base para o desenvolvimento da OntoPRIME, risco tem um significado negativo para o projeto. São classificados como: conhecidos, previsíveis e imprevisíveis. Os riscos conhecidos podem ser descobertos após uma avaliação cuidadosa do plano do projeto, ambiente técnico e do negócio, como por exemplo: prazos irreais, escopo mal definido, ambiente de desenvolvimento ruim. Os previsíveis são percebidos a partir de experiências em projetos anteriores (rotatividade de pessoal, comunicação ruim com o cliente, canalização de esforços para manutenção) e os imprevisíveis são aqueles difíceis de serem identificados, mas que podem ocorrer.

Modelo de Gerência de Risco

A Taxonomia de Riscos [CARR, 1993] está baseada no Paradigma de Gerenciamento de Risco do SEI [HIGUERA, 1996], o modelo define que gerenciar riscos faz parte da essência do gerenciamento de projetos e é uma atividade que se estende ao longo de todo o projeto. O paradigma é composto por seis macro atividades: identificar, analisar, planejar, acompanhar, controlar e comunicar riscos.

Em primeiro lugar, obviamente, é necessário identificar os possíveis riscos. A seguir, deve-se analisar o risco de forma a determinar tanto a probabilidade de que ocorra como também o impacto que terá no projeto se vier a ocorrer. Logo, se o risco identificado for julgado merecedor de tratamento, deve-se planejar: o que fazer para reduzir a chance do risco ocorrer, e o que fazer quando ele vier a ocorrer. No primeiro caso, estamos falando de estratégias de mitigação, enquanto, no segundo, de planos de contingência.

Uma das tarefas do planejamento é a definição de indicadores (métricas de software) capazes de "disparar o alarme" quando um risco identificado estiver preste a ocorrer. Portanto, acompanhar, monitorar diz respeito exatamente à tarefa de se verificar continuamente estes indicadores.

Se o risco realmente ocorrer, executa-se a atividade de controlar, isto é, executar o plano de contingência anteriormente definido, e garantir a continuidade do projeto com sucesso.

Por fim, a última atividade citada: comunicar. Esta atividade é contínua, diz respeito a que toda e qualquer pessoa relacionada com o projeto tem o direito e dever de comunicar aos outros interessados qualquer coisa que diga respeito aos riscos do projeto, como por exemplo, a

identificação de novos riscos anteriormente não previstos, a detecção de sinais que indiquem que um determinado risco está acontecendo, etc.

As três primeiras atividades, identificar, analisar e planejar são realizadas ainda na fase de planejamento do projeto. São, na verdade, um dos itens do plano de projeto. Embora a responsabilidade pelo planejamento seja do gerente do projeto, todos os envolvidos com o projeto devem participar. Ainda que estas atividades sejam realizadas na fase de planejamento, é claro que, ao longo do projeto, novos riscos ainda não identificados, podem surgir e serão tratados. Esses novos riscos devem ser tratados da mesma forma, isto é, uma vez identificados, devem ser analisados e deve ser feito o planejamento necessário. Na Figura 2 é apresentado o fluxo de atividades do modelo SEI.

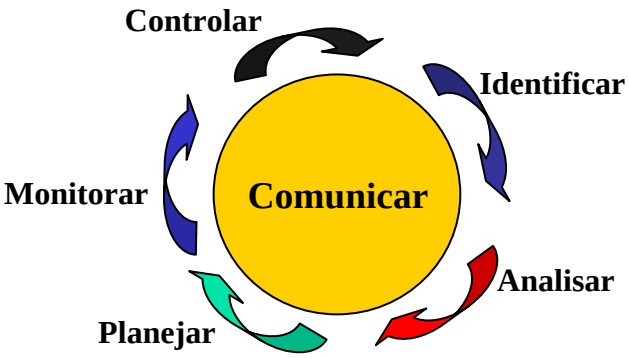

Figura 2. Gerência de Risco, segundo Modelo do SEI.

Cada possível risco, nominalmente percorre seqüencialmente todas as funções, mas as atividades ocorrem continuamente, concorrentemente (riscos são rastreados em paralelo enquanto outros riscos são identificados e analisados) e iterativamente (o plano de diminuição para um risco pode causar o aparecimento de outro risco) através do ciclo de vida do projeto.

2.2. TAXONOMIA DE RISCOS

A Taxonomia de Riscos do SEI foi desenvolvida para subsidiar a fase de identificação de riscos e está baseada na utilização de um questionário. Primeiro passo para um gerenciamento contínuo e compreensivo de riscos [CARR, 1993].

A Taxonomia de riscos está dividida em três níveis: classes, elementos e atributos conforme Figura 3. As três superclasses que compõem a taxonomia são:

- **Engenharia de Produto** – esta classe 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. As atividades incluem:
    - 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

  Os elementos cobrem as atividades da Engenharia de Software. Incluem os fatores técnicos associados com a entrega do produto final, independentemente do processo utilizado ou das limitações impostas por recursos limitados ou fatores externos fora do controle do programa.

  Riscos da Engenharia de Produto geralmente resultam de requisitos que são tecnicamente difíceis ou impossíveis de implementar, algumas vezes em combinação com a inabilidade de negociar requisitos relaxados ou orçamentos e cronogramas revisados; pela análise inadequada de requisitos ou especificação de projeto (*design*); ou pela pobre qualidade do projeto ou especificação do código.

- **Ambiente de Desenvolvimento** – esta classe endereça o ambiente do projeto em desenvolvimento e o processo utilizado. Inclui processo de desenvolvimento e sistema, métodos de gerenciamento e ambiente de trabalho.

- **Restrições de Programa** – esta classe faz referência aos fatores externos ao projeto. Fatores que normalmente estão fora do controle do projeto e mesmo assim influenciam o sucesso ou constituem fonte de riscos substanciais.

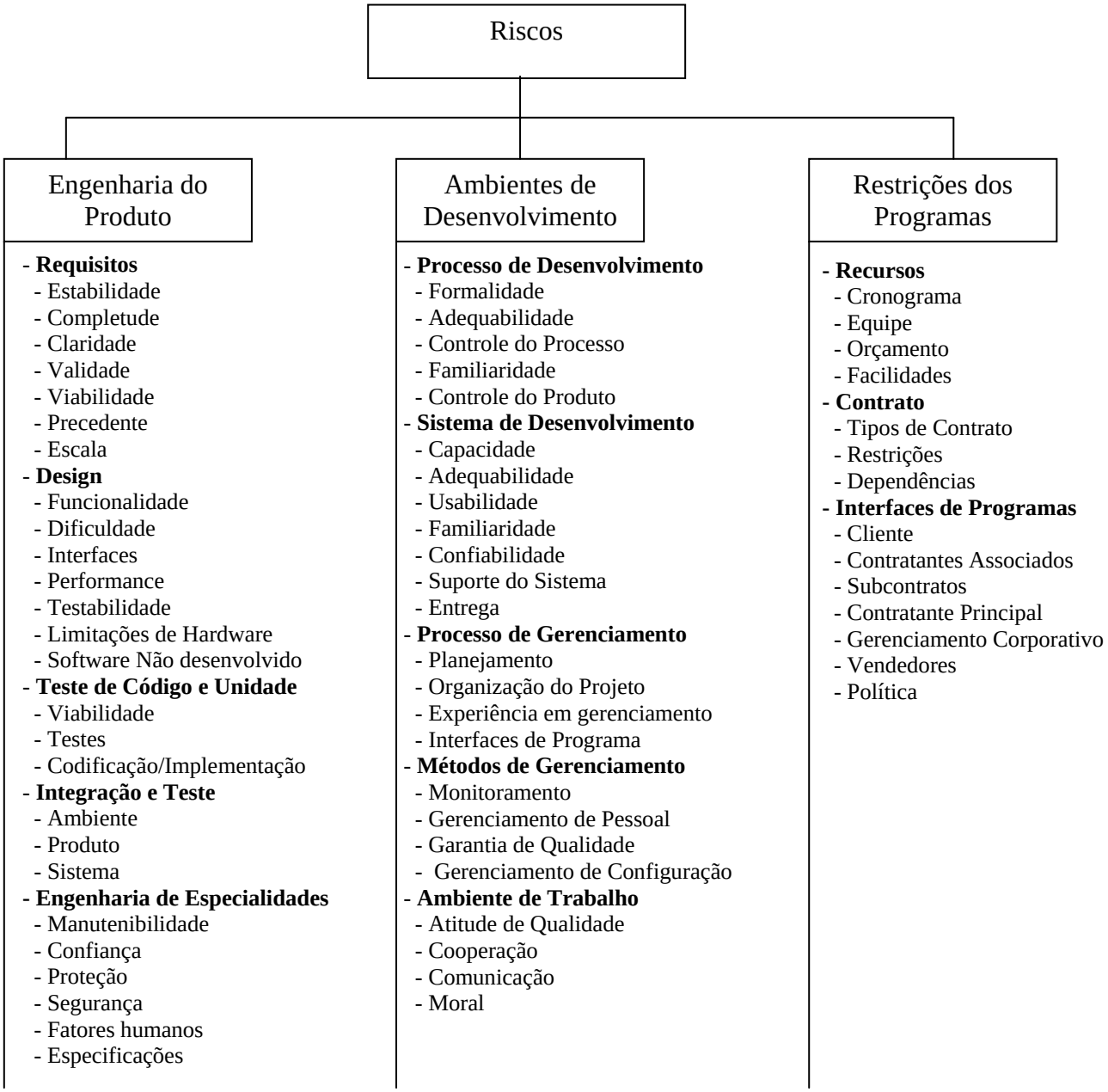

Figura 3 – Taxonomia de Riscos

Premissas de utilização do método

O foco do método de identificação de riscos através do uso da Taxonomia tem uma forte dependência com a aplicação do questionário. A utilização de um sem o outro pode ocasionar falhas.

As premissas básicas são:

- Os riscos de desenvolvimento de software são geralmente conhecidos pela equipe técnica de projeto, mas são pobremente comunicados.
- Um método estruturado e repetitivo de identificação de riscos é necessário para um gerenciamento de risco consistente.
- Identificação efetiva de riscos deve cobrir todas as áreas de desenvolvimento e suporte de um projeto.
- O processo de identificação de riscos deve criar e manter um ambiente de elicitação de riscos sem julgamentos prévios e aberto a comunicação. Desta forma terá condições de absorver várias visões dos riscos.
- Não se pode julgar o sucesso ou a falha do projeto baseado no número ou na natureza dos riscos não descobertos.

A Taxonomia é o método central de identificação de riscos, servindo com base para a elicitação e organização de grande quantidade dos riscos técnicos e não-técnicos dos ambientes de desenvolvimento de software.

A seguir, serão descritas, em linhas gerais, as subclasses (ou *elementos*, na terminologia utilizada pelo SEI) de cada uma das três principais classes.

3.2.1. Engenharia de Produto

Esta classe consiste das atividades intelectuais e físicas requeridas para desenvolver um produto e entregá-lo ao cliente. Esta classe está composta pelas seguintes subclasses:

- **Requisitos** – A definição do que o produto de software deve fazer, as necessidades que deve atingir, como deve se comportar e como será usado.
- **Projeto** (*design*) – Transformação dos requisitos no projeto do software com todas as limitações de projeto e operacionais.
- **Código e Teste Unitário** – Transformação do projeto de software no código, satisfazendo os requisitos alocados em pequenas unidades.

- **Integração e Teste** – A integração das unidades em um sistema e a validação de que o software está atendendo aos requisitos.

- **Engenharia de Especialidades** – Requisitos do produto ou atividades de desenvolvimento que necessitam de um conhecimento especializado.

3.2.2. Ambiente de Desenvolvimento

Esta classe diz respeito ao ambiente do projeto onde o software será construído, sendo composto pelas seguintes subclasses:

- **Processo de Desenvolvimento** – A definição, planejamento, documentação e comunicação dos métodos e procedimentos utilizados para desenvolver o produto.

- **Sistema de Desenvolvimento** – As ferramentas e os sistemas de suporte utilizados no desenvolvimento do produto.

- **Gerenciamento de Processo** – O planejamento, acompanhamento e controle dos orçamentos e cronogramas; a experiência do gerente de projeto no desenvolvimento do software, gerenciamento e domínio do produto.

- **Métodos de Gerenciamento** – Os métodos, ferramentas e equipamentos de suporte usados para gerenciar e controlar o desenvolvimento do produto.

- **Ambiente de Trabalho** – O local onde serão desenvolvidas as atividades de desenvolvimento do produto de software, incluindo as atitudes das pessoas, níveis de cooperação, comunicação e moral.

3.2.3. Restrições do Programa

Esta classe endereça os problemas externos ao projeto. Fatores que não são diretamente controlados pelo projeto, mas que podem influenciar no seu sucesso.

- **Recursos** – Limitações externas impostas pelo cronograma, orçamento, equipe ou facilidades.

- **Contrato** – Termos e condições do contrato de desenvolvimento do produto.

- **Interfaces dos Programas** – A interface externa com os clientes, outros contratos, gerenciamento corporativo e vendas.

CAPÍTULO 3

ONTOLOGIA DE RISCOS

Este capítulo apresenta os requisitos e os axiomas definidos para o desenvolvimento da OntoPRIME.

A definição da ontologia de riscos foi baseada, nesta primeira fase, em revisão bibliográfica da área de Engenharia de Software e na utilização dos conceitos básicos apresentados no capítulo 2.

3.1. DEFINIÇÃO E FORMALIZAÇÃO

Para iniciar a captura da ontologia, termos e frases potencialmente relevantes foram identificados e atribuídos a áreas de trabalho, que tiveram sua semântica definida e deram origem as sub-ontologias, conforme Figura 4.

Para cada uma das sub-ontologias, foram elaborados modelos utilizando a linguagem UML. Os conceitos, relacionamentos e restrições foram descritos em linguagem natural. Na formalização foi utilizada lógica de primeira ordem, definindo-se as constantes, predicados e axiomas.

Todos os predicados e axiomas são apresentados em inglês para facilitar publicações futuras, além de utilizar uma língua universalmente aceita.

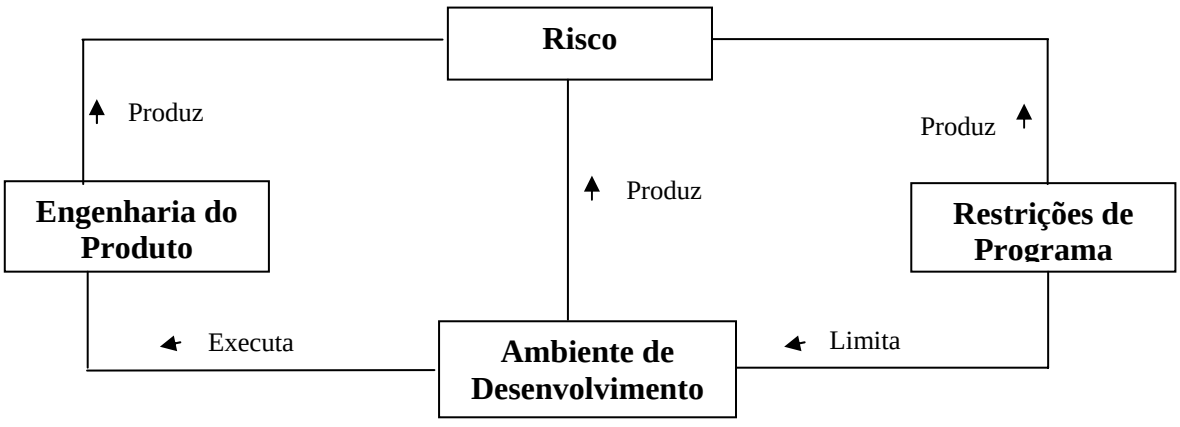

Figura 4 – Sub-ontologias da Ontologia de Risco

Para formalizar as sub-ontologias da Ontologia de Risco foram criados os seguintes predicados, que fazem referência as Classes da Taxonomia de Riscos, secção 3.2.: ***Risk(p),*** ***Project(p),*** ***ProductEngineeringRisk(p)*** (dado que ***Project(p)***), ***DevelopmentEnvironmentRisk(p)*** e ***ProgramConstraintsRisk(p)***.

Além disso, foram definidos os seguintes axiomas para representar os relacionamentos de **especialização / generalização**:

- *Se um projeto possui **risco de engenharia de produto**, então este projeto possui um **risco**.*
  - $\forall p.\ \text{Project}(p) \wedge \text{ProductEngineeringRisk}(p) \rightarrow \text{Risk}(p)$
- *Se um projeto possui **risco de ambiente de desenvolvimento**, então este projeto possui um **risco**.*
  - $\forall p.\ \text{Project}(p) \wedge \text{DevelopmentEnvironment}(p) \rightarrow \text{Risk}(p)$
- *Se um projeto possui **risco de restrições de programa**, então este projeto possui um **risco**.*
  - $\forall p.\ \text{Project}(p) \wedge \text{ProgramConstraints}(p) \rightarrow \text{Risk}(p)$
- *Se um projeto possui **risco de requisitos**, então este projeto possui **risco de engenharia de produto**.*
  - $\forall p.\ \text{Project}(p) \wedge \text{RequirementsRisk}(p) \rightarrow \text{ProductEngineeringRisk}\ (p)$
- *Se um projeto possui **risco de design**, então este projeto possui **risco de engenharia de produto**.*
  - $\forall p.\ \text{Project}(p) \wedge \text{DesignRisk}(p) \rightarrow \text{ProductEngineeringRisk}\ (p)$
- *Se um projeto possui **risco de código e teste unitário**, então este projeto possui **risco de engenharia de produto**.*
  - $\forall p.\ \text{Project}(p) \wedge \text{CodeAndUnitTestRisk}(p) \rightarrow \text{ProductEngineeringRisk}\ (p)$

- *Se um projeto possui **risco de integração e teste**, então este projeto possui **risco de engenharia de produto**.*
  - $\forall p.\ Project(p) \wedge IntegrationAndTestRisk(p) \rightarrow ProductEngineeringRisk\ (p)$
- *Se um projeto possui **risco de especialidades de engenharia**, então este projeto possui **risco de engenharia de produto**.*
  - $\forall p.\ Project(p) \wedge EngineeringSpecialitiesRisk(p) \rightarrow ProductEngineeringRisk\ (p)$
- *Se um projeto possui **risco de estabilidade de requisitos**, então este projeto possui **risco de requisitos**.*
  - $\forall p.\ Project(p) \wedge StabilityRisk(p) \rightarrow RequirementsRisk\ (p)$
- *Se um projeto possui **risco de completude de requisitos**, então este projeto possui **risco de requisitos**.*
  - $\forall p.\ Project(p) \wedge CompletenessRisk(p) \rightarrow RequirementsRisk\ (p)$
- *Se um projeto possui **risco de clareza de requisitos**, então este projeto possui **risco de requisitos**.*
  - $\forall p.\ Project(p) \wedge ClarityRisk(p) \rightarrow RequirementsRisk\ (p)$
- *Se um projeto possui **risco de validade de requisitos**, então este projeto possui **risco de requisitos**.*
  - $\forall p.\ Project(p) \wedge ValidityRisk(p) \rightarrow RequirementsRisk\ (p)$
- *Se um projeto possui **risco de viabilidade de requisitos**, então este projeto possui **risco de requisitos**.*
  - $\forall p.\ Project(p) \wedge FeasibilityRisk(p) \rightarrow RequirementsRisk\ (p)$
- *Se um projeto possui **risco de precedentes**, então este projeto possui **risco de requisitos**.*

- $\forall p.\ Project(p) \wedge PrecedentRisk(p) \rightarrow RequirementsRisk\ (p)$

- *Se um projeto possui **risco de escala**, então este projeto possui **risco de requisitos**.*
  - $\forall p.\ Project(p) \wedge ScaleRisk(p) \rightarrow RequirementsRisk\ (p)$
- *Se um projeto possui **risco de funcionalidade**, então este projeto possui **risco de design**.*
  - $\forall p.\ Project(p) \wedge FunctionalityRisk(p) \rightarrow DesignRisk\ (p)$
- *Se um projeto possui **risco de dificuldade**, então este projeto possui **risco de design**.*
  - $\forall p.\ Project(p) \wedge DifficultyRisk(p) \rightarrow DesignRisk\ (p)$
- *Se um projeto possui **risco de interfaces**, então este projeto possui **risco de design**.*
  - $\forall p.\ Project(p) \wedge InterfacesRisk(p) \rightarrow DesignRisk\ (p)$
- *Se um projeto possui **risco de performance**, então este projeto possui **risco de design**.*
  - $\forall p.\ Project(p) \wedge PerformanceRisk(p) \rightarrow DesignRisk\ (p)$
- *Se um projeto possui **risco de testabilidade**, então este projeto possui **risco de design**.*
  - $\forall p.\ Project(p) \wedge TestabilityRisk(p) \rightarrow DesignRisk\ (p)$
- *Se um projeto possui **risco de restrições de hardware**, então este projeto possui **risco de design**.*
  - $\forall p.\ Project(p) \wedge HardwareConstraintsRisk(p) \rightarrow DesignRisk\ (p)$
- *Se um projeto possui **risco de software externo**, então este projeto possui **risco de design**.*
  - $\forall p.\ Project(p) \wedge NonDevelopmentalSoftwareRisk(p) \rightarrow DesignRisk\ (p)$
- *Se um projeto possui **risco de viabilidade de implementação**, então este projeto possui **risco de código e teste unitário**.*

- $\forall p.\ Project(p) \wedge ImplementationFeasibilityRisk(p) \rightarrow CodeAndUnitTestRisk\ (p)$

- *Se um projeto possui* ***risco de teste****, então este projeto possui* ***risco de código e teste unitário****.*

  - $\forall p.\ Project(p) \wedge TestingRisk(p) \rightarrow CodeAndUnitTestRisk\ (p)$

- *Se um projeto possui* ***risco de implementação****, então este projeto possui* ***risco de código e teste unitário****.*

  - $\forall p.\ Project(p) \wedge ImplementationRisk(p) \rightarrow CodeAndUnitTestRisk\ (p)$

- *Se um projeto possui* ***risco de ambiente de integração e teste****, então este projeto possui* ***risco de integração e teste****.*

  - $\forall p.\ Project(p) \wedge IntegrationAndTestEnvironmentRisk\ (p) \rightarrow IntegrationAndTestRisk\ (p)$

- *Se um projeto possui* ***risco de integração e teste de produto****, então este projeto possui* ***risco de integração e teste****.*

  - $\forall p.\ Project(p) \wedge ProductIntegrationAndTestRisk(p) \rightarrow IntegrationAndTestRisk\ (p)$

- *Se um projeto possui* ***risco de integração e teste de sistema****, então este projeto possui* ***risco de integração e teste****.*

  - $\forall p.\ Project(p) \wedge SystemIntegrationAndTestRisk(p) \rightarrow IntegrationAndTestRisk\ (p)$

- *Se um projeto possui* ***risco de processo de desenvolvimento****, então este projeto possui* ***risco de ambiente de desenvolvimento****.*

  - $\forall p.\ Project(p) \wedge DevelopmentProcessRisk(p) \rightarrow DevelopmentEnvironmentRisk\ (p)$

- *Se um projeto possui* ***risco de sistema de desenvolvimento****, então este projeto possui* ***risco de ambiente de desenvolvimento****.*

  - $\forall p.\ Project(p) \wedge DevelopmentSystemRisk(p) \rightarrow DevelopmentEnvironmentRisk\ (p)$

- *Se um projeto possui* ***risco de processo de gerenciamento****, então este projeto possui* ***risco de ambiente de desenvolvimento****.*
  - $\forall p.\ Project(p) \wedge ManagementProcessRisk(p) \rightarrow DevelopmentEnvironmentRisk\ (p)$
- *Se um projeto possui* ***risco de métodos de gerenciamento****, então este projeto possui* ***risco de ambiente de desenvolvimento****.*
  - $\forall p.\ Project(p) \wedge ManagementMethodsRisk(p) \rightarrow DevelopmentEnvironmentRisk\ (p)$
- *Se um projeto possui* ***risco de ambiente de trabalho****, então este projeto possui* ***risco de ambiente de desenvolvimento****.*
  - $\forall p.\ Project(p) \wedge WorkEnvironmentRisk(p) \rightarrow DevelopmentEnvironmentRisk\ (p)$
- *Se um projeto possui* ***risco de formalidade de processo de desenvolvimento****, então este projeto possui* ***risco de processo de desenvolvimento****.*
  - $\forall p.\ Project(p) \wedge DevelopmentProcessFormalityRisk(p) \rightarrow DevelopmentProcessRisk\ (p)$
- *Se um projeto possui* ***risco de adequabilidade de processo de desenvolvimento****, então este projeto possui* ***risco de processo de desenvolvimento****.*
  - $\forall p.\ Project(p) \wedge DevelopmentProcessSuitabilityRisk(p) \rightarrow DevelopmentProcessRisk\ (p)$
- *Se um projeto possui* ***risco de controle de processo de desenvolvimento****, então este projeto possui* ***risco de processo de desenvolvimento****.*
  - $\forall p.\ Project(p) \wedge DevelopmentProcessControlRisk(p) \rightarrow DevelopmentProcessRisk\ (p)$
- *Se um projeto possui* ***risco de familiaridade com processo de desenvolvimento****, então este projeto possui* ***risco de processo de desenvolvimento****.*

- $\forall p.\ Project(p) \wedge DevelopmentProcessFamiliarityRisk(p) \rightarrow DevelopmentProcessRisk\ (p)$

- *Se um projeto possui* ***risco de controle de produto****, então este projeto possui* ***risco de processo de desenvolvimento****.*

  - $\forall p.\ Project(p) \wedge ProductControlRisk(p) \rightarrow DevelopmentProcessRisk\ (p)$

- *Se um projeto possui* ***risco de capacidade de sistema de desenvolvimento****, então este projeto possui* ***risco de sistema de desenvolvimento****.*

  - $\forall p.\ Project(p) \wedge CapacityRisk(p) \rightarrow DevelopmentSystemRisk\ (p)$

- *Se um projeto possui* ***risco de adequabilidade de sistema de desenvolvimento****, então este projeto possui* ***risco de sistema de desenvolvimento****.*

  - $\forall p.\ Project(p) \wedge DevelopmentSystemSuitabilityRisk(p) \rightarrow DevelopmentSystemRisk\ (p)$

- *Se um projeto possui* ***risco de usabilidade de sistema de desenvolvimento****, então este projeto possui* ***risco de sistema de desenvolvimento****.*

  - $\forall p.\ Project(p) \wedge DevelopmentSystemUsabilityRisk(p) \rightarrow DevelopmentSystemRisk\ (p)$

- *Se um projeto possui* ***risco de familiaridade com sistema de desenvolvimento****, então este projeto possui* ***risco de sistema de desenvolvimento****.*

  - $\forall p.\ Project(p) \wedge DevelopmentSystemFamiliarityRisk(p) \rightarrow DevelopmentSystemRisk\ (p)$

- *Se um projeto possui* ***risco de confiabilidade de sistema de desenvolvimento****, então este projeto possui* ***risco de sistema de desenvolvimento****.*

  - $\forall p.\ Project(p) \wedge DevelopmentSystemReliabilityRisk(p) \rightarrow DevelopmentSystemRisk\ (p)$

- *Se um projeto possui* ***risco de suporte de sistema de desenvolvimento****, então este projeto possui* ***risco de sistema de desenvolvimento****.*
  - $\forall p.\ Project(p) \wedge DevelopmentSystemSupportRisk(p) \rightarrow DevelopmentSystemRisk\ (p)$
- *Se um projeto possui* ***risco de entrega****, então este projeto possui* ***risco de sistema de desenvolvimento****.*
  - $\forall p.\ Project(p) \wedge DeliverabilityRisk(p) \rightarrow DevelopmentSystemRisk\ (p)$
- *Se um projeto possui* ***risco de planejamento****, então este projeto possui* ***risco de processo de gerenciamento****.*
  - $\forall p.\ Project(p) \wedge PlanningRisk(p) \rightarrow ManagementProcessRisk\ (p)$
- *Se um projeto possui* ***risco de organização do projeto****, então este projeto possui* ***risco de processo de gerenciamento****.*
  - $\forall p.\ Project(p) \wedge ProjectOrganizationRisk(p) \rightarrow ManagementProcessRisk\ (p)$
- *Se um projeto possui* ***risco de experiência de gerenciamento****, então este projeto possui* ***risco de processo de gerenciamento****.*
  - $\forall p.\ Project(p) \wedge ManagementExperienceRisk(p) \rightarrow ManagementProcessRisk\ (p)$
- *Se um projeto possui* ***risco de interfaces de programa****, então este projeto possui* ***risco de processo de gerenciamento****.*
  - $\forall p.\ Project(p) \wedge ProgramInterfacesRisk(p) \rightarrow ManagementProcessRisk\ (p)$
- *Se um projeto possui* ***risco de monitoração****, então este projeto possui* ***risco de métodos de gerenciamento****.*
  - $\forall p.\ Project(p) \wedge MonitoringRisk(p) \rightarrow ManagementMethodsRisk\ (p)$
- *Se um projeto possui* ***risco de gerenciamento de pessoal****, então este projeto possui* ***risco de métodos de gerenciamento****.*

- $\forall p.\ Project(p) \wedge PersonnelManagementRisk(p) \rightarrow ManagementMethodsRisk\ (p)$

- *Se um projeto possui* ***risco de garantia de qualidade****, então este projeto possui* ***risco de métodos de gerenciamento****.*

  - $\forall p.\ Project(p) \wedge QualityAssuranceRisk(p) \rightarrow ManagementMethodsRisk\ (p)$

- *Se um projeto possui* ***risco de gerência de configuração****, então este projeto possui* ***risco de métodos de gerenciamento****.*

  - $\forall p.\ Project(p) \wedge ConfigurationManagementRisk(p) \rightarrow ManagementMethodsRisk\ (p)$

- *Se um projeto possui* ***risco de atitude de qualidade****, então este projeto possui* ***risco de ambiente de trabalho****.*

  - $\forall p.\ Project(p) \wedge QualityAttitudeRisk(p) \rightarrow WorkEnvironmentRisk\ (p)$

- *Se um projeto possui* ***risco de cooperação****, então este projeto possui* ***risco de ambiente de trabalho****.*

  - $\forall p.\ Project(p) \wedge CooperationRisk(p) \rightarrow WorkEnvironmentRisk\ (p)$

- *Se um projeto possui* ***risco de comunicação****, então este projeto possui* ***risco de ambiente de trabalho****.*

  - $\forall p.\ Project(p) \wedge CommunicationRisk(p) \rightarrow WorkEnvironmentRisk\ (p)$

- *Se um projeto possui* ***risco de moral****, então este projeto possui* ***risco de ambiente de trabalho****.*

  - $\forall p.\ Project(p) \wedge MoraleRisk(p) \rightarrow WorkEnvironmentRisk\ (p)$

- *Se um projeto possui* ***risco de recursos****, então este projeto possui* ***risco de restrições de programa****.*

  - $\forall p.\ Project(p) \wedge ResourcesRisk(p) \rightarrow ProgramConstraintsRisk\ (p)$

- *Se um projeto possui **risco de contrato**, então este projeto possui **risco de restrições de programa**.*
  - $\forall p.\ Project(p) \wedge ContractRisk(p) \rightarrow ProgramConstraintsRisk\ (p)$
- *Se um projeto possui **risco de interfaces de restrições de programa**, então este projeto possui **risco de restrições de programa**.*
  - $\forall p.\ Project(p) \wedge ProgramConstraintsProgramInterfacesRisk(p) \rightarrow ProgramConstraintsRisk\ (p)$
- *Se um projeto possui **risco de cronograma**, então este projeto possui **risco de recursos**.*
  - $\forall p.\ Project(p) \wedge ScheduleRisk(p) \rightarrow ResourcesRisk\ (p)$
- *Se um projeto possui **risco de pessoal**, então este projeto possui **risco de recursos**.*
  - $\forall p.\ Project(p) \wedge StaffRisk(p) \rightarrow ResourcesRisk\ (p)$
- *Se um projeto possui **risco de orçamento**, então este projeto possui **risco de recursos**.*
  - $\forall p.\ Project(p) \wedge BufgetRisk(p) \rightarrow ResourcesRisk\ (p)$
- *Se um projeto possui **risco de facilidades**, então este projeto possui **risco de recursos**.*
  - $\forall p.\ Project(p) \wedge FacilitiesRisk(p) \rightarrow ResourcesRisk\ (p)$
- *Se um projeto possui **risco de tipo de contrato**, então este projeto possui **risco de contrato**.*
  - $\forall p.\ Project(p) \wedge TypeOfContractRisk(p) \rightarrow ContractRisk\ (p)$
- *Se um projeto possui **risco de restrições**, então este projeto possui **risco de contrato**.*
  - $\forall p.\ Project(p) \wedge RestrictionsRisk(p) \rightarrow ContractRisk\ (p)$
- *Se um projeto possui **risco de dependências**, então este projeto possui **risco de contrato**.*

- $\forall p.\ Project(p) \wedge DependenciesRisk(p) \rightarrow ContractRisk\ (p)$

- *Se um projeto possui* ***risco de cliente****, então este projeto possui* ***risco de interfaces de de restrições de programa****.*

  - $\forall p.\ Project(p) \wedge CustomerRisk(p) \rightarrow ProgramConstraintsProgramInterfacesRisk(p)$

- *Se um projeto possui* ***risco de contrato associado****, então este projeto possui* ***risco de interfaces de restrições de programa****.*

  - $\forall p.\ Project(p) \wedge AssociateContractorsRisk(p) \rightarrow ProgramConstraintsProgramInterfacesRisk(p)$

- *Se um projeto possui* ***risco de sub-contratação****, então este projeto possui* ***risco de interfaces de restrições de programa****.*

  - $\forall p.\ Project(p) \wedge SubcontractorsRisk(p) \rightarrow ProgramConstraintsProgramInterfacesRisk(p)$

- *Se um projeto possui* ***risco de contrato principal****, então este projeto possui* ***risco de interfaces de restrições de programa****.*

  - $\forall p.\ Project(p) \wedge PrimeContractorRisk(p) \rightarrow ProgramConstraintsProgramInterfacesRisk(p)$

- *Se um projeto possui* ***risco de administração corporativa****, então este projeto possui* ***risco de interfaces de restrições de programa****.*

  - $\forall p.\ Project(p) \wedge CorporateManagementRisk(p) \rightarrow ProgramConstraintsProgramInterfacesRisk(p)$

- *Se um projeto possui* ***risco de vendedores****, então este projeto possui* ***risco de interfaces de programa de restrições de programa****.*

  - $\forall p.\ Project(p) \wedge VendorsRisk(p) \rightarrow ProgramConstraintsProgramInterfacesRisk(p)$

- *Se um projeto possui **risco de política**, então este projeto possui **risco de interfaces de programa de restrições de programa**.*
    - $\forall p.\ Project(p) \wedge PoliticsRisk(p) \rightarrow ProgramConstraintsProgramInterfacesRisk(p)$

Nas próximas seções, serão definidos os axiomas para cada classe e subclasse da taxonomia.

3.2.1. Engenharia de Produto

Para a Classe ***Risco de Requisitos*** e suas subclasses foram definidos os seguintes axiomas:

Estabilidade:

- *Se existe algum projeto P no qual o requisito muda durante o processo de desenvolvimento, então existe **risco de estabilidade de requisitos** para P.*
    - $\exists\ p, y\ .\ \text{Project}(p) \wedge \text{Requirement}(y, p) \wedge \text{Change}(y) \rightarrow \text{RequirementsStabilityRisk}(p)$
- *Se existe algum projeto P que possui uma interface externa e esta interface muda, então existe **risco de estabilidade de requisitos** para P.*
    - $\exists\ p, i\ .\ \text{Project}(p) \wedge \text{Interface}(i, p) \wedge \text{Change}(i) \rightarrow \text{RequirementsStabilityRisk}(p)$

Completude:

- *Se existe algum projeto P que possui requisito e este requisito não está completamente definido , então existe **risco de completude de requisitos** para P.*
    - $\exists\ p, y\ .\ \text{Project}(p) \wedge \text{Requirement}(y, p) \wedge \neg\ \text{CompletelyDefined}(y) \rightarrow \text{RequirementsCompletenessRisk}(p)$
- *Se existe algum projeto P que possui um requisito e este requisito não está completamente especificado, então existe **risco de completude de requisitos** para P.*

- $\exists\, p, y$. Project(p) $\wedge$ Requirement(y, p) $\wedge \neg$ CompletelySpecified(y) $\rightarrow$ RequirementsCompletenessRisk(p)

- *Se existe algum projeto P que possui uma interface externa e esta interface não está completamente definida, então existe* ***risco de completude de requisitos*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ ExternalInterface(x, p) $\wedge \neg$ CompletelyDefined(x) $\rightarrow$ RequirementsCompletenessRisk(*p*

Clareza:

- *Se existe algum projeto P que possui requisito e este requisito não está claro, então existe* ***risco de clareza de requisitos*** *para P.*

  - $\exists\, p, y$. Project(p) $\wedge$ Requirement(y, p) $\wedge \neg$ Clarity(y) $\rightarrow$ RequirementsClarityRisk(p)

- *Se existe algum projeto P que possui um requisito e este requisito precisa de interpretação, então existe* ***risco de clareza de requisitos*** *para P.*

  - $\exists\, p, y$. Project(p) $\wedge$ Requirement(y, p) $\wedge$ Interpretation(y) $\rightarrow$ RequirementsClarityRisk(p)

Validade:

- *Se existe algum projeto P que possui requisito e este requisito não representa a necessidade do usuário, então existe* ***risco de validade de requisitos*** *para P.*

  - $\exists\, p, y$. Project(p) $\wedge$ Requirement(y, p) $\wedge \neg$ RepresentsCustomerNeed(y) $\rightarrow$ RequirementsValidityRisk(p)

- *Se existe algum projeto P que possui requisito e este requisito não tem o mesmo entendimento para o cliente e desenvolvedor, então existe* ***risco de validade de requisitos*** *para P.*

  - $\exists\, p, y$. Project(p) $\wedge$ Requirement(y, p) $\wedge$ NotUnderstood(y) $\rightarrow$ RequirementsValidityRisk(p)

Viabilidade:

- *Se existe algum projeto P que possui requisito e este requisito é inviável sob o ponto de vista de análise, então existe* ***risco de viabilidade de requisitos*** *para P.*

  - $\exists\, p, y\ .\ \text{Project}(p) \wedge \text{Requirement}(y, p) \wedge \neg\ \text{Feasible}(y) \rightarrow \text{RequirementsFeasibilityRisk}(p)$

- *Se existe algum projeto P que possui requisito e este requisito é tecnicamente difícil de implementar, então existe* ***risco de viabilidade de requisitos*** *para P.*

  - $\exists\, p, y\ .\ \text{Project}(p) \wedge \text{Requirement}(y, p) \wedge \neg\ \text{TechnicalDificult}(y) \rightarrow \text{RequirementsFeasibilityRisk}(p)$

Precedente:

- *Se existe algum projeto P que possui requisito e este requisito nunca foi desenvolvido anteriormente, então existe* ***risco de precedente*** *para P.*

  - $\exists\, p, y\ .\ \text{Project}(p) \wedge \text{Requirement}(y, p) \wedge \text{NeverDone}(y) \rightarrow \text{PrecedentRisk}(p)$

- *Se existe algum projeto P que possui requisito e este requisito especifica algo nunca antes realizado, então existe* ***risco de precedente*** *para P.*

  - $\exists\, p, y\ .\ \text{Project}(p) \wedge \text{Requirement}(y, p) \wedge \text{SpecifiedNeverDone}(y) \rightarrow \text{PrecedentRisk}(p)$

Escala:

- *Se existe algum projeto P para o qual será desenvolvido um produto complexo, então existe* ***risco de escala*** *para P.*

  - $\exists\, p, z\ .\ \text{Project}(p) \wedge \text{Product}(z, p) \wedge \text{Complex}(z) \rightarrow \text{ScaleRisk}(p)$

- *Se existe algum projeto P para o qual será desenvolvido um produto grande, então existe* ***risco de escala*** *para P.*

  - $\exists\, p, z\ .\ \text{Project}(p) \wedge \text{Product}(z, p) \wedge \text{Large}(z) \rightarrow \text{ScaleRisk}(p)$

- *Se existe algum projeto P para o qual será desenvolvido um produto que a organização não tem experiência, então existe **risco de escala** para P.*

  - $\exists\, p, z\,.\ \text{Project}(p) \wedge \text{Product}(z, p) \wedge \neg\, \text{CompanyExperience}(z) \rightarrow \text{ScaleRisk}(p)$

Para a Classe ***Risco de Design*** e suas subclasses foram definidos os seguintes axiomas:

Funcionalidade:

- *Se existe algum projeto P no qual existe um algoritmo que não satisfaz ao requisito para o qual este algoritmo foi projetado, então existe **risco de funcionalidade** para P.*

  - $\exists\, p, x, y\,.\ \text{Project}(p) \wedge \text{Algorithm}(x) \wedge \text{Requirement}(y, p) \wedge \neg\, \text{Satisfacts}(x,y) \rightarrow \text{FunctionalityRisk}(p)$

Dificuldade:

- *Se existe algum projeto P no qual existe um requisito cujo design não é fácil, então existe **risco de dificuldade** para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Requirement}(x, p) \wedge \neg\, \text{EasyDesign}(x) \rightarrow \text{DificultyRisk}(p)$

- *Se existe algum projeto P no qual existe um requisito para o qual não existe solução, então existe **risco de dificuldade** para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Requirement}(x, p) \wedge \neg\, \exists\, y\,.\ \text{Solution}(y, x) \rightarrow \text{DificultyRisk}(p)$

Interfaces:

- *Se existe algum projeto P que possui uma interface interna e esta interface não está completamente definida, então existe **risco de interface** para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Interface}(x, p) \wedge \neg\, \text{Defined}(x) \rightarrow \text{InterfaceRisk}(p)$

- *Se existe algum projeto P para o qual não há um processo para definição de interfaces internas, então existe **risco de interface** para P.*

  - $\exists\, p\,.\ \text{Project}(p) \wedge \neg\, \exists\, x\,.\ \text{ProcessForDefiningInternalInterfaces}(x, p) \rightarrow \text{InterfaceRisk}(p)$

- *Se existe algum projeto P para o qual há um processo para definição de interfaces internas mas não há um processo para controle de mudanças para interfaces internas, então existe* ***risco de interface*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge \exists\, x$. ProcessForDefiningInternalInterfaces(x, p) $\wedge \neg \exists\, y$. ChangeControlProcessForInternalInterfaces(y, p) $\rightarrow$ InterfaceRisk(p)

- *Se existe algum projeto P para o qual existe hardware sendo desenvolvido em paralelo com software e as especificações deste hardware estão mudando, então existe* ***risco de interface*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ Hardware(x, p) $\wedge$ BeingDeveloped(x) $\wedge$ SpecificationIsChanging(x) $\rightarrow$ InterfaceRisk(x)

- *Se existe algum projeto P para o qual existe hardware sendo desenvolvido em paralelo com software e existe uma interface com software para este hardware que não está completamente definida, então existe* ***risco de interface*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ Hardware(x, p) $\wedge$ BeingDeveloped(x) $\wedge \exists\, y$. InterfaceToSoftware(y, x, p) $\wedge \neg$ Defined(y) $\rightarrow$ InterfaceRisk(p)

Performance:

- *Se existe um projeto P para o qual existe um requisito de vazão mínima e a vazão obtida é inferior ao valor esperado para este requisito, então existe* ***risco de performance*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ ThroughputRequirement(x, p) $\wedge$ Throughput(p) < Value(x) $\rightarrow$ PerformanceRisk(p)

- *Se existe um projeto P para o qual existe um evento de tempo real assíncrono que não está sendo escalonado corretamente, então existe* ***risco de performance*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ AsynchronousRealTimeEvent(x, p) $\wedge \neg$ CorrecltyScheduled(x) $\rightarrow$ PerformanceRisk(p)

- *Se existe um projeto P para o qual existe um requisito de tempo máximo de recuperação para uma determinada falha e o tempo de recuperação para esta falha é superior ao valor do requisito, então existe* ***risco de performance*** *para P.*

  - $\exists\, p, x, y\ .\ \text{Project}(p) \wedge \text{Failure}(x, p) \wedge \text{RecoveryTimelineRequirement}(y, x, p)\ \text{RecoveryTimeline}(x, p) < \text{Value}(y) \rightarrow \text{PerformanceRisk}(p)$

- *Se existe um projeto P para o qual existe um requisito de tempo máximo de resposta e o tempo médio de resposta é superior ao valor do requisito, então existe* ***risco de performance*** *para P.*

  - $\exists\, p, x\ .\ \text{Project}(p) \wedge \text{ResponseTimeRequirement}(x, p) \wedge \text{AverageResponseTime}(p) < \text{Value}(x) \rightarrow \text{PerformanceRisk}(p)$

- *Se existe um projeto P para o qual existe um requisito de tempo máximo de resposta do banco de dados e o tempo médio de resposta do banco de dados é superior ao valor do requisito, então existe* ***risco de performance*** *para P.*

  - $\exists\, p, x\ .\ \text{Project}(p) \wedge \text{DatabaseResponseTimeRequirement}(x, p) \wedge \text{AverageDatabaseResponseTime}(p) < \text{Value}(x) \rightarrow \text{PerformanceRisk}(p)$

- *Se existe um projeto P para o qual não foi realizada uma análise de performance, então existe* ***risco de performance*** *para P.*

  - $\exists\, p\ .\ \text{Project}(p) \wedge \neg\ \text{PerformanceAnalysisDone}(p) \rightarrow \text{PerformanceRisk}(p)$

- *Se existe um projeto P para o qual foi realizada uma análise de performance, porém não existe um modelo de acompanhamento de performance, então existe* ***risco de performance*** *para P.*

  - $\exists\, p\ .\ \text{Project}(p) \wedge \text{PerformanceAnalysisDone}(p) \wedge \neg\ \exists\, x.\ \text{ModelToTrackPerformance}(x, p) \rightarrow \text{PerformanceRisk}(p)$

Testabilidade:

- *Se existe um projeto P para o qual não existe um processo de testes que seja fácil de executar, então existe **risco de testabilidade** para P.*

  - $\exists\, p\, .\, \text{Project}(p) \wedge \neg \exists\, x\, .\, (\text{TestProcess}(x, p) \wedge \text{Easy}(x)) \rightarrow \text{TestabilityRisk}(p)$

- *Se existe um projeto P para o qual não existe um elemento que auxilie os testes, então existe **risco de testabilidade** para P.*

  - $\exists\, p\, .\, \text{Project}(p) \wedge \neg \exists\, x\, .\, \text{AidsTest}(x, p) \rightarrow \text{TestabilityRisk}(p)$

- *Se existe um projeto P no qual os indivíduos responsáveis pelos testes não participarem da análise de requisitos, então existe **risco de testabilidade** para P.*

  - $\exists\, p\, .\, \text{Project}(p) \wedge \neg \exists\, x\, .\, (\text{Person}(x) \wedge \text{PerformsTest}(x, p) \wedge \text{GetsInvolvedInAnalyzingRequirements}(x, p)) \rightarrow \text{TestabilityRisk}(p)$

Limitação de Hardware:

- *Se existe um projeto P que utiliza um hardware que limita a arquitetura de P, então existe **risco de limitação de hardware** para P.*

  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \text{LimitsArchitecture}(x, p) \rightarrow \text{HardwareConstraintsRisk}(p)$

- *Se existe um projeto P que utiliza um hardware que diminui a capacidade de memória desejada para P, então existe **risco de limitação de hardware** para P.*

  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \text{LimitsDesiredMemoryCapacity}(x, p) \rightarrow \text{HardwareConstraintsRisk}(p)$

- *Se existe um projeto P que utiliza um hardware que diminui a vazão de P, então existe **risco de limitação de hardware** para P.*

  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \text{LimitsThroughput}(x, p) \rightarrow \text{HardwareConstraintsRisk}(p)$

- *Se existe um projeto P que utiliza um hardware que aumenta o tempo de resposta a eventos de tempo real de P, então existe* ***risco de limitação de hardware*** *para P.*

  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \text{IncreasesRealTimeEventsResponseTime}(x, p) \rightarrow \text{HardwareConstraintsRisk}(p)$

- *Se existe um projeto P que utiliza um hardware que aumenta o tempo médio de resposta de P, então existe* ***risco de limitação de hardware*** *para P.*

  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \text{IncreasesResponseTime}(x, p) \rightarrow \text{HardwareConstraintsRisk}(p)$

- *Se existe um projeto P que utiliza um hardware que aumenta o tempo de recuperação após falha de P, então existe* ***risco de limitação de hardware*** *para P.*

  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \text{IncreasesRecoveryTimeline}(x, p) \rightarrow \text{HardwareConstraintsRisk}(p)$

- *Se existe um projeto P que utiliza um hardware que diminui a performance do banco de dados usado por P, então existe* ***risco de limitação de hardware*** *para P.*

  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \text{LimitsDatabasePerformance}(x, p) \rightarrow \text{HardwareConstraintsRisk}(p)$

- *Se existe um projeto P que utiliza um hardware que impede a implementação de uma funcionalidade de P, então existe* ***risco de limitação de hardware*** *para P.*

  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \exists\, y\, .\, (\text{Functionality}(y, p) \wedge \text{Limits}\,(x, y)) \rightarrow \text{HardwareConstraintsRisk}(p)$

- *Se existe um projeto P que utiliza um hardware que diminui a confiabilidade de P, então existe* ***risco de limitação de hardware*** *para P.*

  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \text{LimitsReliability}(x, p) \rightarrow \text{HardwareConstraintsRisk}(p)$

- *Se existe um projeto P que utiliza um hardware que diminui a disponibilidade de P, então existe* ***risco de limitação de hardware*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Hardware}(x, p) \wedge \text{LimitsAvailability}(x, p) \rightarrow \text{HardwareConstraintsRisk}(p)$

Software externo (não desenvolvido pela equipe):

- *Se existe um projeto P que utiliza um software não desenvolvido pelo programa que não é bem documentado, então existe* ***risco de software externo*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{NonDevelopmentalSoftware}(x, p) \wedge \text{Uses}(p, x) \wedge \neg \text{Documented}(x) \rightarrow \text{NonDevelopmentalSoftwareRisk}(p)$

- *Se existe um projeto P que utiliza um software não desenvolvido pelo programa que diminui a performance de P, então existe* ***risco de software externo*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{NonDevelopmentalSoftware}(x, p) \wedge \text{Uses}(p, x) \wedge \text{LimitsPerformance}(x, p) \rightarrow \text{NonDevelopmentalSoftwareRisk}(p)$

- *Se existe um projeto P que utiliza um software não desenvolvido pelo programa que compromete uma funcionalidade de P, então existe* ***risco de software externo*** *para P.*

  - $\exists\, p, x, y\,.\ \text{Project}(p) \wedge \text{NonDevelopmentalSoftware}(x, p) \wedge \text{Uses}(p, x) \wedge \text{Functionality}(y, p) \wedge \text{Limits}(x, y) \rightarrow \text{NonDevelopmentalSoftwareRisk}(p)$

- *Se existe um projeto P que utiliza um software não desenvolvido pelo programa e que cujo tempo de entrega é superior ao tempo de entrega estabelecido, então existe* ***risco de software externo*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{NonDevelopmentalSoftware}(x, p) \wedge \text{Uses}(p, x) \wedge \text{DeliveryTime}(x, p) > \text{StablishedDeliveryTime}(x, p) \rightarrow \text{NonDevelopmentalSoftwareRisk}(p)$

- *Se existe um projeto P que utiliza um software não desenvolvido pelo programa que não é customizável, então existe* ***risco de software externo*** *para P.*

- $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{NonDevelopmentalSoftware}(x, p) \wedge \text{Uses}(p, x) \wedge \neg \text{Customisable}(x) \rightarrow \text{NonDevelopmentalSoftwareRisk}(p)$

Para a Classe ***Risco de Código e Teste Unitário*** e suas subclasses foram definidos os seguinte axiomas:

Viabilidade:

- *Se existe um projeto P que possui um módulo implementável que não foi especificado durante a fase de design então existe* ***risco de viabilidade de implementação*** *para P.*
  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Module}(x, p) \wedge \neg\text{DefinedByTheDesignSpecification}(x) \rightarrow \text{ImplementationFeasibilityRisk}(p)$
- *Se existe um projeto P que possui um algoritmo cuja implementação não é fácil, então existe* ***risco de viabilidade de implementação*** *para P.*
  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Algorithm}(x, p) \wedge \neg\text{EasyImplementation}(x) \rightarrow \text{ImplementationFeasibilityRisk}(p)$

Testes Unitários:

- *Se existe um projeto P para o qual os testes unitários são iniciados antes da revisão de código com respeito ao design, então existe* ***risco de testes unitários*** *para P.*
  - $\exists\, p, x, y\,.\ \text{Project}(p) \wedge \text{Module}(x, p) \wedge \text{UnitTest}(y, x) \wedge \text{StartDate}(y) < \text{DesignVerificationDate}(x) \rightarrow \text{TestingRisk}(p)$
- *Se existe um projeto P no qual existe um módulo implementável para o qual não foram especificados testes unitários, então existe* ***risco de testes unitários*** *para P.*
  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Module}(x, \text{p}) \wedge \neg\ (\exists\, y\,.\ \text{UnitTest}(y, x)) \rightarrow \text{TestingRisk}(p)$
- *Se existe um projeto P no qual existe um módulo implementável para o qual não houve tempo suficiente para a execução dos testes unitários deste módulo, então existe* ***risco de testes unitários*** *para P.*

- $\exists\, p, x, y$. Project(p) $\wedge$ Module(x, p) $\wedge$ UnitTest(y, x) $\wedge \neg$SufficientTime(y) $\rightarrow$ TestingRisk(p)

Codificação / Implementação:

- *Se existe um projeto P no qual existe um módulo implementável cuja especificação de design não é suficiente para escrever o código, então existe* ***risco de implementação*** *para P.*
  - $\exists\, p$, x . Project(p) $\wedge$ Module(x, p) $\wedge \neg\, \exists\, y$. DesignSpecification(y, x) $\rightarrow$ ImplementationRisk(p)
- *Se existe um projeto P no qual existe um módulo implementável que está sendo codificado enquanto o design de P está sendo modificado, então existe* ***risco de implementação*** *para P.*
  - $\exists\, p, x$. Project(p) $\wedge$ Module(x, p) $\wedge$ BeingCoded(x) $\wedge$ DesignChanging(p) $\rightarrow$ ImplementationRisk(p)
- *Se existe um projeto P no qual existe uma restrição de sistema que dificulta a implementação de P, então existe* ***risco de implementação*** *para P.*
  - $\exists\, p, x$. Project(p) $\wedge$ SystemConstraint(x, p) $\wedge$ MakesCodeDifficultToWrite(x, p) $\rightarrow$ ImplementationRisk(p)
- *Se existe um projeto P a linguagem de programação utilizada para codificar não é adequada para a finalidade do(s) módulo(s) nela codificado(s), então existe* ***risco de implementação*** *para P.*
  - $\exists\, p$. Project(p) $\wedge \neg$ SuitableProgrammingLanguage(p) $\rightarrow$ ImplementationRisk(p)
- *Se existe um projeto P no qual mais de uma linguagem de programação estão sendo utilizadas para codificar e não existe compatibilidade de interface entre os códigos produzidos pelos respectivos compiladores das linguagens, então existe* ***risco de implementação*** *para P.*

- $\exists\, p, x, y$. Project(p) $\wedge$ ProgrammingLanguage(x, p) $\wedge$ ProgrammingLanguage(y, p) $\wedge\, x \neq y \wedge \neg$ InterfaceCompatible(ProducedCode(Compiler(x, p), ProducedCode(Compiler(y, p)) $\rightarrow$ ImplementationRisk(p)

- *Se existe um projeto P no qual o computador utilizado para desenvolvimento é diferente do computador alvo e existem diferenças de compiladores entre os dois computadores, então existe* ***risco de implementação*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ DevelopmentComputer(x, p) $\wedge\, x \neq$ TargetComputer(p) $\rightarrow$ ImplementationRisk(p)

- *Se existe um projeto P cujo hardware alvo está com as especificações em processo de modificação, então existe* ***risco de implementação*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ TargetHardware(x, p) $\wedge$ SpecificationIsChanging(x) $\rightarrow$ ImplementationRisk(p)

Para a Classe ***Risco de Integração e Testes*** e suas subclasses foram definidos os seguintes relacionamentos e axiomas.

Ambiente:

- *Se existe um projeto P paro o qual não há hardware suficiente para integração e testes adequados, então existe* ***risco de ambiente de integração e testes*** *para P.*

  - $\exists\, p$. Project(p) $\wedge \neg$ SufficientEnvironmentHardwareForIntegrationAndTesting(p) - $\rightarrow$ IntegrationAndTestEnvironmentRisk(p)

- *Se existe um projeto P paro o qual não é possível desenvolver um cenário realista de tráfego de dados, então existe* ***risco de ambiente de integração e testes*** *para P.*

  - $\exists\, p$. Project(p) $\wedge \neg \exists$ x . DataTrafficScenario(x, p) $\rightarrow$ IntegrationAndTestEnvironmentRisk(p)

- *Se existe um projeto P paro o qual não é possível desenvolver um cenário realista de resposta a eventos de tempo real, então existe* ***risco de ambiente de integração e testes*** *para P.*

  - $\exists\, p$. Project(p) $\wedge \neg \exists\, x$. RealTimeResponseScenario(x, p) $\rightarrow$ IntegrationAndTestEnvironmentRisk(p)

- *Se existe um projeto P paro o qual não é possível desenvolver um cenário realista de tratamento de eventos assíncronos, então existe* ***risco de ambiente de integração e testes*** *para P.*

  - $\exists\, p$. Project(p) $\wedge \neg \exists\, x$. AsynchronousEventHandlingScenario(x, p) $\rightarrow$ IntegrationAndTestEnvironmentRisk(p)

- *Se existe um projeto P paro o qual não é possível desenvolver um cenário realista de interação multi-usuário, então existe* ***risco de ambiente de integração e testes*** *para P.*

  - $\exists\, p$. Project(p) $\wedge \neg \exists\, x$. MultiUserInteractionScenario(x, p) $\rightarrow$ IntegrationAndTestEnvironmentRisk(p)

- *Se existe um projeto P paro o qual não existe software e hardware que facilite os testes de integração, então existe* ***risco de ambiente de integração e testes*** *para P.*

  - $\exists\, p$. Project(p) $\wedge \neg \exists\, x, y$. Software(x, p) $\wedge$ Hardware(y, p) $\wedge$ FacilitatesIntegrationTest(x, p) $\rightarrow$ IntegrationAndTestEnvironmentRisk(p)

  - *Se existe um projeto P paro o qual existe um software ou hardware que não é suficiente para a execução de testes de integração, então existe* ***risco de ambiente de integração e testes*** *para P.*

  - $\exists\, p$. Project(p) $\wedge \exists\, x$. ((Software(x, p) $\vee$ Hardware(x, p)) $\wedge$ FacilitatesIntegrationTest(x, p) $\wedge \neg$ Sufficient(x, p)) $\rightarrow$ IntegrationAndTestEnvironmentRisk(p)

Produto:

- *Se existe um projeto P para o qual o hardware alvo não está sempre disponível quando necessário, então existe* ***risco de integração e teste de produto*** *para P.*

  - $\exists\, p$. Project(p) $\wedge \neg$ AvailableTargetHardware(p) $\rightarrow$ ProductIntegrationAndTestRisk(p)

- *Se existe um projeto P no qual existe um requisito para o qual não foram definidos critérios formais de aceitação, então existe* ***risco de integração e teste de produto*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ Requirement(x, p) $\neg \exists\, y$. FormalAcceptanceCriteria(y, x) $\rightarrow$ ProductIntegrationAndTestRisk(p)

- *Se existe um projeto P para o qual existe uma interface externa que não é bem definida ou bem documentada, então existe* ***risco de integração e teste de produto*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ ExternalInterface(x, p) $\neg$ (Defined(x) $\wedge$ Documented(x)) $\rightarrow$ ProductIntegrationAndTestRisk(p)

- *Se existe um projeto P no qual existe um requisito para o qual não é fácil executar teste de integração de produto, então existe* ***risco de integração e teste de produto*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ Requirement(x, p) $\neg$ EasyToTest(x) $\rightarrow$ ProductIntegrationAndTestRisk(p)

- *Se existe um projeto P no qual a integração de produto não foi especificada suficientemente, então existe* ***risco de integração e teste de produto*** *para P.*

  - $\exists\, p$. Project(p) $\wedge \neg$ ($\exists\, x$. ProductIntegration(x, p) $\wedge$ Sufficient(x, p)) $\rightarrow$ ProductIntegrationAndTestRisk(p)

- *Se existe um projeto P para o qual o tempo alocado para integração e teste de produto é menor do que o tempo necessário para integração e teste de produto, então existe* ***risco de integração e teste de produto*** *para P.*

- $\exists p$, Project(p) $\wedge$ AvailableProductIntegrationAndTestTime(p) < TempoIntegracaoTesteProdutoNecessario(p) $\rightarrow$ ProductIntegrationAndTestRisk(p)

Sistema:

- *Se existe um projeto P para o qual não foi a integração de sistema não foi especificada suficientemente, então existe* ***risco de integração e teste de sistema*** *para P.*
  - $\exists p$. Project(p) $\wedge \neg$ ($\exists x$. ProductIntegrationSpecification(x, p) $\wedge$ Sufficient(x, p)) $\rightarrow$ SystemIntegrationAndTestRisk(p)
- *Se existe um projeto P no qual o tempo alocado para integração e teste de sistema é menor do que o tempo necessário para integração e teste de sistema, então existe* ***risco de integração e teste de sistema*** *para P.*
  - $\exists p$, Project(p) $\wedge$ AvailableSystemIntegrationAndTestTime(p) < NecessarySystemIntegrationAndTestTime(p) $\rightarrow$ SystemIntegrationAndTestRisk (p)
- *Se existe um projeto P n o qual existe um contratante que não faz parte do time de integração de sistema, então existe* ***risco de integração e teste de sistema*** *para P.*
  - $\exists p, x$. Project(p) $\wedge$ Contractor(x, p) $\wedge \neg$ IsPartOfIntegrationTeam(x, p) $\rightarrow$ SystemIntegrationAndTestRisk(p)
- *Se existe um projeto P para o qual a integração de sistema não será executada no local do cliente, então existe* ***risco de integração e teste de sistema*** *para P.*
  - $\exists p$. Project(p) $\wedge \neg$ IntegrationOnCustomerSite(p) $\rightarrow$ SystemIntegrationAndTestRisk(p)

Para a Classe ***Risco de Especialidades de Engenharia*** e suas subclasses foram definidos os seguintes relacionamentos e axiomas.

Manutenibilidade:

- *Se existe um projeto P paro o qual o produto será difícil de manter, então existe* ***risco de manutenibilidade*** *para P.*

- $\exists\ p,t$. Project(p) $\wedge$ Product(t, p) $\wedge$ DificultMaintain(t) $\rightarrow$ MaintainbilityRisk(p)

- *Se existe um projeto P paro o qual a fase de implementação será difícil de entender, então existe* ***risco de manutenibilidade*** *para P.*

  - $\exists\ p, h$. Project(p) $\wedge$ ImplementationPhase(h, p) $\wedge$ DificultUnderstand(h) $\rightarrow$ MaintainbilityRisk(p)

- *Se existe um projeto P paro o qual o projeto de arquitetura pode trazer dificuldades para manutenção, então existe* ***risco de manutenibilidade*** *para P.*

  - $\exists\ p,a$. Project(p) $\wedge$ ArquitectureProject(a, p) $\wedge$ DificultMaintain(a) $\rightarrow$ MaintainbilityRisk(p)

- *Se existe um projeto P paro o qual a construção do código pode trazer dificuldades para manutenção, então existe* ***risco de manutenibilidade*** *para P.*

  - $\exists\ p,c$. Project(p) $\wedge$ CreateCode(c, p) $\wedge$ DificultMaintain(c) $\rightarrow$ MaintainbilityRisk()

- *Se existe um projeto P paro o qual a modelagem pode trazer dificuldades para manutenção, então existe* ***risco de manutenibilidade*** *para P.*

  - $\exists\ p,d$. Project(p) $\wedge$ Design(d, p) $\wedge$ DificultMaintain(d) $\rightarrow$ MaintainbilityRisk(p)

Confiabilidade:

- *Se existe algum projeto P no qual requisito de confiança não foi alocado, então existe* ***risco de*** **Confiabilidade** *para P.*

  - $\exists\ p,y$. Project(p) $\wedge$ ReliabilityRequirement(y, p) $\wedge \neg$ Allocated(y) $\rightarrow$ ReliabilityRisk(p)

- *Se existe algum projeto P no qual requisito de avaliação não foi alocado, então existe* ***risco de*** **Confiabilidade** *para P.*

  - $\exists\ p,y$. Project(p) $\wedge$ EvaluationRequirement(y, p) $\wedge \neg$ Allocated(y) $\rightarrow$ ReliabilityRisk(p)

Proteção:

- *Se existe algum projeto P no qual requisito de proteção não foi alocado, então existe* ***risco de proteção*** *para P.*
  - $\exists\, p,y$. Project(p) $\wedge$ SafetyRequirement(y, p) $\wedge \neg$ Allocated(y) $\rightarrow$ SafetyRisk(p)
- *Se existe algum projeto P que possui requisito e este requisito é inviável , então existe* ***risco de proteção*** *para P.*
  - $\exists\, p,y$. Project(p) $\wedge$ Requirement(y, p) $\wedge \neg$ Feasible(y) $\rightarrow$ SafetyRisk(p)
- *Se existe algum projeto P que possui requisito e este requisito não é demonstrável, então existe* ***risco de proteção*** *para P.*
  - $\exists\, p,y$. Project(p) $\wedge$ Requirement(y, p) $\wedge \neg$ Demonstrable(y) $\rightarrow$ SafetyRisk(p)
- *Se existe algum projeto P que possui requisito e este requisito é de proteção e é difícil verificar a satisfação deste requisito, então existe* ***risco de proteção*** *para P.*
  - $\exists\, p,y$. Project(p) $\wedge$ SafetyRequirement(y, p) $\wedge$ HardEvaluateSatisfaction(y) $\rightarrow$ SafetyRisk(p)

Segurança:

- *Se existe algum projeto P que possui requisito e este requisito tem um nível de segurança nunca realizado antes, então existe* ***risco de segurança*** *para P.*
  - $\exists\, p,y$. Project(p) $\wedge$ Requirement(y, p) $\wedge$ SecutyLevelNeverDone(y) $\rightarrow$ SecurityRisk(p)
- *Se existe algum projeto P que possui requisito e este requisito não é um requisito precedente, então existe* ***risco de segurança*** *para P.*
  - $\exists\, p,y$. Project(p) $\wedge$ Requirement(y, p) $\wedge \neg$ PrecedentRequirement(y) $\rightarrow$ SecurityRisk(p)

Fatores Humanos:

- *Se existe algum projeto P que possui uma interface e esta interface está pobremente definida, então existe* ***risco de fatores humanos*** *para P.*

  - $\exists\, p,x\ .\ \text{Project}(p) \wedge \text{Interface}(x, p) \wedge \neg\, \text{Defined}(x) \rightarrow \text{HumanFactorsRisk}(p)$

- *Se existe algum projeto P que possui requisito e este requisito não satisfaz os requisitos de Fatores Humanos, então existe* ***risco de fatores humanos*** *para P.*

  - $\exists\, p,y\ .\ \text{Project}(p) \wedge \text{Requirement}(y, p) \wedge \neg\, \text{Satisfaction}(y) \rightarrow \text{HumanFactorsRisk(p)}$

Especificações:

- *Se existe algum projeto P no qual a documentação de design não é adequada, então existe* ***risco de especificações*** *para P.*

  - $\exists\, p, u\ .\ \text{Project}(p) \wedge \text{DesignDocumentation}(u, p) \wedge \neg\, \text{Adequate}(u) \rightarrow \text{SpecificationRisk}(p)$

- *Se existe algum projeto P no qual a documentação de implementação não é adequada, então existe* ***risco de especificações*** *para P.*

  - $\exists\, p, u\ .\ \text{Project}(p) \wedge \text{ImplementationDocumentation}(u, p) \wedge \neg\, \text{Adequate}(u) \rightarrow \text{SpecificationRisk}(p)$

- *Se existe algum projeto P no qual a documentação de teste do sistema não é adequada, então existe* ***risco de especificações*** *para P.*

  - $\exists\, p, u\ .\ \text{Project}(p) \wedge \text{TestDocumentation}(u, p) \wedge \neg\, \text{Adequate}(u) \rightarrow \text{SpecificationRisk}(p)$

- *Se existe algum projeto P no qual a documentação de especificação de requisitos não adequada para o design , então existe* ***risco de especificações*** *para P.*

  - $\exists\, p, u\ .\ \text{Project}(p) \wedge \text{SpecificationDocumentation}(u, p) \wedge \neg\, \text{Adequate}(u) \rightarrow \text{SpecificationRisk}(p)$

- *Se existe algum projeto P no qual a especificação de hardware não é adequada para o design , então existe* ***risco de especificações*** *para P.*

  - $\exists\, p, s$. Project(p) $\wedge$ HardwareSpecification(s, p) $\wedge \neg$ AdequateDesign(s) $\rightarrow$ SpecificationRisk(p)

- *Se existe algum projeto P no qual a especificação de hardware não é adequada para a implementação, então existe* ***risco de especificações*** *para P.*

  - $\exists\, p, s$. Project(p) $\wedge$ HardwareSpecification(s, p) $\wedge \neg$ AdequateImplementation(s) $\rightarrow$ SpecificationRisk(p)

- *Se existe algum projeto P que possui uma interface externa e esta interface não está bem especificada, então existe* ***risco de especificações*** *para P.*

  - $\exists\, p, r$. Project(p) $\wedge$ ExternalInterface(r, p) $\wedge \neg$ WellSpecified(r) $\rightarrow$ SpecificationRisk(p)

- *Se existe algum projeto P no qual a especificação de testes não é adequada para testar todo o sistema, então existe* ***risco de especificações*** *para P*

  - $\exists\, p, f$. Project(p) $\wedge$ TestSpecification(f, p) $\wedge \neg$ Adequate(f) $\rightarrow$ SpecificationRisk(p)

3.2.1.1. Relacionamentos da Classe Engenharia de Produto:

- ***Risco de estabilidade de requisitos*** *acarreta* ***risco de escala de requisitos.***

  - $\exists\, p$. Project(p) $\wedge$ RequirementsStabilityRisk(p) $\rightarrow$ ScaleRisk(p)

- ***Risco de estabilidade de requisitos*** *acarreta* ***risco de completude de requisitos.***

  - $\exists\, p$. Project(p) $\wedge$ RequirementsStabilityRisk(p) $\rightarrow$ RequirementsCompletenessRisk(p)

- ***Risco de estabilidade de requisitos*** *acarreta* ***risco de validade de requisitos.***

- $\exists\, p$. Project(p) $\wedge$ RequirementsStabilityRisk(p) $\rightarrow$ RequirementsValidityRisk(p)

- ***Risco de viabilidade de requisitos*** *acarreta* ***risco de requisitos precedentes.***

  - $\exists\, p$. Project(p) $\wedge$ RequirementsFeasibilityRisk(p) $\rightarrow$ PrecedentRisk(p)

- ***Risco de estabilidade de requisitos*** *acarreta* ***risco de claridade de requisitos.***

  - $\exists\, p$. Project(p) $\wedge$ RequirementsCompletenessRisk(p) $\rightarrow$ ImplementationRisk(p)

- ***Risco de completude de requisitos*** *acarreta* ***risco de validade de requisitos.***

  - $\exists\, p$. Project(p) $\wedge$ RequirementsCompletenessRisk(p) $\rightarrow$ RequirementsValidityRisk(p)

- ***Risco de clareza de requisitos*** *acarreta* ***risco de validade de requisitos.***

  - $\exists\, p$. Project(p) $\wedge$ RequirementsClarityRisk(p) $\rightarrow$ RequirementsValidityRisk(p)

- ***Risco de completude de requisitos*** *acarreta* ***risco de implementação.***

  - $\exists\, p$. Project(p) $\wedge$ RequirementsCompletenessRisk(p) $\rightarrow$ ImplementationRisk(p)

- ***Risco de interface*** *acarreta* ***risco de integração e teste de produto.***

  - $\exists\, p$. Project(p) $\wedge$ InterfaceRisk(p) $\rightarrow$ ProductIntegrationAndTestRisk(p)

- ***Risco de limitações de hardware*** *acarreta* ***risco de implementação.***

  - $\exists\, p$. Project(p) $\wedge$ HardwareConstraintsRisk(p) $\rightarrow$ ImplementationRisk(p)

- ***Risco de limitações de hardware*** *acarreta* ***risco de performance.***

  - $\exists\, p$. Project(p) $\wedge$ HardwareConstraintsRisk(p) $\rightarrow$ PerformanceRisk(p)

- ***Risco de software externo*** *acarreta* ***risco de performance.***

  - $\exists\, p$. Project(p) $\wedge$ NonDevelopmentalSoftwreRisk(p) $\rightarrow$ PerformanceRisk(p)

- ***Risco de testabilidade*** *acarreta* ***risco de integração e teste de produto.***

- $\exists\, p$. Project(p) $\wedge$ TestabilityRisk(p) $\rightarrow$ ProductIntegrationAndTestRisk(p)

- ***Risco de especificações*** *causa* ***risco de clareza dos requisitos.***
  - $\exists\, p$. Project(p) $\wedge$ SpecificationRisk(p) $\rightarrow$ RequirementsClarityRisk(p)
- ***Risco de especificações*** *causa* ***risco de completude dos requisitos.***
  - $\exists\, p$. Project(p) $\wedge$ SpecificationRisk(p) $\rightarrow$ RequirementsCompletenessRisk(p)
- ***Risco de especificações*** *causa* ***risco de design.***
  - $\exists\, p$. Project(p) $\wedge$ HardwareConstraintsRisk(p) $\rightarrow$ ReliabilityRisk(p)
- ***Risco de especificações*** *causa* ***risco de implementação.***
  - $\exists\, p$. Project(p) $\wedge$ ImplementationRisk(p) $\rightarrow$ MaintainabilityRisk(p)
- ***Risco de escala*** acarreta ***risco de confiabilidade***
  - $\exists\, p$. Project(p) $\wedge$ ScaleRisk(p) $\rightarrow$ ReliabilityRisk(p)
- ***Risco de estabilidade de requisitos*** *acarreta* ***risco de integração e testes.***
  - $\exists\, p$. Project(p) $\wedge$ RequirementsStabilityRisk (p) $\rightarrow$ IntegrationAndTestRisk(p)
- ***Risco de estabilidade de requisitos*** *acarreta* ***risco de design.***
  - $\exists\, p$. Project(p) $\wedge$ RequirementsStabilityRisk(p) $\rightarrow$ DesignRisk(p)
- ***Risco de estabilidade de requisitos*** *acarreta* ***risco de funcionalidade.***
  - $\exists\, p$. Project(p) $\wedge$ RequirementsStabilityRisk(p) $\rightarrow$ FunctionalityRisk(p)
- ***Risco de implementação*** acarreta ***risco de manutenibilidade.***
  - $\exists\, p$. Project(p) $\wedge$ ImplementationRisk(p) $\rightarrow$ MaintainabilityRisk(p)
- ***Risco de limitações de hardware*** acarreta ***risco de confiabilidade.***

- ∃ *p* . Project(*p*) ∧ HardwareConstraintRisk(*p*) → ReliabilityRisk(*p*)

- ***Risco de requisitos precedentes*** *causa* ***risco de segurança.***

  - ∃ *p* . Project(*p*) ∧ PrecedentRisk(*p*) → SecurityRisk(*p*)

- ***Risco de requisitos precedentes*** *acarreta* ***risco de viabilidade de requisitos.***

  - ∃ *p* . Project(*p*) ∧ PrecedentRisk(*p*) → RequirementsFeasibilityRisk(*p*)

- ***Risco de software externo*** *acarreta* ***risco de requisitos.***

  - ∃ *p* . Project(*p*) ∧ NonDevelopmentalSoftwareRisk(*p*) → RequirementsRisk(*p*)

- ***Risco de viabilidade de requisitos*** *acarreta* ***risco de integração e testes.***

  - ∃ *p* . Project(*p*) ∧ RequirementsFeasibilityRisk (*p*) → IntegrationAndTestRisk(*p*)

3.2.2. Ambiente de Desenvolvimento

Para a Classe ***Risco de Processo de Desenvolvimento*** e suas subclasses foram definidos os seguintes axiomas:

Formalidade:

- *Se existe algum projeto P no qual um processo não é bem definido, então existe um* ***risco de formalidade de processo de desenvolvimento*** *para P.*

  - ∃ *p, x* . Project(*p*) ∧ Process(x, *p*) ∧ ¬ WellDefined(*x*) → DevelopmentProcessFormalityRisk(*p*)

- *Se existe algum projeto P no qual um processo não é bem documentado, então existe um* ***risco de formalidade de processo de desenvolvimento*** *para P.*

  - ∃ *p, x* . Project(*p*) ∧ Process(*x, p*) ∧ ¬ WellDocumented(*x*) → DevelopmentProcessFormalityRisk (*p*)

- *Se existe algum 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*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Process}(x, p) \wedge \neg\ \text{CommunicatedForAllAspectsAndPhases}(x) \rightarrow \text{DevelopmentProcessFormalityRisk}\ (p)$

Adequabilidade:

- *Se existe algum 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*** *para P.*

  - $\exists\, p, x, y, z, w\,.\ \text{Project}(p) \wedge \text{DevelopmentModel}(x, p) \wedge \text{Process}(y, p) \wedge \text{Method}(z, p) \wedge \text{RequiredActivities}(w, p) \wedge \neg\ \text{Supportive}(x) \wedge \neg\ \text{Supportive}(y) \wedge \neg\ \text{Supportive}(z) \wedge \neg\ \text{Supportive}(w) \rightarrow \text{DevelopmentProcessSuitabilityRisk}(p)$

Controle do Processo:

- *Se existe algum projeto P no qual não há garantia dos processos definidos, então existe um* ***risco de controle do processo de desenvolvimento*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{DefinedProcess}(x, p) \wedge \neg\ \text{Guarantee}(x) \rightarrow \text{DevelopmentProcessControlRisk}(p)$

- *Se existe algum projeto P no qual existe um processo que não é dimensionado e aperfeiçoado, então existe um* ***risco de controle do processo de desenvolvimento*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Process}(x, p)\ \wedge \neg\ \text{Improvement}(x) \wedge \neg\ \text{Measurement}(x) \rightarrow \text{DevelopmentProcessControlRisk}(p)$

Familiaridade:

- *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.*

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

Controle do Produto:

- *Se existe algum 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*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{Requirements}(x, p) \wedge \neg\, \text{Trace}(x) \rightarrow \text{ProductControlRisk}(p)$

Para a Classe ***Risco de Sistema de Desenvolvimento*** e suas subclasses foram definidos os seguintes axiomas:

Capacidade:

- *Se existe algum projeto P que possui poucas estações de desenvolvimento, então existe um* ***risco de capacidade de sistema de desenvolvimento*** *para P.*

  - $\exists\, p\,.\ \text{Project}(p) \wedge \text{FewWorkstations}(p) \rightarrow \text{DevelopmentSystemCapacityRisk}(p)$

- *Se existe algum projeto P que possui máquinas com pouco poder de processamento, então existe um* ***risco de capacidade de sistema de desenvolvimento*** *para P.*

  - $\exists\, p,x\,.\ \text{Project}(p) \wedge \text{DevelopmentMachine}(x, p) \wedge \text{InsufficientProcessingPower}(x) \rightarrow \text{DevelopmentSystemCapacityRisk}\ (p)$

- *Se existe algum projeto P no qual o banco de dados utilizado possui armazenamento insuficiente, então existe um* ***risco de capacidade de sistema de desenvolvimento*** *para P.*

  - $\exists\, p,x\,.\ \text{Project}(p) \wedge \text{Database}(x, p) \wedge \text{InsufficientStorage}(x) \rightarrow \text{DevelopmentSystemCapacityRisk}\ (p)$

- *Se existe algum 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** para P.*

  - $\exists\, p,x\,.\ \text{Project}(p) \wedge \text{EquipmentToSupportParallelActivities}(x, p) \wedge \text{Inadequate}(x) \rightarrow \text{DevelopmentSystemCapacityRisk}\ (p)$

Adequabilidade:

- *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.*

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

Usabilidade:

- *Se existe algum projeto P cujo sistema de desenvolvimento não possui documentação ou não é acessível, então existe um **risco de usabilidade de sistema de desenvolvimento** para P.*

  - $\exists\, p\,.\ \text{Project}(p) \wedge \neg\, (\exists\, x\,.\ \text{DevelopmentSystemDocumentation}(x, p) \wedge \text{Accessible}(x, p)) \rightarrow \text{DevelopmentSystemUsabilityRisk}(p)$

- *Se existe algum projeto P cujo sistema de desenvolvimento não é fácil de usar, então existe um **risco de usabilidade de sistema de desenvolvimento** para P.*

  - $\exists\, p, x.\ \text{Project}(p) \wedge \text{DevelopmentSystem}(x, p) \wedge \neg\, \text{EasyToUse}(x) \rightarrow \text{DevelopmentSystemUsabilityRisk}(p)$

Familiaridade:

- *Se existe algum 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*** *para P.*

  - $\exists\, p,x.\ \text{Project}(p) \wedge \text{DevelopmentSystem}(x, p) \wedge (\neg\ \text{PriorUseByCompany}(x)\ \vee \neg\ \text{PriorUseByProjectPersonnel}(x)) \rightarrow \text{DevelopmentSystemFamiliarityRisk}(p)$

- *Se existe algum 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*** *para P.*

  - $\exists\, p.\ \text{Project}(p) \wedge \neg\ (\exists\, x\ .\ \text{DevelopmentSystemTraining}(x, p) \wedge \text{AdequateForNewUsers}(x)) \rightarrow \text{DevelopmentSystemFamiliarityRisk}(p)$

Confiabilidade:

- *Se existe algum projeto P onde os componentes do sistema de desenvolvimento não estejam disponíveis, então existe um* ***risco de confiabilidade de sistema de desenvolvimento*** *para P.*

  - $\exists\, p, x\ .\ \text{Project}(p) \wedge \text{DevelopmentSystemComponents}(x, p)\ \wedge \neg\ \text{Available}(x)\ \rightarrow \text{DevelopmentSystemReliabilityRisk}(p)$

- *Se existe algum projeto P onde os componentes do sistema de desenvolvimento não estejam funcionando corretamente, então existe um* ***risco de confiabilidade de sistema de desenvolvimento*** *para P.*

  - $\exists\, p, x\ .\ \text{Project}(p) \wedge \text{DevelopmentSystemComponents}\ (x, p)\ \wedge \neg\ \text{WorkingProperly}(x)\ \rightarrow \text{DevelopmentSystemReliabilityRisk}(p)$

Suporte do Sistema:

- *Se existe algum projeto P que não envolveu um treinamento na utilização do sistema de desenvolvimento, então existe um* ***risco de suporte do sistema de desenvolvimento*** *para P.*

- $\exists p, x$. Project(p) $\wedge$ DevelopmentSystem(x, p) $\wedge \neg$ InvolveTraining(p) $\rightarrow$ DevelopmentSystemSupportRisk(p)

- *Se existe algum projeto P com algum sistema de desenvolvimento que não oferece acesso a usuários experientes, então existe um* ***risco de suporte do sistema de desenvolvimento*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ DevelopmentSystem(x, p) $\wedge \neg$ AccessToExpertUsers(x) $\rightarrow$ DevelopmentSystemSupportRisk (p)

- *Se existe algum 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*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ DevelopmentSystem(x, p) $\wedge \neg$ ProblemResolution(x) $\rightarrow$ DevelopmentSystemSupportRisk (p)

Entrega:

- *Se existe algum projeto P no qual não são alocados os recursos necessários para satisfazer os requisitos de entrega, então existe* ***um risco de entrega*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ AllocatedResources(x, p) $\wedge \neg$ EnsureMeetDeliverableRequirements(x) $\rightarrow$ DeliverabilityRisk(p)

Para a Classe ***Risco de Processo de Gerenciamento*** e suas subclasses foram definidos os seguintes axiomas:

Planejamento:

- *Se existe 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****.*

  - $\exists p, x$. Project(p) $\wedge$ Plan(x) $\wedge \neg$ Managed(p, x) $\rightarrow$ PlanningRisk(p)

- *Se existe algum projeto P para o qual ocorre alguma interrupção e não existe o replanejamento, então existe* ***risco de planejamento*** .

- $\exists\, p, x, y.\ \text{Project}(p) \wedge \text{Plan}(x) \wedge \text{Interruption}(y) \wedge \neg\, \text{Replan}(p, x) \rightarrow \text{PlanningRisk}(p)$

- *Se existe algum projeto P no qual as pessoas em todos os níveis não são incluídas no planejamento de suas atividades, então existe* ***risco de planejamento****.*

  - $\exists\, p, x, y, z.\ \text{Project}(p) \wedge\ \text{Manager}(z) \wedge \text{Person}(y) \wedge \text{Resource}(y, p) \wedge \neg\, \text{PlanActivities}(y, z, p) \rightarrow \text{PlanningRisk}(p)$

- *Se existe algum projeto P no qual não existem planos de contingência para todos os riscos conhecidos, então existe* ***risco de planejamento****.*

  - $\exists\, p, x, y.\ \text{Project}(p) \wedge\ \text{Risk}(y) \wedge \neg\, \text{ContingencyPlan}(y, p) \rightarrow \text{PlanningRisk}(p)$

- *Se existe algum projeto P para o qual as tarefas de longo prazo não estão adequadamente sendo tratadas, então existe* ***risco de planejamento****.*

  - $\exists\, p, x, y.\ \text{Project}(p) \wedge\ \text{LongTermIssue}(y) \wedge \neg\, \text{Adressed}(y, p) \rightarrow \text{PlanningRisk}(p)$

Organização do Projeto:

- *Se existe algum projeto P no qual não existe uma organização do projeto, então existe* ***risco de organização do projeto****.*

  - $\exists\, p, x\ .\ \text{Project}(p) \wedge \neg\, \text{OrganizationProgram}(x, p) \rightarrow \text{ProjectOrganizationRisk}(p)$

- *Se existe algum projeto P no qual as pessoas não entendem seus papeis ou os papeis dos demais no projeto, então existe* ***risco de organização do projeto****.*

  - $\exists\, p, x\ .\ \text{Project}(p) \wedge \text{Person}(x)\ \wedge \text{Roles}(x, p)\ \wedge \neg\, (\ \text{UnderstandRole}(x, p)\ \text{OR}\ \text{UnderstandOthersRole}(x, p)) \rightarrow \text{ProjectOrganizationRisk}(p)$

- *Se existe algum projeto P no qual as pessoas não sabem quem tem autoridade para o que, então existe* ***risco de organização do projeto****.*

  - $\exists\, p, x\ .\ \text{Project}(p) \wedge \text{Person}(x) \wedge \text{Task}(y) \wedge \text{Authority}(x, y)\ \wedge \exists\, x, \neg\, \text{KnowAuthority}(x, y, p) \rightarrow \text{ProjectOrganizationRisk}(p)$

Experiência de Gerenciamento:

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

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

Interfaces de comunicação do programa (*design*):

- *Se existe um 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**.*

  - $\exists\, p, x, y\,.\ \text{Project}(p) \wedge \text{HierachyLineUpDown}(x) \wedge \text{Problems}(y, p) \wedge \neg\ \text{Comunicate}(y, x) \rightarrow \text{InterfaceProgramRisk}(p)$

- *Se existe um projeto P no qual existem conflitos não documentados e resolvidos em tempo apropriados, então existe **risco de interface de comunicação**.*

  - $\exists\, p, x, y\,.\ \text{Project}(p) \wedge \neg\ \text{ConflictsDocumentedResolved}(x)\ \rightarrow\ \text{InterfaceProgramRisk}(p)$

- *Se existe um projeto P no qual os membros apropriados não são envolvidos em reuniões com os clientes, então existe **risco de interface de comunicação**.*

  - $\exists\, p, x, y\,.\ \text{Project}(p) \wedge \text{ApropriateMember}(x) \wedge \neg\ \text{MeetingCustomer}(x, p) \rightarrow\ \text{InterfaceProgramRisk}(p)$

- *Se existe um 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**.*

  - $\exists\, p, x, y\,.\ \text{Project}(p) \wedge \neg\ \text{AllFactionCustomerRepresentant}(x) \rightarrow\ \text{InterfaceProgramRisk}(p)$

Para a Classe ***Risco de Métodos de Gerenciamento*** e suas subclasses foram definidos os seguintes axiomas:

Monitoramento:

- *Se existe um projeto P no qual não existem relatórios de acompanhamento estruturados periódicos, então existe* ***risco de monitoramento.***
  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \neg\, \text{FrequentStatusReport}(x) \rightarrow \text{MonitoringRisk}(p)$
- *Se existe um projeto P no qual as informações apropriadas não são relatadas aos níveis organizacionais corretos, então existe* ***risco de monitoramento****.*
  - $\exists\, p, x\, .\, \text{Project}(p) \wedge \neg\, \text{RightInformationLevelStatusReport}(x) \rightarrow \text{MonitoringRisk}(p)$
- *Se existe um projeto P no qual o progresso não é comparado com o planejado, então existe o* ***risco de monitoramento****.*
  - $\exists\, p, x, y\, .\, \text{Project}(p) \wedge \text{Plan(x)} \wedge \text{Progress(y)} \wedge \neg\, \text{Compared}(x,y,p) \rightarrow \text{MonitoringRisk}(p)$

Gerenciamento de Pessoal:

- *Se existe um projeto P no qual as pessoas não foram treinadas nas qualificações necessárias para este programa, então existe* ***risco de gerenciamento de pessoal****.*
  - $\exists\, p, x, y\, .\, \text{Project}(p) \wedge \text{Skills}(x) \wedge \text{RequiredSkills}(x, p) \wedge \neg\, \text{PeopleTrained}(x,y,p) \rightarrow \text{PersonnelManagementRisk}(p)$
- *Se existe um projeto P no qual as pessoas envolvidas não possuem experiências em suas áreas, então existe* ***risco de gerenciamento de pessoal****.*
  - $\exists\, p, x, y\, .\, \text{Project}(p) \wedge \text{Skills}(x) \wedge \neg\, \text{PeopleHave}(x, p) \rightarrow \text{PersonnelManagementRisk}(p)$
- *Se existe um projeto P no qual não é fácil para os membros do projeto serem gerenciados, então existe* ***risco de gerenciamento de pessoal****.*

- ∃ *p, x, y, z* . Project(*p*) ∧ People(*y*) ∧ Managed(*y, p*) ∧ ¬ Easy(z) PeopleHave(*x, p*) → PersonnelManagement(*p*)

- *Se existe um 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****.*

  - ∃ *p, x* . Project(*p*) ∧ ¬ PeopleKnowRealStatusVersusPlanned(*x*) → PersonnelManagementRisk(*p*)

- *Se existe um projeto P no qual os membros do projeto não sentem que é importante seguir o planejamento, então existe* ***risco de gerenciamento de pessoal****.*

  - ∃ *p, x*. Project(*p*) ∧ ¬ FeelImportantFollowPlan(*x*) → PersonnelManagementRisk(*p*)

- *Se existe um projeto P no qual os membros do projeto não são consultados sobre as decisões gerenciais que afetam seu trabalho, então existe r****isco de gerenciamento de pessoal****.*

  - ∃ *p, x, y, z*. Project(*p*) ∧ People(*x*) ∧ Decisions(*y*) ∧ AffectWork(*z*) ¬ Consulted(*x, y, z, p*) → PersonnelManagementRisk(*p*)

- *Se existe um projeto P no qual as pessoas apropriadas não são envolvidas nas reuniões com o cliente, então existe* ***risco de gerenciamento de pessoal****.*

  - ∃ *p, x, y*. Project(*p*) ∧ RightPeople(*x*) ∧ MeetingClient(*y*) ∧ ¬ Involved(*x, y, p*)→ PersonnelManagementRisk(*p*)

Controle de Qualidade:

- *Se existe um 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****.*

  - ∃ *p, x, y*. Project(*p*) ∧ QualityAssuranceFuncions(*x*) ∧ People(*y*) ∧ ¬ QualityAssuranceSkills(*y, x, p*)→ QualityAssuranceRisk(*p*)

- *Se existe um projeto P no qual não possui mecanismos definidos para garantir qualidade então existe* ***risco de controle de qualidade****.*

  - $\exists\, p, x.\ \text{Project}(p) \wedge \neg\, \text{DefinedQualityAssuranceMechanisms}(x) \rightarrow \text{QualityAssuranceRisk}(p)$

Gerência de Configuração:

- *Se existe um projeto P no qual não existe um sistema de gerenciamento de configuração adequado, então existe* ***risco de gerência de configuração****.*

  - $\exists\, p, x.\ \text{Project}(p) \wedge \neg\, \text{ConfigurationManagementSystem}(x) \rightarrow \text{ConfigurationManagementRisk}(p)$

- *Se existe um projeto P no qual as funções de gerenciamento de configuração não estão adequadamente alocadas a membros qualificados, então existe* ***risco de gerência de configuração****.*

  - $\exists\, p, x, y.\ \text{Project}(p) \wedge\ \text{People}(x) \wedge \text{ConfigurationManagementFuncions}(y) \wedge \neg\, \text{Alocated}(x, y, p) \rightarrow \text{ConfigurationManagementRisk}(p)$

- *Se existe um 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****.*

  - $\exists\, p, x, y.\ \text{Project}(p) \wedge\ \text{InstalledSystem}(x) \wedge \text{Coordination}(x, p) \wedge \neg\, \text{ConfigurationManagement}(x) \rightarrow \text{ConfigurationManagementRisk}(p)$

- *Se existe um 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****.*

  - $\exists\, p, x, y.\ \text{Project}(p) \wedge \text{ConfigurationManagement}(x) \wedge \neg\, \text{SupportMultipleSites}(x) \rightarrow \text{ConfigurationManagementRisk}(p)$

Para a Classe ***Risco de Ambiente de Trabalho*** e suas subclasses foram definidos os seguintes relacionamentos e axiomas.

Atitude para Qualidade:

- *Se existe um projeto P no qual os múltiplos níveis dos membros não estão orientados a procedimento de qualidade, então existe* ***risco de atitude para qualidade****.*
  - $\exists\, p, x.\ \text{Project}(p) \wedge \text{People}(x) \wedge \neg\, \text{QualityOriented}(x) \rightarrow \text{QualityAtitudeRisk}(p)$
- *Se existe um projeto P no qual o cronograma não considera a qualidade, então existe* ***risco de atitude para qualidade****.*
  - $\exists\, p, x.\ \text{Project}(p) \wedge \text{Schedule}(x) \wedge \neg\, \text{QualityOriented}(x) \rightarrow \text{QualityAtitudeRisk}(p)$

Cooperação:

- *Se existe um projeto P no qual as pessoas não trabalham cooperativamente além de seus limites funcionais, então existe* ***risco de cooperação****.*
  - $\exists\, p, x.\ \text{Project}(p) \wedge \text{People}(x) \wedge \neg\, \text{WorkCooperatively}(x, p) \rightarrow \text{CooperationRisk}(p)$
- *Se existe um projeto P no qual as pessoas não trabalham efetivamente em prol dos objetivos em comum, então existe* ***risco de cooperação****.*
  - $\exists\, p, x.\ \text{Project}(p) \wedge \text{People}(x) \wedge \neg\, \text{EffectiveWorkInCommomObjectives}(x, p) \rightarrow \text{CooperationRisk}(p)$
- *Se existe um projeto P no qual são necessárias intervenções gerenciais para que as pessoas possam trabalham conjuntamente, então existe* ***risco de cooperação****.*
  - $\exists\, p, x, y.\ \text{Project}(p) \wedge \text{CooperativeWork}(y) \wedge \text{ManagementIntervention}(y) \rightarrow \text{CooperationRisk}(p)$

Comunicação:

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

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

- *Se existe um projeto P no qual os gerentes não estão receptivos a comunicação dos membros do projeto, então existe* ***risco de comunicação****.*

  - $\exists p, x.\ \text{Project}(p) \wedge \text{Manager}(x) \wedge \neg\ \text{ReceptiveCommunication}(x, p) \rightarrow \text{CommunicationRisk}(p)$

- *Se existe um projeto P no qual os membros não recebem notificações acerca de suas atividades em tempo adequado, então existe* ***risco de comunicação****.*

  - $\exists p, x, y.\ \text{Project}(p) \wedge \text{People}(x) \wedge \text{Activities}(y) \wedge\neg\ \text{TimelyNotification}(x, p) \rightarrow \text{CommunicationRisk}(p)$

Moral:

- *Se existe um projeto P no qual a moral está baixa, então existe* ***risco de moral****.*

  - $\exists p, x\ .\ \text{Project}(p) \wedge \text{LowMorale}(x) \rightarrow \text{MoraleRisk}(p)$

- *Se existe um projeto P no qual os membros estão sendo afetados por problemas, então existe* ***risco de moral****.*

  - $\exists p, x, y\ .\ \text{Project}(p) \wedge \text{People}(x) \wedge \text{Problem}(y) \wedge \text{Affected}(y, x, p) \rightarrow \text{MoraleRisk}(p)$

3.2.2.1. Relacionamentos da Classe Ambiente de Desenvolvimento

- ***Risco de usabilidade*** *acarreta* ***um risco de familiaridade do sistema de desenvolvimento.***

  - $\exists p.\ \text{Project}(p) \wedge \text{UsabilityRisk}(p) \rightarrow\ \text{DevelopmentSystemFamiliarityRisk}(p)$

- ***Risco de familiaridade do sistema de desenvolvimento*** *acarreta* ***um risco de suporte do sistema.***

- $\exists p.\ \text{Project}(p) \wedge \text{DevelopmentSystemFamiliarityRisk}(p) \rightarrow \text{SystemSupportRisk}(p)$

- ***Risco de capacidade*** *acarreta* ***um risco de confiabilidade.***

  - $\exists p.\ \text{Project}(p) \wedge \text{CapacityRisk}(p) \rightarrow \text{ReliabilityRisk}(p)$

- ***Risco de familiaridade do processo de desenvolvimento*** *acarreta* ***um risco de adequabilidade do processo de desenvolvimento.***

  - $\exists p.\ \text{Project}(p) \wedge \text{DevelopmentProcessFamiliarityRisk}(p) \rightarrow \text{DevelopmentProcessSuitabilityRisk}(p)$

- ***Risco de familiaridade do processo de desenvolvimento*** *acarreta* ***um risco de adequabilidade do sistema de desenvolvimento***

  - $\exists p.\ \text{Project}(p) \wedge \text{DevelopmentProcessFamiliarityRisk}(p) \rightarrow \text{DevelopmentSystemSuitabilityRisk}(p)$

- ***Risco de controle do processo*** *acarreta* ***um risco de controle do produto***

  - $\exists p.\ \text{Project}(p) \wedge \text{ProcessControlRisk}(p) \rightarrow \text{ProductControlRisk}(p)$

- ***Risco de entrega*** *acarreta* ***um risco de controle de qualidade***

  - $\exists p.\ \text{Project}(p) \wedge \text{DeliverabilityRisk}(p) \rightarrow \text{QualityAssuranceRisk}(p)$

- ***Risco de formalidade*** *acarreta* ***um risco de gerenciamento de configuração***

  - $\exists p.\ \text{Project}(p) \wedge \text{FormalityRisk}(p) \rightarrow \text{ConfigurationManagementRisk}(p)$

- ***Risco de familiaridade do processo de desenvolvimento*** *acarreta* ***um risco atitude de qualidade***

  - $\exists p.\ \text{Project}(p) \wedge \text{DevelopmentProcessFamiliarity}(p) \rightarrow \text{QualityAttitudeRisk}(p)$

- ***Risco de monitoramento*** *acarreta em* ***risco de gerenciamento de configuração***

- $\exists p$. Project(p) $\wedge$ MonitoringRisk(p) $\rightarrow$ ConfigurationManagementRisk(p)

- ***Risco de gerenciamento de pessoal*** *acarreta em* ***risco de gerenciamento de configuração***
  - $\exists p$. Project(p) $\wedge$ PersonnelManagementRisk(p) $\rightarrow$ ConfigurationManagementRisk(p)

- ***Risco de gerenciamento de configuração*** *acarreta em* ***risco de comunicação***
  - $\exists p$. Project(p) $\wedge$ ConfigurationManagementRisk(p) $\rightarrow$ CommunicationRisk(p)

- ***Risco de controle da qualidade*** *acarreta em* ***risco de atitude de qualidade***
  - $\exists p$. Project(p) $\wedge$ QualityAssuranceRisk(p) $\rightarrow$ QualityAttitudeRisk(p)

- ***Risco de planejamento*** *acarreta em* ***risco de atitude de qualidade***
  - $\exists p$. Project(p) $\wedge$ PlanningRisk(p) $\rightarrow$ QualityAttitudeRisk(p)

- ***Risco de gerenciamento de pessoal*** *acarreta em* ***risco de cooperação***
  - $\exists p$. Project(p) $\wedge$ PersonnelManagementRisk(p) $\rightarrow$ CooperationRisk(p)

- ***Risco de gerenciamento de pessoal*** *acarreta em* ***risco de comunicação***
  - $\exists p$. Project(p) $\wedge$ PersonnelManagementRisk(p) $\rightarrow$ CommunicationRisk(p)

- ***Risco de organização do projeto*** *acarreta em* ***risco de comunicação***
  - $\exists p$. Project(p) $\wedge$ ProjectOrganizationRisk(p) $\rightarrow$ CommunicationRisk(p)

- ***Risco de organização do projeto*** *acarreta em* ***risco de moral***
  - $\exists p$. Project(p) $\wedge$ ProjectOrganizationRisk(p) $\rightarrow$ MoraleRisk(p)

- ***Risco de gerenciamento de pessoal*** *acarreta em* ***risco de moral***
  - $\exists p$. Project(p) $\wedge$ PersonnelManagementRisk(p) $\rightarrow$ MoraleRisk(p)

3.2.3. Restrições do Programa

Para a Classe ***Risco de Recursos*** e suas subclasses foram definidos os seguintes axiomas:

Cronograma:

- *Se existe algum projeto P no qual existe um cronograma com eventos internos e externos, cujos prazos previstos de realização dos mesmos não são realistas, então existe **risco de cronograma** para P.*
  - $\exists\, p$. Project(p) $\wedge$ Schedule(x,p) $\wedge \neg$ Realistic(x) $\rightarrow$ ScheduleRisk(p)
- *Se existe algum projeto P no qual existe um cronograma com eventos internos e externos, cujo método para estimar os eventos não se baseia em dados históricos, então existe **risco de cronograma** para P.*
  - $\exists\, p$. Project(p) $\wedge$ Schedule(x,p) $\wedge \neg$ DataHistorical(x) $\rightarrow$ ScheduleRisk(p)

Pessoal:

- *Se existe algum projeto P no qual existe uma equipe de trabalho e esta equipe pode ser insuficiente para o desenvolvimento do projeto, então existe **risco de equipe** para P.*
  - $\exists\, p$. Project(p) $\wedge$ Staff(x,p) $\wedge \neg$ AdequateNumber(x) $\rightarrow$ StaffRisk(p)
- *Se existe algum projeto P no qual existe uma equipe de trabalho e esta equipe pode não apresentar o perfil requerido para o desenvolvimento do projeto, então existe **risco de equipe** para P.*
  - $\exists\, p$. Project(p) $\wedge$ Staff(x,p) $\wedge \neg$ TechnicalSkills(x) $\rightarrow$ StaffRisk(p)
- *Se existe algum projeto P no qual existe uma equipe de trabalho e esta equipe pode não ter o domínio na área de aplicação envolvida, então existe **risco de equipe** para P.*
  - $\exists\, p$. Project(p) $\wedge$ Staff(x,p) $\wedge \neg$ ApplicationDomain(x) $\rightarrow$ StaffRisk(p)

- *Se existe algum projeto P no qual existe uma equipe de trabalho e esta equipe pode não apresentar disponibilidade quando requisitada, então existe* ***risco de equipe*** *para P.*

  - $\exists\, p$. Project(p) $\wedge$ Staff(x,p) $\wedge \neg$ AccessPeople(x) $\rightarrow$ StaffRisk(p)

Orçamento:

- *Se existe algum projeto P no qual existe um orçamento com eventos internos e externos, cujos custos previstos não são realistas, então existe* ***risco de orçamento*** *para P.*

  - $\exists\, p$. Project(p) $\wedge$ Budget(x,p) $\wedge \neg$ RealisticEstimate(x) $\rightarrow$ BudgetRisk(p)

- *Se existe algum projeto P no qual existe um orçamento com eventos internos e externos, cujas mudanças no orçamento não são compatíveis com as mudanças nos requisitos do Projeto, então existe* ***risco de orçamento*** *para P.*

  - $\exists\, p$. Project(p) $\wedge$ Budget(x,p) $\wedge \neg$ RealisticChanges(x) $\rightarrow$ BudgetRisk(p)

Facilidades:

- *Se existe algum projeto P no qual os recursos disponíveis para desenvolvimento, integração e teste do produto não são adequados, então existe* ***risco de facilidades*** *para P.*

  - $\exists\, p$. Project(p) $\wedge$ Facilities(x,p) $\wedge \neg$ Adequate(x) $\rightarrow$ FacilitiesRisk(p)

Para a Classe ***Risco de Contrato*** e suas subclasses foram definidos os seguintes axiomas:

Tipo de Contrato:

- *Se existe algum projeto P no qual existe um contrato com condições de pagamento fixas ou variáveis, e ou requisitos como a alocação de pessoal por parte do cliente, então existe* ***risco de tipo de contrato*** *para P.*

  - $\exists\, p$. Project(p) $\wedge$ TypeContract(x,p) $\wedge \neg$ FixedPrice(x) $\rightarrow$ TypeOfContractRisk (p)

- *Se existe algum projeto P no qual existe um contrato que envolve um grande valor financeiro, então existe* ***risco de tipo de contrato*** *para P.*

  - $\exists\, p$. Project(p) $\wedge$ TypeContract(x,p) $\wedge$ ExcessiveAmount(x) $\rightarrow$ TypeOfContractRisk (p)

- *Se existe algum projeto P no qual existe um contrato que envolve um cliente desonesto, então existe* ***risco de tipo de contrato*** *para P.*

  - $\exists\, p$. Project(p) $\wedge$ TypeContract(x,p) $\wedge$ PickyCustomer(x) $\rightarrow$ TypeOfContractRisk(p)

Restrições de Contrato:

- *Se existe algum projeto P no qual existem diretivas de contrato, como o uso específico de uma metodologia ou equipamento, ou ainda específico software, então existe* ***risco de restrições de contrato*** *para P.*

  - $\exists\, p$. Project(p) $\wedge$ RestrictionsContract(x,p) $\wedge$ (CotsSoftware(x) v SpecifEquipament(x) v SpecifDevelopmentMethods(x) $\rightarrow$ ContractRestrictionsRisk(p)

Dependências Contratuais:

- *Se existe algum projeto P no qual existe dependência contratual de outros contratos ou fornecedores de produtos ou serviços, então existe* ***risco de dependência contratual*** *para P.*

  - $\exists\, p, x$. Project(p) $\wedge$ DependenciesContract(x,p) $\wedge$ (ExternalProducts(x) v ExternalServices(x)) $\rightarrow$ DependenciesRisk(p)

Para a Classe ***Risco de Interfaces*** e suas subclasses foram definidos os seguintes axiomas:

Cliente:

- *Se existe algum projeto P no qual existe uma equipe de trabalho do cliente e esta equipe pode não apresentar o perfil e a experiência requeridos para o desenvolvimento do projeto, então existe* ***risco de cliente*** *para P.*

- $\exists p, x$. Project(p) $\wedge$ Customer(x,p) $\wedge \neg$ DomainExpertise(x) $\rightarrow$ CustomerRisk(p)

- *Se existe algum projeto P no qual existe uma equipe de trabalho do cliente e existe dificuldade de relacionamento ou de se obter concordância e aprovação dos produtos, então existe* ***risco de cliente*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ Customer(x,p) $\wedge \neg$ AdequateCommunication(x) $\rightarrow$ CustomerRisk(p)

- *Se existe algum projeto P no qual existe dificuldade de se ter acesso a certos setores da organização ou a não disponibilidade de comunicação direta, então existe* ***risco de cliente*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ Customer(x,p) $\wedge \neg$ Access(x) $\rightarrow$ CustomerRisk(p)

Contratos Associados:

- *Se existe algum projeto P no qual existem contratos associados, podendo gerar conflitos de prioridade entre os mesmos, então existe* ***risco de contratos associados*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ AssociateContract(x,p) $\wedge \neg$ Cooperation(x) $\rightarrow$ AssociateContractRisk(p)

- *Se existe algum projeto P no qual existem contratos associados, podendo gerar conflitos de interface com os sistemas desenvolvidos por outras empresas, então existe* ***risco de contratos associados*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ AssociateContract(x,p) $\wedge \neg$ AdequareInterface(x) $\rightarrow$ AssociateContractRisk(p)

- *Se existe algum projeto P no qual existem contratos associados, podendo gerar conflitos de cooperação nas mudanças de configuração, então existe* ***risco de contratos associados*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ AssociateContract(x,p) $\wedge \neg$ GettingSchedule(x) $\rightarrow$ AssociateContractRisk(p)

Sub-Contratação:

- *Se existe algum projeto P no qual existe sub-contratação, podendo gerar conflitos nos mecanismos de gerenciamento dos sub-contratados, então existe* ***risco de sub-contratação*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{SubContractors}(x,p) \wedge \neg\, \text{Administration}(x) \rightarrow \text{SubContractorRisk}(p)$

- *Se existe algum projeto P no qual existe sub-contratação, podendo gerar conflitos na transferência de tecnologia e conhecimentos aos sub-contratados, então existe* ***risco de sub-contratação*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{SubContractors}(x,p) \wedge \neg\, \text{TransferringTechnology}(x) \rightarrow \text{SubContractorRisk}(p)$

Contrato Principal:

- *Se existe algum projeto P o qual é um contrato secundário de um outro contrato principal, podendo gerar conflitos nos mecanismos de gerenciamento do contrato secundário, então existe* ***risco de contrato principal*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{PrimeContractor}(x,p) \wedge \text{ComplexReportingArrangements}(x) \rightarrow \text{PrimeContractorRisk}(p)$

- *Se existe algum projeto P o qual é um contrato secundário de um outro contrato principal, podendo gerar problemas de dependência tecnológica do projeto principal, então existe* ***risco de contrato principal*** *para P.*

  - $\exists\, p, x\,.\ \text{Project}(p) \wedge \text{PrimeContractor}(x,p) \wedge \neg\, \text{TransferringTechnology}(x) \rightarrow \text{PrimeContractorRisk}(p)$

Administração Corporativa:

- *Se existe algum projeto P o qual é administrado pela alta administração da organização, existindo problemas de comunicação e dificuldade de acesso, então existe* ***risco de administração corporativa*** *para P.*

- $\exists p, x$. Project(p) $\wedge$ CorporateManagement(x,p) $\wedge \neg$ AdequateCommunication (x) $\rightarrow$ CorporateManagementRisk(p)

- *Se existe algum projeto P o qual é administrado pela alta administração da organização, existe dificuldade de apoio à gerência do projeto na solução de problemas, então existe* ***risco de administração corporativa*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ CorporateManagement(x,p) $\wedge \neg$ AdequateSupport (x) $\rightarrow$ CorporateManagementRisk(p)

Vendedores:

- *Se existe algum projeto P para o qual existem vendedores que dependem da liberação e suporte de componentes críticos, então existe* ***risco de vendedores*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ Vendors(x,p) $\wedge \neg$ ComponentsDeliveries (x) $\rightarrow$ VendorsRisk(p)

Políticas:

- *Se existe algum projeto P, existe relacionamento entre cliente, fornecedor, contratantes associados, sub-contratantes, para os quais as políticas podem afetar o relacionamento e as decisões técnicas, existindo então* ***risco de políticas*** *para P.*

  - $\exists p, x$. Project(p) $\wedge$ Politics(x,p) $\rightarrow$ PoliticsRisk(p)

3.2.3.1. Relacionamentos da Classe Restrições de Programa

- ***Risco de pessoal*** *acarreta* ***risco de cronograma***

  - $\exists p$. Project(p) $\wedge$ StaffRisk(p) $\rightarrow$ ScheduleRisk(p)

- ***Risco de pessoal*** *acarreta* ***risco de orçamento***

  - $\exists p$. Project(p) $\wedge$ StaffRisk(p) $\rightarrow$ BudgetRisk(p)

- ***Risco de cronograma*** *causa* ***risco de orçamento***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{ScheduleRisk}(p) \rightarrow \text{BudgetRisk}(p)$
- ***Risco de facilidades*** *carreta* ***risco de cronograma***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{FacilitiesRisk}(p) \rightarrow \text{ScheduleRisk}(p)$
- ***Risco de cronograma*** *causa* ***risco de tipo de contrato***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{ScheduleRisk}(p) \rightarrow \text{TypeContractRisk}(p)$
- ***Risco de orçamento*** *acarreta* ***risco de tipo de contrato***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{BudgetRisk}(p) \rightarrow \text{TypeContractRisk}(p)$
- ***Risco de restrições de contrato*** acarreta ***risco de cronograma***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{ContractRestrictionsRisk}(p) \rightarrow \text{ScheduleRisk}(p)$
- ***Risco de cliente*** *acarreta* ***risco de cronograma***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{CustomerRisk}(p) \rightarrow \text{ScheduleRisk}(p)$
- ***Risco de contratos associados*** acarreta ***risco de cronograma***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{AssociateContractorsRisk}(p) \rightarrow \text{ScheduleRisk}(p)$
- ***Risco de sub-contratação*** acarreta ***risco de cronograma***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{SubcontractorsRisk}(p) \rightarrow \text{ScheduleRisk}(p)$
- ***Risco de contrato principal*** *causa* ***risco de cronograma***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{PrimeContractRisk}(p) \rightarrow \text{ScheduleRisk}(p)$
- ***Risco de administração corporativa*** *acarreta* ***risco de cronograma***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{CorporateManagementRisk}(p) \rightarrow \text{ScheduleRisk}(p)$

- ***Risco de políticas*** *acarreta* ***risco de cronograma***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{PoliticsRisk}(p) \rightarrow \text{ScheduleRisk}(p)$
- ***Risco de cronograma*** *acarreta* ***risco de vendedores***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{ScheduleRisk}(p) \rightarrow \text{VendorsRisk}(p)$

3.2.4. Engenharia de Produto *versus* Ambiente de Desenvolvimento

Os relacionamentos e axiomas a seguir referem-se as classes Engenharia de Produto e Ambiente de Desenvolvimento.

- ***Risco de atitudes de qualidade*** *acarreta* ***risco de engenharia de produto***
  - $\exists\, p\ .\ \text{Project}(p) \wedge \text{CapacityRisk}(p) \rightarrow \text{ImplementationRisk}(p)$
- ***Risco de experiência gerencial acarreta risco de escala***
  - $\exists\, p\ .\ \text{Project}(p) \wedge \text{ManagemetExperienceRisk}(p) \rightarrow \text{ScaleRisk}(p)$
- ***Risco de experiência gerencial*** *acarreta* ***risco de processo***
  - $\exists\, p\ .\ \text{Project}(p) \wedge \text{ManagemetExperienceRisk}(p) \rightarrow \text{ProcessRisk}(p)$
- ***Risco de gerência de configuração*** *acarreta* ***risco de integração de produto***
  - $\exists\, p\ .\ \text{Project}(p) \wedge \text{ConfigurationManagementRisk}(p) \rightarrow \text{ProductIntegrationAndTestRisk}(p)$
- ***Risco de gerência de configuração*** *acarreta* ***risco de integração de sistema***
  - $\exists\, p\ .\ \text{Project}(p) \wedge \text{ConfigurationManagementRisk}(p) \rightarrow \text{SystemIntegrationAndTestRisk}(p)$
- ***Risco de métodos de gerenciamento*** *acarreta* ***risco de engenharia de produto***

- $\exists p$. Project(p) $\wedge$ ManagementMethodsRisk(p) $\rightarrow$ ProductEngineeringRisk(p)

- ***Risco de moral*** *acarreta* ***risco de engenharia de produto***
  - $\exists p$. Project(p) $\wedge$ MoraleRisk(p) $\rightarrow$ ProductEngineeringRisk(p)
- ***Risco de processo de desenvolvimento*** *acarreta* ***risco de engenharia de produto***
  - $\exists p$. Project(p) $\wedge$ DevelopmentProcessRisk(p) $\rightarrow$ ProductEngineeringRisk(p)
- ***Risco de processo de gerenciamento*** *acarreta* ***risco de engenharia de produto***
  - $\exists p$. Project(p) $\wedge$ ManagementProcessRisk(p) $\rightarrow$ ProductEngineeringRisk(p)
- ***Risco de sistema de desenvolvimento*** *acarreta* ***risco de requisito***
  - $\exists p$. Project(p) $\wedge$ DevelopmentSystemRisk(p) $\rightarrow$ RequirementRisk(p)
- ***Risco de métodos de gerenciamento*** *acarreta* ***risco de engenharia de produto***
  - $\exists p$. Project(p) $\wedge$ ManagementMethodsRisk(p) $\rightarrow$ ProductEngineeringRisk(p)
- ***Risco de garantia da qualidade*** *acarreta* ***risco de engenharia de produto***
  - $\exists p$. Project(p) $\wedge$ QualityAssuranceRisk(p) $\rightarrow$ ProductEngineeringRisk(p)
- ***Risco de adequabilidade do processo de desenvolvimento*** *acarreta* ***um risco de dificuldade***
  - $\exists p$. Project(p) $\wedge$ DevelopmentProcessSuitabilityRisk(p) $\rightarrow$ DifficultyRisk(p)
- ***Risco de capacidade*** *acarreta um* ***risco de implementação***
  - $\exists p$. Project(p) $\wedge$ CapacityRisk(p) $\rightarrow$ ImplementationRisk(p)
- ***Risco de entrega*** *acarreta um* ***risco de implementação***
  - $\exists p$. Project(p) $\wedge$ DeliverabilityRisk(p) $\rightarrow$ ImplementationRisk(p)
- ***Risco de capacidade*** *acarreta* ***um risco de performance***

- $\exists p$. Project(p) $\wedge$ CapacityRisk(p) $\rightarrow$ PerformanceRisk(p)

- ***Risco de usabilidade*** *acarreta* ***um risco de testabilidade***

  - $\exists p$. Project(p) $\wedge$ UsabilityRisk(p) $\rightarrow$ TestabilityRisk(p)

3.2.5. Engenharia de Produto versus Restrições de Programa

Os relacionamentos e axiomas a seguir referem-se as classes Engenharia de Produto e Restrições de Programa.

- ***Risco de cronograma*** *acarreta* ***risco de engenharia de produto***

  - $\exists p$. Project(p) $\wedge$ ScheduleRisk(p) $\rightarrow$ ProductEngineeringRisk(p)

- ***Risco de contratados associados*** *acarreta* ***risco de software externo***

  - $\exists p$. Project(p) $\wedge$ AssociateContractorsRisk(p) $\rightarrow$ NonDevelopmentalSoftwareRisk(p)

- ***Risco de dependências*** *acarreta* ***risco de software externo***

  - $\exists p$. Project(p) $\wedge$ DependeciesRisk(p) $\rightarrow$ NonDevelopmentalSoftwareRisk(p)

- ***Risco de facilitadores*** *acarreta* ***risco de implementação***

  - $\exists p$. Project(p) $\wedge$ FacilitiesRisk(p) $\rightarrow$ ImplementationRisk(p)

- ***Risco de facilitadores*** *acarreta* ***risco de integração e testes***

  - $\exists p$. Project(p) $\wedge$ FacilitiesRisk(p) $\rightarrow$ IntegrationAndTestRisk(p)

- ***Risco de fornecedores*** *acarreta* ***risco de engenharia de produto***

  - $\exists p$. Project(p) $\wedge$ VendorsRisk(p) $\rightarrow$ ProductEngineeringRisk(p)

- ***Risco de pessoal*** *acarreta* ***risco de engenharia de produto***

- $\exists p$. Project(p) $\wedge$ StaffRisk(p) $\rightarrow$ ProductEngineeringRisk(p)

- ***Risco de políticas*** *acarreta* ***risco de design***

  - $\exists p$. Project(p) $\wedge$ PoliticsRisk(p) $\rightarrow$ DesignRisk(p)

- ***Risco de políticas*** *acarreta* ***risco de engenharia de produto***

  - $\exists p$. Project(p) $\wedge$ PoliticsRisk(p) $\rightarrow$ ProductEngineeringRisk(p)

3.2.6. Ambiente de Desenvolvimento versus Restrições de Programa

Os relacionamentos e axiomas a seguir referem-se as classes Ambiente de Desenvolvimento e Restrições de Programa.

- ***Risco de capacidade*** *acarreta* ***um risco de cronograma***

  - $\exists p$. Project(p) $\wedge$ CapacityRisk(p) $\rightarrow$ ScheduleRisk(p)

- ***Risco de formalidade*** *acarreta* ***um risco de facilitadores***

  - $\exists p$. Project(p) $\wedge$ FormalityRisk(p) $\rightarrow$ FacilitiesRisk(p)

- ***Risco de adequabilidade do processo de desenvolvimento*** *acarreta* ***um risco de restrições***

  - $\exists p$. Project(p) $\wedge$ DevelopmentProcessSuitabilityRisk(p) $\rightarrow$ RestrictionsRisk(p)

- ***Risco de adequabilidade do sistema de desenvolvimento*** *acarreta* ***um risco de restrições***

  - $\exists p$. Project(p) $\wedge$ DevelopmentSystemSuitabilityRisk(p) $\rightarrow$ RestrictionsRisk(p)

- ***Risco de entrega*** *acarreta* ***um risco de venda***

  - $\exists p$. Project(p) $\wedge$ DeliverabilityRisk(p) $\rightarrow$ VendorRisk(p)

- ***Risco de suporte do sistema*** *acarreta* ***um risco de venda***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{SystemSupportRisk}(p) \rightarrow \text{VendorRisk}(p)$
- ***Risco de confiabilidade*** *acarreta* ***um risco de venda***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{ReliabilityRisk}(p) \rightarrow \text{VendorRisk}(p)$
- ***Risco de cronograma*** *acarreta* ***risco de gerenciamento de processos.***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{ScheduleRisk}(p) \rightarrow \text{ManagementProcessRisk}(p)$
- ***Risco de pessoal*** *acarreta* ***risco de gerenciamento de processos.***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{StaffRisk}(p) \rightarrow \text{ManagementProcessRisk}(p)$
- ***Risco de políticas*** *acarreta* ***risco de gerenciamento de processos.***
  - $\exists\, p.\ \text{Project}(p) \wedge \text{PoliticsRisk}(p) \rightarrow \text{ManagementProcessRisk}(p)$

CAPÍTULO 4

ARQUITETURA E PROTÓTIPO

---

*Este capítulo tem por objetivo apresentar uma descrição geral da arquitetura do protótipo desenvolvido para validar o mecanismo de identificação de riscos para um projeto da versão **alfa** da OntoPRIME. Foi dada especial atenção à descrição desta arquitetura uma vez que ela deverá ser utilizada **não apenas** para este protótipo, mas também servirá como um modelo para aplicações futuras que façam uso da OntoPRIME. A seguir, será dada a descrição geral do padrão arquitetural utilizado e da distribuição dos componentes dentro da arquitetura descrita.*

4.1. DESCRIÇÃO GERAL DA ARQUITETURA

O padrão arquitetural utilizado no desenvolvimento do protótipo é uma instância do **Padrão Arquitetural em Camadas**, caracterizado pela divisão dos diferentes níveis do sistema em diferentes camadas. Em particular, a versão utilizada foi adaptada para atender às necessidades de um sistema inteligente típico. Uma arquitetura em camadas como esta em geral possui as camadas mostradas na Figura 5:

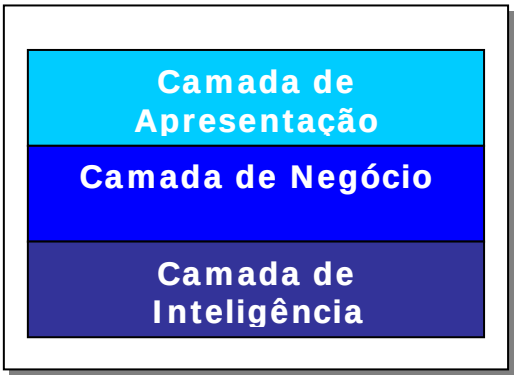

Figura 5 - Arquitetura em Camadas

- **Apresentação:** Esta camada é responsável pela interação do usuário com o sistema, por meio da qual são realizadas as operações de entrada e saída. Pode ser implementada graficamente, por meio de uma *GUI – Graphical User Interface* ou Interface Gráfica do Usuário.

- **Negócio:** Contém as entidades que modelam e implementam os conceitos utilizados no domínio da aplicação.

- **Camada de Inteligência:** Responsável por implementar a inteligência do sistema, fornecendo uma abstração sobre qual sistema dedutivo é utilizado e sobre como ele é implementado. A aplicação em questão foi projetada de modo a permitir completa independência do sistema dedutivo utilizado.

4.2. ORGANIZAÇÃO DO SISTEMA

A Figura 6 mostra a distribuição dos elementos do sistema entre as diferentes camadas da arquitetura, citadas na seção anterior:

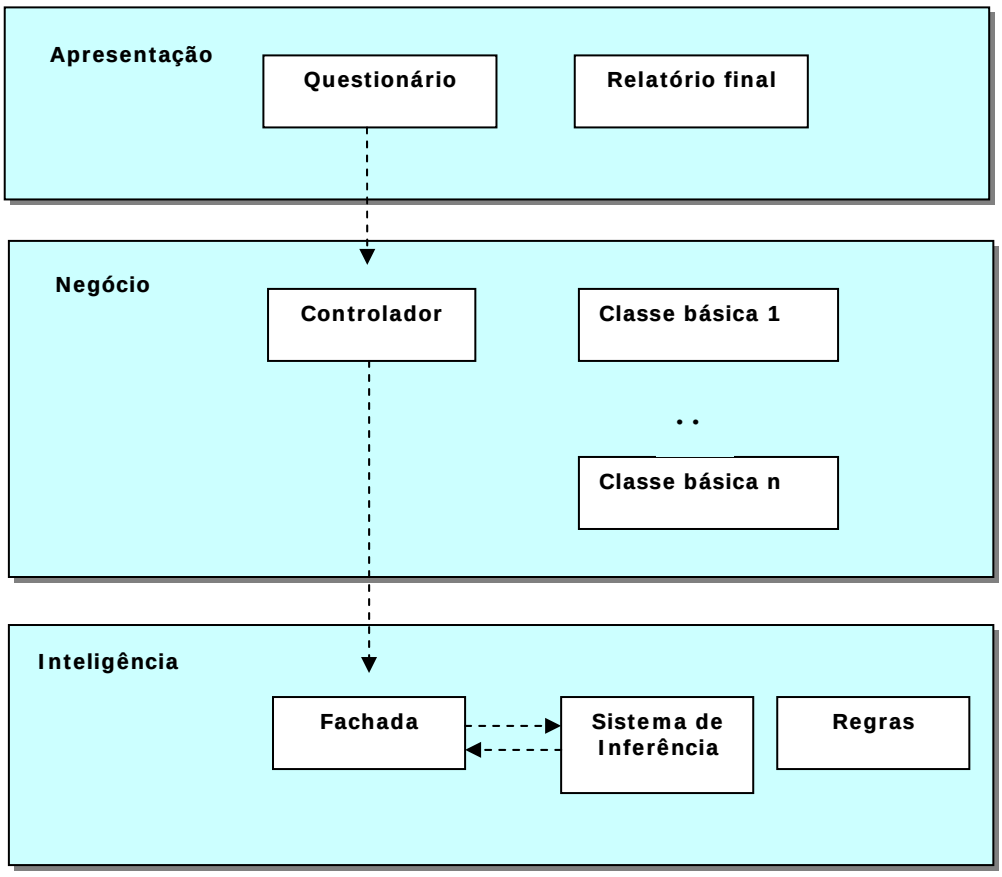

Figura 6 - Organização do Sistema

4.2.1. Camada de Apresentação

Esta camada possui os elementos de interface com o usuário, neste caso, um questionário contendo perguntas sobre as características do projeto a ser analisado, baseado no questionário proposto pelo Paradigma de Gerenciamento de Riscos do SEI. A interface foi construída utilizando componentes do toolkit Swing [ZUKOWSKI, 1999] da plataforma Java [GOSLING, 2000], linguagem utilizada na implementação de todo o sistema. Esta interface comunica-se com o sistema através do controlador da camada de negócio.

4.2.2. Camada de Negócio

Camada responsável por implementar a lógica de negócio da aplicação. Contém os seguintes elementos:

- **Controlador**: Classe que fornece uma interface para os serviços oferecidos pela aplicação. Comunica-se diretamente com a fachada da camada de inteligência.
- **Classes básicas**: Representam os conceitos utilizados no sistema (ie.: risco, projeto, características do projeto), bem como as classes ou conceitos presentes na ontologia desenvolvida.

4.2.3. Camada de Inteligência

Camada responsável pela implementação da inteligência do sistema, projetada de forma que seja possível abstrair nas camadas superiores qual mecanismo dedutivo é utilizado (ex.: lógica de primeira ordem, lógica *fuzzy* [CHEN, 1995], etc.). Contém os seguintes elementos:

- **Fachada**: Interface para os serviços oferecidos pela camada de inteligência. Sua implementação instancia o sistema de inferência e atua como um **adaptador** para o sistema de inteligência utilizado, fazendo a comunicação entre este e a camada imediatamente superior.

- **Sistema de Inferência**: Módulo que encapsula o mecanismo utilizado para implementar a inteligência do sistema. Em particular, o sistema de inferência utilizado nesta aplicação é o *JEOPS* [FIGUEIRA, 2001], que utiliza o formalismo de **lógica de primeira ordem** para representar as regras e axiomas da ontologia, uma vez que este foi o formalismo utilizado para representar tais regras e axiomas na construção da ontologia.

4.3. Pacotes do Sistema

Os pacotes são agrupamentos lógicos das classes do sistema. A divisão do sistema em pacotes pode ser representada por duas formas: a nível de camada ou a nível de implementação. Do ponto de vista conceitual ou de análise, costuma-se utilizar a primeira representação, de modo que a segunda é utilizada em um contexto mais concreto, durante a fase de projeto. Nas próximas seções, serão mostradas as duas representações da arquitetura utilizando **notação UML** [RUMBAUGH, 2004]**.**

4.3.1. Pacotes em nível de camadas

Na Figura 7 é exibida uma divisão em que cada pacote mostrado corresponde basicamente a uma camada da arquitetura. Para cada pacote desta divisão é descrito o conjunto de classes e componentes que dele fazem parte. Esta é uma visão abstrata, com foco na arquitetura e mais próxima do modelo de análise. A Figura 7 pode ser considerada como uma versão simplificada da Figura 6, porém utilizando notação de pacotes.

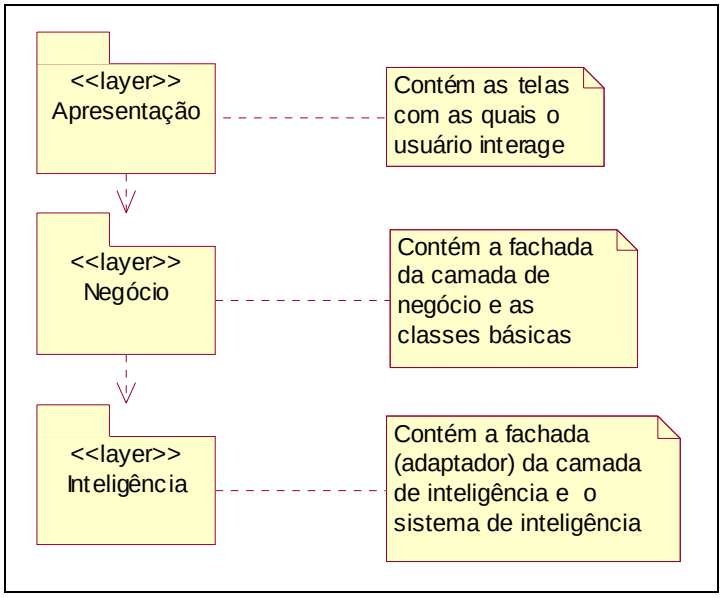

Figura 7 - Pacote a nível de camadas

4.3.2. Pacotes em nível de implementação

A Figura 8 exibe a real divisão do sistema em pacotes, de modo que cada pacote equivale a um diretório no sistema de arquivos ou a um pacote de acordo com o conceito correspondente na linguagem Java. Esta divisão é mais concreta do que a divisão em nível de camadas e aproxima-se mais do modelo de projeto.

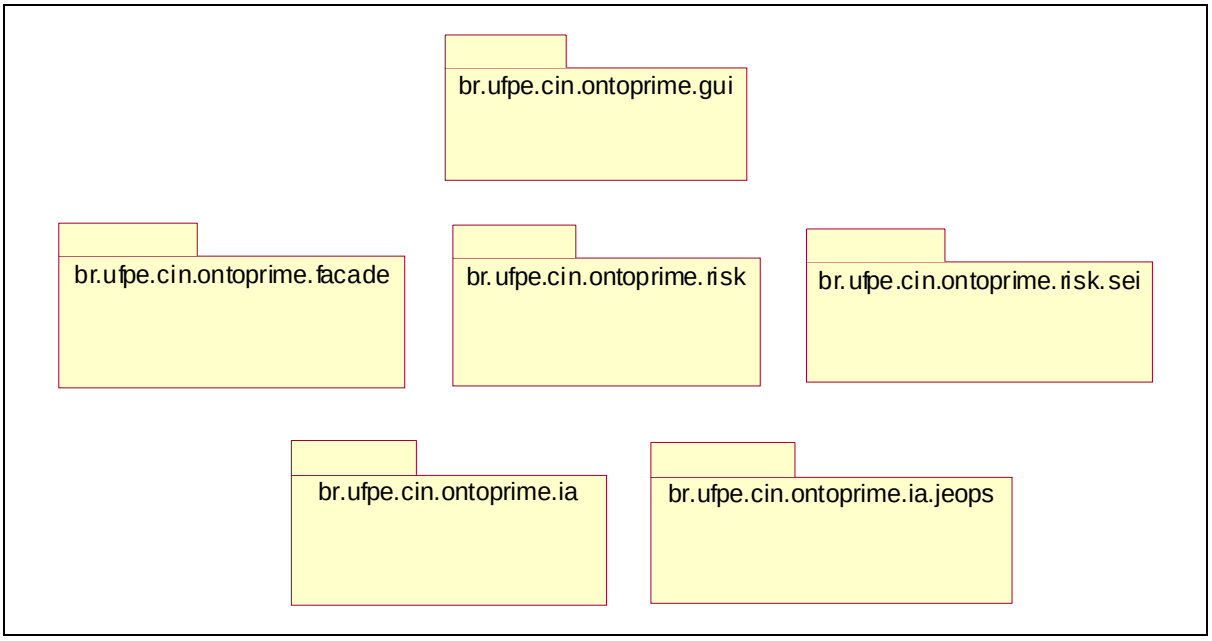

Figura 8 - Pacotes em nível de implementação

A seguir segue uma descrição de cada um dos pacotes exibidos :

- *br.ufpe.cin.ontoprime.gui:* contém as classes referentes à interface gráfica.
- *br.ufpe.cin.ontoprime.facade:* contém o controlador (fachada da camada de negócio).
- *br.ufpe.cin.ontoprime.risk:* contém as classes que representam os conceitos comuns do domínio da aplicação, como **Risco** e **Projeto**, por exemplo. Essas classes são portanto independentes da taxonomia/ontologia utilizada.
- *br.ufpe.cin.ontoprime.risk.sei:* contém as classes da taxonomia proposta pelo Paradigma de Gerenciamento de Risco do SEI - *Software Engineering Institute*. Na implementação atual da OntoPRIME, essas classes são as mesmas utilizadas pela ontologia desenvolvida.
- *br.ufpe.cin.ontoprime.risk.ia:* contém as classes que fornecem a interface externa para a camada de inteligência.
- *br.ufpe.cin.ontoprime.risk.ia.jeops:* contém as classes que interagem diretamente com a ferramenta JEOPS, como a classe que implementa a fachada da camada de inteligência (cuja interface está na camada do item anterior) e as classes que implementam os axiomas da ontologia (traduzidos para *regras* na implementação), além dos componentes do JEOPS.

CAPÍTULO 5

TRABALHOS FUTUROS E CONSIDERAÇÕES FINAIS

Este capítulo apresenta considerações finais e possíveis alternativas de trabalhos futuros como extensão da OntoPRIME.

A gerência de risco adiciona à gerência de projetos, em ambientes organizacionais, uma abordagem estruturada para identificação, análise e controle de riscos ao longo da execução de projetos. As características da gerência de risco (perspectiva global, visão orientada ao futuro, comunicação aberta, gerenciamento integrado, processo contínuo) podem reforçar a natureza sistemática e pró-ativa da gerência de projetos tornando-a eficiente e eficaz.

Neste estudo procurou-se mostrar a importância e a grande utilidade que a implementação de uma estrutura de linguagem – OntoPRIME, baseada no gerenciamento de risco pode trazer para a organização, inclusive porque é um esforço realizado para alcance da maturidade organizacional e cultural

5.1. TRABALHOS FUTUROS

Com este trabalho pretendeu-se estudar todos os aspectos da gerência de risco que são relevantes para os Ambientes de Desenvolvimento de Software, com vistas à utilização de um vocabulário comum que subsidie as atividades e tarefas destes ambientes, no que diz respeito ao gerenciamento de riscos. Trabalhos futuros poderão preocupar-se com:

Estudo aprofundado das atividades de gerência de risco e das atividades de seleção e priorização de projetos, baseadas nas técnicas da gerência de portfólio de projetos;

Extensão da OntoPRIME para suportar as características dos Ambientes de Desenvolvimento de Múltiplos Projetos e Software e Seleção de Portfólio de Projetos;

Implementação das fases finais da OntoPRIME de acordo com a metodologia de desenvolvimento proposta, seção 1.4.

Desenvolvimento de modelo para a gerência de risco, através de uma abordagem contínua, em um ambiente de Múltiplos Projetos Organizacionais, levando em consideração todo o conhecimento e experiência adquiridos pela organização em projetos passados, subsidiado pela utilização da OntoPRIME.

5.2. CONSIDERAÇÕES FINAIS

Conforme ressaltado, neste trabalho, a efetiva utilização de processos de gerenciamento de riscos não é uma tarefa fácil, porém essencial para a melhoria do processo de desenvolvimento de software e por conseguinte para a garantia da qualidade, no que diz respeito ao sucesso dos projetos.

O conjunto de atividades voltadas para a gerência de risco propicia ao gerente de projeto uma ampla visão dos projetos em execução, focando diversos aspectos tais como: garantia da qualidade, progresso, produtividade, acompanhamento de esforço e variação de tamanho do software. A utilização de informações de projetos realizados e dos projetos já em andamento, com certeza, facilitaria a atuação e decisão da gerência para a conclusão do projeto, Figura 9. Dessa forma, a utilização da OntoPRIME estaria promovendo a melhoria do gerenciamento de projetos, e conseqüentemente do desenvolvimento de produtos de software.

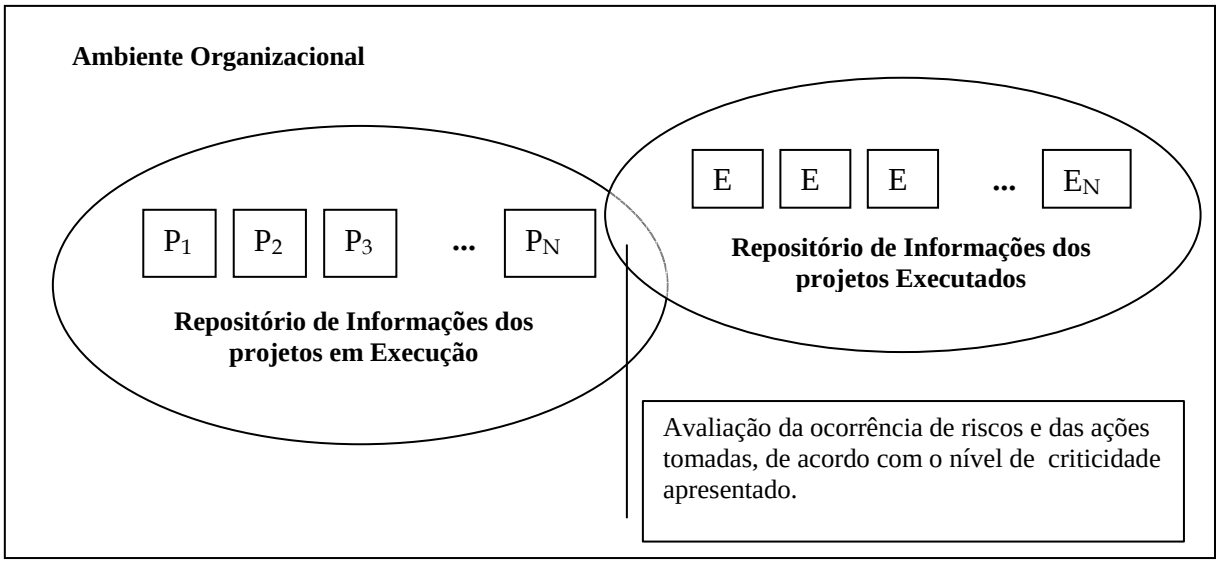

Figura 9 – Visão de Execução de Projetos

Através da Figura 9, tem-se uma visão genérica do subconjunto de projetos executados e em execução dentro de uma organização que poderão fornecer subsídios para a gerência de projeto na identificação do perfil dos riscos, tratamento e controle. A OntoPRIME estaria presente em qualquer das atividades do processo de gerenciamento de riscos, promovendo o vocabulário a ser utilizado, onde:

- Na identificação dos riscos de projetos e entre projetos. Não só os executados, bem como os em andamento;
- Na análise, priorização e categorização dos riscos de acordo com a estratégia organizacional definida;
- Baseada na priorização dos riscos, definição dos planos de contingência;
- No monitoramento e controle dos riscos, uma vez que estaria sendo utilizada através de um processo contínuo, sistemático e incremental.

Desta forma, a OntoPRIME estaria garantindo a efetividade e eficácia do processo de gerência de risco facilitando a análise e rastreamento regular dos riscos dentro do ambiente organizacional através de todo o processo de desenvolvimento de software.

REFERÊNCIAS BIBLIOGRÁFICAS

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

[BOEHM, 1991] BOEHM, B. W. Software Risk Management: principles and practices, *IEEE Software*, Volume 8. No1. pp 32-40. 1991.

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

[CHARETTE, 1990] CHARETTE, R. *Application Strategies for Risk Analysis*. New York: MultiScience Press. pp 17-21. 1990.

[CHEN, 1995] CHEN, C. H. *Fuzzy Logic and Neural Network Handbook.* McGraw-Hill Publishing Company, 1995.

[DEMARCO, 2003] De MARCO, T e LISTER, T. *Waltzing with Bears*. New York: Dorset House Publishing. 2003.

[FIGUEIRA, 2001] FIGUEIRA, C. S. *JEOPS – The Java Embedded Object Production System.* http://www.cin.ufpe.br/~jeops/. Visitado em 09/09/04.

[GOSLING, 2000] GOSLING, J., JOY, B., STEELE, G. e BRACHA, G. *The Java Language Specification.* Sun Microsystems, 2000.

[HALL, 1998] HALL, E. M. *Managing Risk – Methods for Software Systems Development*. Addison-Wesley. pp 88-103. 1998.

[HIGUERA, 1996] HIGUERA, R.P. e HAIMES, Y.Y. Software Risk Management. Technical Report – CMU/SEI-96-TR-012. . Pittsburgh, PA: *Software Engineering Institute*, Carnegie Mellon University. USA. 1996.

[HOUAISS, 2001] DICIONÁRIO HOUAISS DA LÍNGUA PORTUGUESA. 1ª edição. Rio de Janeiro: Editora Objetiva. 2001.

[KNIGHT, 1921] KNIGHT, F.H. *Risk, Uncertainty and Profit*. Houghton Mifflin, Boston. pp 22-24. 1921.

[PRESSMAN, 1995] PRESSMAN, R. S. *Engenharia de Software*. São Paulo: Ed. Makron Books, 1995.

[RUMBAUGH, 2004] RUMBAUGH, J., JACOBSON, IVAR, BOOCH, G.. *The Unified Modeling Language Reference Manual.* 2a. Edição. Addison-Wesley Object Technology Series, 2004.

[WEBSTERS, 1993] WEBSTER'S THIRD NEW INTERNATIONAL DICTIONARY. Merriam-Webster, Incorporated. Konemann. 1993.

[ZUKOWSKI, 1999] ZUKOWSKI, J. e STANCHFIELD, S.. *Fundamentals of JFC/Swing.* http://java.sun.com/developer/onlineTraining/GUI/Swing1/. Visitado em 09/09/04.