UFSM – UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ABORDAGEM MULTICRITÉRIOS PARA ADAPTAÇÃO DE PROCESSOS DE SOFTWARE BASEADA EM SITUATIONAL METHOD ENGINEERING

DISSERTAÇÃO DE MESTRADO

Guilherme Vaz Pereira

Santa Maria, RS, Brasil

2012

ABORDAGEM MULTICRITÉRIOS PARA ADAPTAÇÃO DE PROCESSOS DE SOFTWARE BASEADA EM SITUATIONAL METHOD ENGINEERING

Guilherme Vaz Pereira

Dissertação apresentada ao Curso de Mestrado em Computação do Programa de Pós-Graduação em Informática, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção de grau de
Mestre em Ciência da Computação

Orientadora: Profª.Drª.Lisandra Manzoni Fontoura

Santa Maria, RS, Brasil
2012

UFSM – UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

ABORDAGEM MULTICRITÉRIOS PARA ADAPTAÇÃO DE PROCESSOS DE SOFTWARE BASEADA EM SITUATIONAL METHOD ENGINEERING

elaborada por
Guilherme Vaz Pereira

como requisito parcial de obtenção de grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Lisandra Manzoni Fontoura, Dra.
(Presidente/Orientadora)

Iara Augustin, Dra. (UFSM)

Roberto Tom Price, Dr. (UFRGS)

Santa Maria, 20 de abril de 2012

Agradecimentos

A Deus.

Aos meus pais, Gilberto e Gilce, pelo incentivo e por seus esforços para proporcionar mais esta oportunidade de crescimento.

A Gabriela, minha irmã, amiga e companheira de apartamento durante o mestrado.

A minha namorada Jessica, pelo amor, carinho, apoio, e por tudo mais de bom que ela representa, desde a graduação.

A professora Lisandra Fontoura. Pela confiança e oportunidade de ingressar no mestrado. Pela atenção, dedicação e excelente orientação neste trabalho. Finalmente, pela sua amizade.

Aos meus colegas de mestrado, Fabrício Severo, Daniel Biasoli, Fabio Prass, Marco Copetti, Miguel Brasil e Rosana Wagner, Wagner Lorenz, pela toca de idéias e conhecimentos.

Ao Programa de Pós-Graduação em Informática da UFSM, por proporcionar a estrutura necessária para esta pesquisa.

Por fim, agradeço a todos os amigos que torceram pelo sucesso deste trabalho.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

ABORDAGEM MULTICRITÉRIOS PARA ADAPTAÇÃO DE PROCESSOS DE SOFTWARE BASEADA EM SITUATIONAL METHOD ENGINEERING

AUTOR: Guilherme Vaz Pereira
ORIENTADORA: Profª. Drª. Lisandra Manzoni Fontoura
Local e data da defesa: Santa Maria, 20 de abril de 2012.

As organizações de desenvolvimento de software estão envolvidas em um ambiente dinâmico, com diferentes cenários de acordo com as características e demandas específicas de cada projeto de software. Não há um processo de software que atenda as necessidades específicas de todos os projetos e/ou organizações. Assim, a adequação de um processo de desenvolvimento de software depende do contexto do projeto. *Situational Method Engineering* (SME) propõe a construção de métodos de desenvolvimento de software específicos para cada projeto de acordo com as características situacionais dos mesmos a partir de fragmentos de métodos armazenados em um repositório. Este trabalho propõe uma abordagem sistemática para a construção de processos de desenvolvimento de software específicos para cada projeto a partir da adaptação de processos, com base em conceitos de SME, chamada OSPTA – *Octopus SME Process Tailoring Approach.* OSPTA utiliza fragmentos de métodos armazenados em um repositório, os quais incluem práticas preconizadas por processos ágeis e planejados. Tais fragmentos são recuperados de acordo com um ou mais critério de adaptação, ou requisitos para o processo adaptado, e são priorizados de acordo com fatores de contextualização de projetos de software definidos no *Octopus Model.* A técnica usada para esta priorização é *Analytic Hierarchy Process* (AHP), para guiar os engenheiros de processo na escolha dos melhores fragmentos para serem incluídos no processo de software padrão da organização (PSPO), dando origem ao processo adaptado, específico para o projeto. Um metamodelo foi proposto para a definição de fragmentos de métodos para serem utilizados na abordagem. Foi desenvolvida uma ferramenta para apoiar a abordagem proposta. Estudos de caso considerando "riscos do projeto" como critério de adaptação foram elaborados para validar a abordagem.

Palavras-chave: Adaptação de Processos, Contexto do Projeto, Situational Method Engineering, Octopus Model, Analytic Hierarchy Process (AHP).

ABSTRACT

Master's Dissertation
Post-Graduation Program in Information Science
Federal University of Santa Maria

MULTI-CRITERIA APPROACH FOR SOFTWARE PROCESS TAILORING BASED ON SITUATIONAL METHOD ENGINEERING

AUTHOR: Guilherme Vaz Pereira
ADVISOR: Prof. Dra. Lisandra Manzoni Fontoura
Place and date: Santa Maria, April 20, 2011.

Software development organizations are involved in different scenarios with distinct projects in relation to specific project characteristics and demands. There isn´t a software process model appropriate for all projects and/or organizations. The best process depends of the project context. Situational Method Engineering (SME) proposes the building of specific software development methods for each project according to its situational characteristics from method fragments stored in a repository. This work proposes a systematic approach for building a specific development software process for each project through tailoring process based on SME concepts, called OSPTA – Octopus SME Process Tailoring Approach. OSPTA uses stored method fragments which include practices recommended by agile and planned process. These fragments are retrieved according to one or more tailoring criteria (tailoring process requirements) and are prioritized from contextual factors defined by *Octopus Model*. The technique used for this prioritization is *Analytic Hierarchy Process* (AHP) technique to guide the process engineers in choosing the best fragments to include into organization's standard software process (PSPO). A metamodel was proposed to define method fragments. A support tool was developed to validate the proposed approach through case studies which use "project risks" as tailoring criteria.

Keywords: Process Tailoring, Project Context, Situational Method Engineering, Octopus Model, Analytic Hierarchy Process (AHP).

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AHP	-	*Analytic Hierarchy Process*
CMMI	-	*Capability Maturity Model Integration*
DSDM	-	*Dynamic Systems Development Method*
FDD	-	*Feature Driven Development*
GRC	-	Governança, Gerenciamento de Riscos, e Conformidade
IEC	-	*International Electrotechnical Commission*
ISO	-	*International Organization for Standardization*
LSD	-	*Lean Software Development*
M^2F	-	*Metamodel for Method Fragments*
OMG	-	*Object Management Group*
OSPTA	-	*Octopus SME – Process Tailoring Approach*
PSPO	-	Processo de Software Padrão da Organização
RUP	-	*Rational Unified Process*
SEI	-	*Software Engineering Institute*
SE-MDM	-	*Software Engineering Metamodel for Development Methodologies*
SME	-	*Situational Method Engineering*
SPEM	-	*Software Process Engineering Metamodel*
XP	-	*Extreme programming*

SUMÁRIO

1. INTRODUCÃO

Processo de software é um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos, e artefatos que são necessários para conceber, desenvolver, implantar, e manter um produto de software (FUGGETA, 2000). Estes são representados por modelos de processo, os quais descrevem as práticas ou atividades que devem ser executadas para o desenvolvimento de software, a ordem de execução, a relação entre elas, e as características de cada atividade como artefatos de entrada e saída, papéis e *stakeholders* envolvidos (XU, 2008).

É amplamente aceito que um processo de software bem definido é essencial para melhorar a qualidade e a produtividade do desenvolvimento de software (XU, 2005). Porém, as organizações de desenvolvimento de software estão envolvidas em projetos diferentes que requerem estratégias particulares, pois possuem características distintas em relação a fatores como tecnologia, tamanho, equipe, riscos, requisitos do cliente, requisitos de qualidade e segurança, entre outros. Assim, definir um único processo de desenvolvimento para ser utilizado em todos os projetos da organização é inviável. A adequação de um processo de software depende das particularidades de cada projeto (ALEGRÍA, 2011).

O SEI, por meio do CMMI (SEI, 2010), propõe que seja definido um processo de software padrão da organização (PSPO) contendo elementos essenciais para todos os projetos e que este seja adaptado de acordo com as necessidades de cada projeto, dando origem a um processo específico para o projeto.

A adaptação de processos está relacionada ao ato de customizar uma definição de processo de desenvolvimento de software para atender necessidades específicas de um projeto, ajustar um processo de desenvolvimento de acordo com diferentes ambientes ou contextos (GINSBERG, 1995).

Entretanto, esta não é uma tarefa simples. Autores como Pedreira (2007), Magdaleno (2010) e Alegría (2011) afirmam que, freqüentemente, a adaptação de processos é executada de maneira *ad-hoc*, baseada apenas nas experiências dos gerentes, sem o uso de guias ou regras, sem levar em conta as características especificas

do projeto, possivelmente selecionando um processo que não seria a melhor alternativa para o projeto em questão.

Os processos de software podem ser classificados em ágeis e planejados, por exemplo, XP (*Extreme Programming*) (BECK, 2004) e RUP (*Rational Unified Process*) (RATIONAL, 2003), respectivamente. Através da adaptação de processos, por exemplo, é possível balancear os usos de práticas preconizadas por processos ágeis e planejados dando origem a abordagens híbridas, para situações nas quais a agilidade pura ou o planejamento puro não funcionariam bem (BOEHM, 2002; LYCETT, 2003).

A abordagem *Situational Method Engineering* (SME) (HENDERSON-SELLERS, 2010) propõe a construção de um método[1] de desenvolvimento específico para cara projeto, de acordo com as características situacionais do mesmo, ou seja, para situações específicas. Esta construção acontece a partir de fragmentos, "pedaços" de métodos, armazenados em um repositório que serve como base de conhecimento para a organização.

Algumas questões sobre SME tratam sobre como definir e armazenar os fragmentos e como recuperar os melhores em relação ao contexto do projeto. Este trabalho propõe uma abordagem sistemática para construção de processos de software específicos para cada projeto através da adaptação de processos, com base em conceitos da abordagem SME, chamada *Octopus SME – Process Tailoring Approach* (OSPTA). São utilizados fragmentos de métodos que incluem práticas preconizadas por processos ágeis e planejados. A abordagem proposta utiliza uma técnica multicritérios para seleção e priorização dos fragmentos com objetivo de guiar os engenheiros de processo na escolha dos melhores elementos para serem incluídos no processo adaptado.

A técnica multicritérios proposta para seleção e priorização de fragmentos é baseada na técnica *Analytic Hierarchy Process* (AHP) (SAATY, 1980), um modelo matemático para suporte à tomada de decisão com base em multicritérios. São levados em conta um ou mais critérios de adaptação, que são os requisitos para o processo adaptado, definidos pelo responsável, mais os oito fatores de contextualização de projetos de software propostos pelo *Octopus Model* (KRUCHTEN, 2010). Tais critérios de adaptação podem ser, por exemplo: prevenção a riscos, satisfação de requisitos de qualidade ou de segurança, entre outros.

[1] Com base em Henderson-Sellers (2010), métodos e processos de desenvolvimento de software são tratados como sinônimos usados variavelmente ao longo do texto.

O *Octopus Model* foi proposto por Philippe Kruchten (KRUCHTEN, 2010). Este modelo propõe oito fatores que afetam significantemente o desenvolvimento de software, podendo ser usados para contextualizar um projeto de software.

Nesta dissertação, é proposto um metamodelo para a definição de fragmentos de métodos para serem utilizados na abordagem OSPTA, o qual pode ser adaptado de acordo com as necessidades da organização. Tal metamodelo é flexível o suficiente para definir fragmentos tanto de natureza ágil, quanto planejada. Ele foi criado a partir da integração dos metamodelos do RUP e da ISO/IEC 24744.

Uma ferramenta de apoio à abordagem proposta foi desenvolvida, com objetivo de validar a mesma. Para estudos de caso foi adaptada uma base de conhecimento proposta por Fontoura (2006) e foram propostos e armazenados outros fragmentos, porém, são apenas sugestões e podem ser adaptadas conforme as necessidades da organização e de acordo com o entendimento de cada engenheiro de processo, e devem evoluir à medida que a organização aprenda com projetos passados.

Nos estudos de casos, a abordagem é aplicada em um contexto de prevenção de riscos em projetos de software, assim, o critério de adaptação será “Riscos do Projeto”. Em outras palavras, o requisito para o processo adaptado é que este inclua práticas para prevenir os riscos identificados para o projeto. Dessa forma, a intenção é a recuperação de fragmentos adequados às características situacionais do projeto, ou seja, que contenham práticas que levem a prevenção dos riscos do projeto e que sejam adequados ao contexto do mesmo (definido de acordo com o *Octopus Model*), para formar um processo adaptado a partir do PSPO.

1.1 Definição do Problema

Profissionais, tanto da área acadêmica quanto da indústria, têm investido esforços para adequar processos de desenvolvimento de software ao contexto dos projetos e/ou das organizações.

Processos de software e modelos de referência, como RUP (RATIONAL, 2003) e ISO/IEC 12207 (ISO/IEC, 2008), fornecem guias para o desenvolvimento de software. No entanto, cada projeto possui suas particularidades.

Não há um processo de software único, que atenda as necessidades de todos os projetos, que seja apropriado em relação aos contextos nos quais os projetos estão inseridos (ALEGRÍA, 2011). Assim, o contexto do projeto deve definir estratégias específicas de desenvolvimento (LAPLANTE, 2004).

A adaptação de processos é uma atividade de grande importância, especialmente para organizações que desejam estar em conformidade com padrões como ISO ou CMMI. Entretanto, é executada frequentemente sem o uso de guias ou de um método sistemático (RUI, 2009).

A abordagem SME trata essas questões ao propor a construção de métodos específicos com foco nas características específicas do projeto, ou seja, para uma determinada situação.

Neste sentido, o problema definido para esta pesquisa é:

Como construir, de maneira sistemática, processos de software específicos para cada projeto de acordo com suas características?

1.2 Escopo da pesquisa

O escopo desta pesquisa visa o entendimento sobre a relação entre as características de um projeto e os processos de desenvolvimento de software, e a obtenção de uma abordagem para construção de processos específicos para um projeto, suficientemente genérica para que possa ser usada de acordo com a necessidade da organização. Por exemplo, através de um processo específico para um projeto, pode-se prevenir riscos em projetos de software, garantir a satisfação de requisitos de segurança, garantir que sejam atendidas metas de qualidade, etc. Além disso, tal abordagem deve considerar o contexto do projeto na elaboração de um processo específico de software.

Esta dissertação apresenta como principal contribuição a proposta de uma abordagem sistemática para a construção de um processo de software específico para um projeto de acordo com as suas características, através de uma análise multicritérios para seleção e priorização de fragmentos de métodos armazenados em um repositório, os quais devem ser incluídos no PSPO para formar o processo adaptado.

Outras contribuições do trabalho incluem:

- Um metamodelo para definição de fragmentos de métodos, o qual permite a instanciação de fragmentos de métodos de natureza ágil e planejada, bem como representa os conceitos e relacionamentos dos elementos de processo relacionados aos fragmentos;
- Associação de fragmentos de métodos com diferentes critérios de adaptação e contextualização dos mesmos através do *Octopus Model;*
- Utilização do *Octopus Model* para caracterização de situações específicas em um projeto de software, para uso na abordagem de adaptação de processos baseada em SME;
- Uma base de conhecimento inicial, como sugestão, com fragmentos de métodos associados aos fatores do *Octopus Model,* indicando o contexto ou situação adequada para que tais fragmentos sejam utilizados;
- Um mecanismo multicritérios para seleção e priorização de fragmentos de métodos, utilizando o modelo matemático AHP, para priorização dos fragmentos de acordo com o contexto do projeto.

1.3 Estrutura da dissertação

O texto está organizado como segue.

No Capítulo 2 são apresentados os referenciais teóricos relacionados a processos de software, abordagens ágeis e planejadas, adaptação de processos, e *Octopus Model*.

O Capítulo 3 trata sobre *Situational Method Engineering* (SME).

O Capítulo 4 descreve trabalhos relacionados a esta dissertação.

No Capítulo 5 são descritos os fragmentos de métodos de acordo com um metamodelo proposto.

No Capitulo 6 é apresentada a abordagem proposta, incluindo a técnica para seleção e priorização de fragmentos de métodos.

No Capítulo 0 é descrita a ferramenta de apoio desenvolvida para suportar a abordagem.

No Capítulo 8 são descritos casos de uso para validação da abordagem proposta.

O Capítulo 9 apresenta as considerações finais e os trabalhos futuros a esta dissertação.

2. PROCESSOS DE SOFTWARE

Softwares são produtos complexos, difíceis de serem desenvolvidos e testados, podendo, quando apresentar comportamento indesejado ou inesperado, causar vários problemas e danos como a perda substancial de dinheiro ou até mesmo colocar vidas em riscos (FUGGETA, 2000).

Segundo Humphrey (1990), processo de software é o conjunto completo e ordenado de atividades de engenharia de software necessárias para transformar os requisitos do usuário em software. Fuggeta (2000) define processos de software como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos, e artefatos que são necessários para conceber, desenvolver, implantar, e manter um produto de software.

O processo de software fornece estabilidade, controle e organização para o desenvolvimento de software, evitando que este se torne caótico (PRESSMAN, 2006).

É amplamente aceito que um processo de software bem definido contribui com a qualidade e produtividade em projetos de desenvolvimento de software, bem como o aumento da qualidade do produto de software (FUGGETA, 2000).

Um processo de software bem definido possibilita que os desenvolvedores utilizem uma linguagem comum para se comunicarem, elaborem documentos e usem técnicas similares na resolução de problemas, adotem estilos de projeto e codificação que sejam facilmente entendidos, e facilita o gerenciamento dos projetos (HULL, 2002).

De acordo com Xu (2005), outros benefícios da definição de um processo de software padrão são:

- Melhoria do desempenho, previsibilidade e segurança dos processos;
- Melhoria da comunicação entre a equipe;
- Desenvolvimento de software escalável, transferível e mensurável;
- Controle do desenvolvimento de software e garantia da consistência das saídas;
- Auxílio no treinamento de novas pessoas no projeto.

Processos de software são descritos por meio de modelos de processo, uma representação abstrata do processo. Um modelo processo de software descreve práticas ou atividades que devem ser executadas para o desenvolvimento de software, a ordem de execução, a relação entre elas e as características de cada atividade como artefatos de entrada e saída, papéis e *stakeholders* envolvidos (XU, 2008). Dessa forma, um modelo de processo é um meio de representar, entender, melhorar e comunicar o processo de software.

Em razão da importância da definição de processos de software, existem muitos modelos e métodos já propostos, tais como: RUP (RATIONAL, 2003), XP (BECK, 2004), SCRUM (SCHWABER, 2002), entre outros. Ainda assim, é bem difundida, em modelos de avaliação de processos, como ISO/IEC 12207 (ISO/IEC, 2008) e CMMI-DEV (SEI, 2010), a idéia de que um processo de desenvolvimento padrão deva ser adaptado para atender necessidades específicas de cada projeto e/ou organização.

2.1 Processos ágeis e planejados

Existem duas principais abordagens para elaboração de processos de desenvolvimento de software, que são: ágeis e planejadas.

Os processos planejados, também conhecidos com tradicionais, impõem disciplina no desenvolvimento de software através de um planejamento extensivo das atividades e processos de codificação e um rigoroso reuso para tornar o desenvolvimento mais eficiente e previsível (BOEHM, 2002). Abordagens tradicionais envolvem um processo detalhado com forte ênfase em planejamento e inspirado em outras disciplinas de engenharia (FOWLER, 2005).

Embora sejam usados há anos, esses processos de desenvolvimento ainda apresentam problemas como estimativas imprecisas de prazos e custos, baixa qualidade dos produtos, insatisfação dos clientes, entre outros (FONTOURA, 2006).

Exemplos de processos de software planejados são: RUP (RATIONAL, 2003), OPEN (HENDERSON-SELLERS, 2000), IBM *Global Services Method* (NICHOLS, 2005), e processos elaborados para atender normas ou modelos tais como o CMMI (SEI, 2010) e ISO/IEC12207 (ISO/IEC 2008).

Métodos ágeis, propostos através do manifesto ágil (http://www.agileAliance.org) surgiram em resposta ao peso da burocracia, à desumanização das abordagens planejadas com foco no planejamento detalhado, e às rápidas mudanças no ambiente de tecnologia de informação (BOEHM, 2002). Agilistas argumentam que as abordagens planejadas são pesadas para lidar com as rápidas e frequentes mudanças de um ambiente dinâmico. Segundo eles, a abordagem ágil atinge maior flexibilidade, minimiza esforços desnecessários e satisfaz melhor as necessidades dos clientes (KAMEL, 2010).

As abordagens ágeis visam encontrar o equilíbrio, provendo apenas o processo suficiente para obter retorno, focam na comunicação entre os membros da equipe e com o cliente e são menos centradas em documentação, assumindo que parte da documentação é o próprio código-fonte. Outras características marcantes são que estas não funcionam bem com equipes grandes, a característica adaptativa em relação às mudanças e a orientação a pessoas (FOWLER, 2005).

Conboy (2009) destaca alguns dos métodos ágeis mais populares: *eXtreme Programming* (XP) (BECK 2004), *Dynamic Systems Development Method* (DSDM) (STAPLETON, 1997), *Scrum* (SCHWABER, 2002), *Crystal* (COCKBURN, 2001), *Agile Modeling* (AMBLER, 2002), *Feature Driven Development (FDD)* (COAD et al., 1999), e *Lean Software Development* (LSD) (POPPENDIECK, 2001).

Segundo Boehm (2002) há projetos para os quais cada uma das abordagens, ágeis ou planejadas, funciona muito bem, podendo ser uma muito melhor em relação à outra. Porém, se esse não for o caso, uma abordagem híbrida é preferível.

Em outras palavras, quando agilidade pura ou planejamento puro não satisfazem as necessidades do projeto, uma mistura é desejável. Assim, práticas preconizadas por abordagens ágeis e planejadas podem ser mescladas, dando origem a abordagens híbridas de acordo com o contexto de cada projeto.

2.2 Adaptação de Processos

Embora existam vários modelos de processos propostos, é bem difundida a idéia de que um processo de desenvolvimento padrão deva ser adaptado para atender necessidades específicas de cada projeto.

Organizações de desenvolvimento de software estão envolvidas em um ambiente dinâmico, onde cada projeto possui suas particularidades. Segundo Alegría (2001), a adequação de um processo de software depende das características do projeto, da organização e do produto, e estas evoluem continuamente. Cada projeto possui características próprias e requer técnicas e estratégias de desenvolvimento particulares.

Adaptação de processos está relacionada à customização de um processo de desenvolvimento de software padrão para atender necessidades da organização e/ou do projeto. Pode ser definida como o ato de ajustar uma definição de processo, particularizando os termos de uma descrição geral para derivar uma descrição aplicável a um ambiente alternativo (menos geral) (GINSBERG, 1995). As atividades de adaptação envolvem adicionar, excluir e/ou modificar elementos de um processo padrão, como por exemplo, papéis, atividades, artefatos, entre outros. (GINSBERG, 1995; XU, 2005).

Através da adaptação de processos é possível balancear o uso de práticas ágeis e planejadas, dando origem a processos híbridos (BOEHM, 2003; LYCETT, 2003).

O SEI, através do CMMI (SEI, 2010), propõe que seja definido um processo de software padrão da organização (PSPO), com elementos de processo fundamentais em todos os projetos da organização, e esse processo seja adaptado para cada projeto de acordo com suas características e restrições, dando origem ao processo definido de software, específico para o projeto.

A definição do processo padrão da organização pode ser considerada como generalização, a partir da qual diferentes processos são especializados. Essa definição generalizada pode estar em conformidade com alguma norma ou modelo, e pode ser avaliada e melhorada constantemente. Como os demais processos são gerados a partir dessa definição, os benefícios e melhorias no processo padrão serão refletidos nos processos dos projetos (FONTOURA, 2006).

A adaptação de processos a partir de um PSPO pode prevenir riscos, reduzir o retrabalho e assegurar a qualidade dos produtos finais Além disso, é importante para organizações que desejam estar em conformidade com padrões como ISO e CMMI (DAÍ, 2007).

Entretanto, a adaptação de processos não é uma tarefa simples devido a fatores como a complexidade do desenvolvimento de software, a variedade de modelos existentes (MAGDALENO, 2010), por ser uma tarefa fortemente baseada em

conhecimento (XU, 2005), e pela dificuldade de entendimento sobre como as características do projeto afetam o processo de desenvolvimento (XU, 2008).

Segundo Pedreira (2007), a adaptação de processos realizada de maneira inadequada pode trazer consequências para a organização, tais como problemas com orçamento do projeto, o tempo de desenvolvimento e a qualidade do produto, desenvolvimento de atividades desnecessárias, entre outros problemas.

A adaptação de processos é um desafio porque é difícil compreender como as características de um projeto afetam as estratégias de adaptação (XU, 2008). É uma atividade que requer conhecimento, experiência e especialidade dos gerentes de projeto (FALBO, 2004). Além disso, abordagens *ad-hoc*, usadas tipicamente, dificultam a repetibilidade e são propensas a erros (ALEGRÍA, 2011).

De acordo com Xu (2008), pesquisadores não dão muita atenção para questões sobre como diferentes fatores moldam um processo de software, e ainda, há falta de estratégias sistemáticas e detalhadas para adaptação de processos e para entendimento desta atividade. Essa ausência acaba levando ao uso de estratégias *ad-hoc,* inapropriadas.

2.3 Contexto do Projeto – Octopus Model

Os melhores processos, mais adequados, dependem das particularidades de cada projeto. Assim, o contexto de cada projeto deve guiar as estratégias de adaptação de processos e de desenvolvimento (LAPLANTE, 2004).

Kruchten (2010) destaca a importância de contextualizar os projetos de software, pois considera o contexto do projeto como fator crítico para o sucesso de um processo de desenvolvimento de software. Assim, Philippe Kruchten propôs o *Octopus Model* para contextualizar projetos de software.

O *Octopus Model* (KRUCHTEN, 2010) propõe oito fatores que afetam significantemente o desenvolvimento de software, e são usados neste trabalho para contextualizar projetos de software, apresentados na Figura 1.

Figura 1 - *Octopus Model* (adaptado de KRUCHTEN, 2010)

Os Fatores do *Octopus Model* são:

- **Tamanho (*Size*):** o tamanho do sistema está relacionado ao tamanho de equipe, ou tempo de desenvolvimento, ou orçamento, ou tamanho do código;
- **Arquitetura Estável (*Stable Architecture*):** os projetos são novos o suficiente para exigir um grande esforço de arquitetura (*middleware*, linguagem de programação, etc.), ou seguem padrões comumente aceitos em seus respectivos domínios;
- **Modelo de negócio (*Business Model*):** considera se o desenvolvimento é de um sistema interno (*in-house*), personalizado para um cliente, ou componente de um sistema grande;
- **Distribuição da equipe (*Team Distribution*):** considera alocação da equipe que, que pode ser em um mesmo local ou distribuída geograficamente. Pode estar relacionado também ao número de equipes;
- **Taxa de mudança (*Rate of Change*):** é referente à estabilidade do ambiente, dos requisitos do sistema;

- **Idade do sistema (*Age of System*):** diz respeito a ser um sistema legado em evolução, um sistema em manutenção ou um novo sistema em desenvolvimento;
- **Criticidade (*Criticity*):** diz respeito às possíveis perdas em caso de falhas do sistema;
- **Controle (*Governance)*:** relacionado à como o projeto de software é gerenciado.

Este modelo fornece guias sobre a adoção de práticas ágeis em processos de software. Kruchten (2010) define o *agile sweet spot*, que identifica as condições ideais nas quais abordagens ágeis tem maior probabilidade de sucesso.

Na da Figura 2 é pode ser visualizado o *agile sweet spot* definido por Kruchten, sendo que os valores em azul representam tal conceito.

Figura 2 - *Agile sweet spot* (adaptado KRUCHTEN, 2010)

Nesta dissertação é considerado que para situações nas quais práticas ágeis não são recomendadas, práticas preconizadas por abordagens planejadas são indicadas. A partir destes oito fatores é possível definir contextos ágeis ou planejados para cada projeto.

Este modelo foi escolhido por ser atual e por facilitar a definição de contextos ágeis e planejados de acordo com as necessidades da abordagem proposta nesta dissertação.

3. SITUATIONAL METHOD ENGINEERING (SME)

Situational Method Engineering (SME) (HENDERSON-SELLERS, 2010) propõe a construção de métodos de desenvolvimento de software específicos para cada projeto de acordo com características situacionais do mesmo, situações específicas, a partir de fragmentos de métodos previamente armazenados em um repositório, chamado de base de métodos.

Encontram-se divergências na literatura sobre o significado exato do termo "*situational*". Segundo Bucher (2007), o termo "situação" (*situational*) se refere a contingências do projeto que devem ser levadas em conta para construção do método de desenvolvimento. Aharoni (2008) define situação como diferentes características relacionadas à organização, ao projeto, à equipe de desenvolvimento, ao cliente, entre outros fatores.

Neste trabalho a situação específica de um projeto é caracterizada por um ou mais requisitos do método específico, os quais são chamados de critérios de adaptação, e pelo contexto do projeto definido a partir do *Octopus Model* (KRUCHTEN, 2010).

Em SME, a construção dos métodos/processos específicos se dá partir de elementos de construção reusáveis chamados fragmentos de métodos (PUVIANI, 2009). Agerfalk (2007) define fragmentos de métodos como uma parte coerente de um método através dos quais é possível construir um método situacional.

Estes elementos ou blocos de construção são tipicamente extraídos, por exemplo, de boas práticas, de outros processos, de padrões de processo, ou modelos de referência. Uma vez identificados e documentados, são armazenados em uma base de métodos, formando uma base de conhecimento da organização.

Existem duas principais definições para os elementos de construção, fragmentos de métodos em SME, as quais serão discutidas na Seção 3.1. São elas: *method fragments* e *method chunks* (HENDERSON-SELLERS, 2008).

Em SME, os elementos de construção são recuperados e adaptados de acordo com uma situação específica (AHARONI, 2008). Ao recuperar fragmentos para a construção de um método específico, é necessário levar em conta as características

situacionais, que auxiliam na determinação de quais elementos são mais apropriados (HENDERSON-SELLERS, 2008). A identificação de cada situação para a seleção dos fragmentos mais adequados é um aspecto crucial em SME (BÖRNER, 2011).

Henderson-Sellers (2010) defende a hipótese de que SME é uma potencial solução para o problema de selecionar a melhor estratégia para o desenvolvimento de determinado projeto. Segundo Puviani (2009), esta abordagem é bem conhecida no campo da engenharia de software, embora não muito difundida devido à sua complexidade. Deneckere (2008) resume SME em três atividades:

- Definir fragmentos, que são armazenados em um repositório;
- Recuperar os elementos mais adequados de acordo com as especificações do projeto;
- Construir um método a partir dos componentes selecionados.

Ralyté (2006) coloca como primeiro passo para abordagens baseadas em SME a definição da situação. Resumidamente, SME gira em torno da identificação dos fragmentos adequados em relação à situação do projeto e sua ligação de maneira apropriada para a elaboração de métodos de desenvolvimento (HENDERSON-SELLERS, 2008).

A Figura 3 apresenta uma visão geral da abordagem SME, onde os fragmentos de métodos são selecionados a partir da base de métodos, de acordo com as características situacionais do projeto, para construção de um método específico.

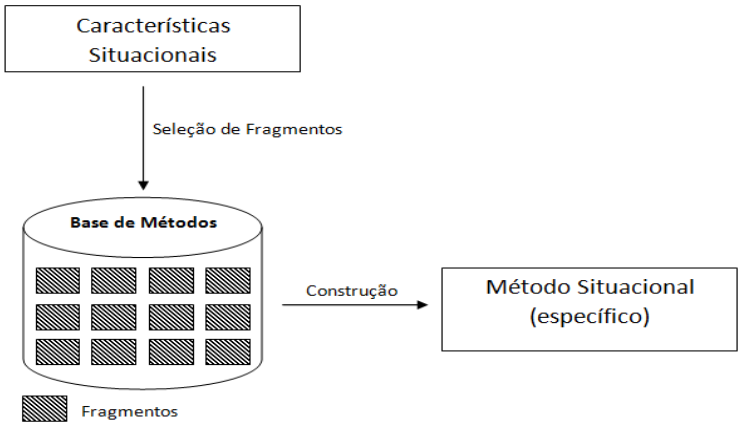


Figura 3 - Visão geral de SME (adaptado de HENDERSON-SELLERS, 2010)

3.1 Elementos de construção em SME

Existem duas principais definições propostas para fragmentos a partir dos quais um processo situacional pode ser construído, que são: *method chunks* e *method fragments* (HENDERSON-SELERS, 2008).

Segundo Henderson-Sellers (2008), é amplamente aceito que um *method fragment*[2] é um elemento gerado a partir de um metamodelo por instanciação. Há uma distinção entre *process-fragments* (por exemplo, uma atividade), e *product-fragments* (por exemplo, um artefato), e estes são definidos separadamente, de forma independente.

Em abordagens SME que utilizam o conceito de *method fragments,* deve haver algum tipo de associação entre *process-fragments* e *product-fragments* para capturar as dependências apropriadas, e informações situacionais estão distribuídas em classes do metamodelo (HENDERSON-SELLERS, 2008).

Um *method chunk* é uma combinação de um *process-fragment* um *product-fragment*, integrando os dois aspectos em um elemento, sendo que a parte produto contém os elementos de produto necessários para execução da parte processo. Há um link predeterminado entre as partes *process-fragment* e *product-fragment*, com uma relação 1...1 (um para um) entre estes. Além das partes processo e produto, um *chunk* possui uma interface e um descritor com informações para identificar o elemento e sobre a situação de uso (HENDERSON-SELLERS, 2008).

[2] O termo "fragmentos de método" é usado nesta dissertação referindo-se a *process-fragments*, os quais são focados neste trabalho. Para se referir a *product-fragments*, o termo específico será utilizado.

4. Trabalhos Relacionados

Esforços de pesquisa sobre *Situational Method Engineering*, a fim de prover mecanismos para a construção de métodos/processos de desenvolvimento de acordo com as características de cada projeto, são encontrados na literatura. Alguns destes, relacionados a esta dissertação, são comentados a seguir.

Sonia (2007.) propõe uma abordagem para a construção de métodos situacionais para desenvolvimento de sistemas web, a partir de componentes reusáveis originados de outros métodos que suportam o desenvolvimento web. Entretanto, não há esclarecimentos sobre os componentes armazenados no repositório, como eles são modelados e tratados. Nesta abordagem, os componentes são selecionados, com apoio da técnica AHP, de acordo com características de modelos de produto correspondentes a aspectos de sistemas web (por exemplo, modelo de navegação e modelo de apresentação), sendo que estes são definidos de acordo com fatores situacionais relacionados a características do projeto. Estes fatores não são especificados e não há uma discussão sobre como são utilizados. Além disso, não são levados em conta na seleção dos componentes que formarão o método situacional, sendo considerados apenas para definição de quais modelos de produto estarão envolvidos no processo.

Kornyshova (2007) propõe o uso de técnicas multicritérios para a seleção de elementos de construção apropriados em SME, utilizando a técnica AHP. Neste trabalho não há uma definição dos critérios que afetam significantemente o desenvolvimento de software e que servem como critérios de comparação para avaliação das alternativas, dificultando a criação de uma base de conhecimento da organização contendo a contextualização dos fragmentos em relação a critérios de comparação. Além disso, não é abordada a maneira pela qual os fragmentos são avaliados.

Gericke (2009) propõe o uso de *Situational Method Engineering* para uma solução integrada de governança, gerenciamento de riscos e conformidade (GRC) em sistemas de informação. Este trabalho tem foco na definição, apresentação e validação de fragmentos de métodos relacionados a aspectos conceituais, estratégicos, organizacionais, técnicos e culturais de uma solução de GRC. Entretanto, Gericke

(2009) não apresenta mecanismos e/ou critérios pelos quais os fragmentos possam ser recuperados de um repositório para compor um método situacional, como selecionar os mais adequados, e exemplos de aplicação dos fragmentos propostos.

Abad (2010) propõe fragmentos de métodos para serem utilizados em abordagens baseadas em SME. Os elementos propostos neste trabalho são limitados ao contexto ágil e estão relacionados ao conceito de *method chunks* (Seção 3.1), cujas desvantagens são comentadas no Capítulo 5. Além disso, não há exemplos de aplicação para o modelo de fragmento proposto e também não trata da recuperação e seleção dos elementos.

O trabalho apresentado nesta dissertação difere dos demais por propor uma abordagem para adaptação de processos de desenvolvimento de software baseada em SME, desde a definição das características situacionais do projeto, definição de fragmentos de métodos e um mecanismo para seleção dos melhores fragmentos para serem inseridos no processo específico do projeto, de acordo com os requisitos para o processo adaptado e com o contexto do projeto. Ainda, este trabalho apresenta um detalhamento da abordagem proposta através de estudos de caso.

A Tabela 1 – Comparação da abordagem OSPTA com trabalhos relacionados. apresenta uma comparação da abordagem proposta nesta dissertação com os trabalhos relacionados.

Para tal comparação foram avaliados os seguintes critérios:

- ***Definição de Fragmentos –*** se apresenta uma definição para os fragmentos utilizados.
- ***Aplicação dos Fragmentos –*** se apresenta uma ou mais aplicações com utilizando os fragmentos definidos.
- ***Definição de Fatores Situacionais –*** se define os fatores situacionais a serem levados em conta, com base nos conceitos de SME.
- ***Técnicas de Priorização –*** se apresenta uma técnica de priorização para identificar os fragmentos mais utilizados em relação à situação específica do projeto.
- ***Ferramenta de Apoio –*** se apresenta uma ferramenta de apoio à abordagem que propõe.

A abordagem OSPTA, proposta nesta dissertação, satisfaz todos os critérios citados acima.

Tabela 1 – Comparação da abordagem OSPTA com trabalhos relacionados.

	Definição de Fragmentos	**Aplicação dos Fragmentos**	**Definição de Fatores Situacionais**	**Técnicas de Priorização**	**Ferramenta de Apoio**
Sonya (2007)	NÃO	NÃO	NÃO	SIM	NÃO
Kornyshova (2007)	NÃO	NÃO	NÃO	SIM	NÃO
Gerike (2009)	SIM	NÃO	NÃO	NÃO	NÃO
Abad (2010)	SIM	NÃO	NÃO	NÃO	PARCIAL
OSPTA	SIM	SIM	SIM	SIM	SIM

5. Fragmentos de métodos para a abordagem OSPTA

O modelo de fragmentos de métodos utilizados na abordagem aqui proposta é baseado no conceito de *method fragments* descrito na Seção 3.1.

Os *method fragments* permitem relacionamentos n...n (muitos para muitos) entre os *process-fragments* e *product-fragments*, mantendo a individualidade dos mesmos no repositório (base de métodos). Esta característica torna este modelo mais adequado para situações reais, onde, por exemplo, algumas tarefas podem ter múltiplos artefatos envolvidos. Além disso, favorece a flexibilidade proposta pela abordagem SME (HENDERSON-SELLERS, 2008).

Esta flexibilidade obtida pelo uso de *method fragments* evita redundância na base de métodos, que seria causada pelo uso de fragmentos baseados no conceito de *chunks*. Em casos de situações em que relações 1...1 (um para um) não sejam adequadas, um *chunk* separado deve ser criado para cada configuração específica. Por exemplo, deve ser criado um *chunk* com uma parte produto e uma parte processo, e outro com a mesma parte processo e uma parte produto diferente do primeiro, para representar mais de um artefato relacionado a uma mesma atividade.

Além disso, os *chunks* não incluem um elemento para representar as pessoas envolvidas no desenvolvimento de software, e as ferramentas que usam.

Henderson-Sellers (2008) afirma que abordagens baseadas em *method fragments* têm sido usadas com sucesso em projetos reais.

Na Seção 5.1 é apresentado o metamodelo M^2F (*Metamodel for Method Fragments*), proposto nesta dissertação, por meio do qual são definidos os fragmentos de método a serem utilizados na Abordagem OSPTA.

5.1 M^2F – *Metamodel for method fragments*

Os fragmentos utilizados na abordagem OSPTA são definidos a partir do metamodelo M^2F (*Metamodel for Method Fragments*), publicado em Pereira (2012b).

Este metamodelo permite a instanciação de fragmentos de métodos e foi desenvolvido a partir da integração do metamodelo do RUP (BENCOMO, 2005), descrito na Seção 5.1.1, com o metamodelo proposto na ISO/IEC 24744 (ISO/IEC 2007), descrita na Seção 5.1.2.

Na Seção 5.1.3 é descrito o metamodelo M^2F.

5.1.1 Metamodelo RUP

Rational Unified Process (RUP) é um *framework* de processo para desenvolvimento de software iterativo e incremental que provê um enfoque disciplinado para distribuição de tarefas e responsabilidades em uma organização de desenvolvimento de sistemas (SHUJA, 2007).

O RUP está organizado em fases, cada qual composta por uma ou mais iterações. Disciplinas são definidas como uma categorização de atividades baseada em similaridade. Dependendo da fase em que o projeto está, cada iteração é formada por um conjunto de disciplinas, com ênfase variada. Cada fase tem objetivos bem definidos que são verificados ao final da fase (IBM, 2007).

A Figura 4 mostra o metamodelo que descreve os elementos utilizados na representação do RUP (BENCOMO, 2005). A seguir uma breve descrição dos elementos descritos no metamodelo, segundo (BENCOMO, 2005):

- ***Lifecycle*:** do ponto de vista gerencial o ciclo de vida é composto por fases (*phases*) concluídas por um marco do projeto, e do ponto de vista técnico o ciclo de vida consiste de várias disciplinas (*disciplines*).
- ***Phases*:** são associadas a *Workflows Detail*. Diz respeito a intervalos de tempo entre marcos do projeto.

- ***Discipline*:** identifica um conjunto de trabalhadores (*workers*) que participam na disciplina (*discipline*) e agrupa um conjunto de *workflows detail* relacionados.

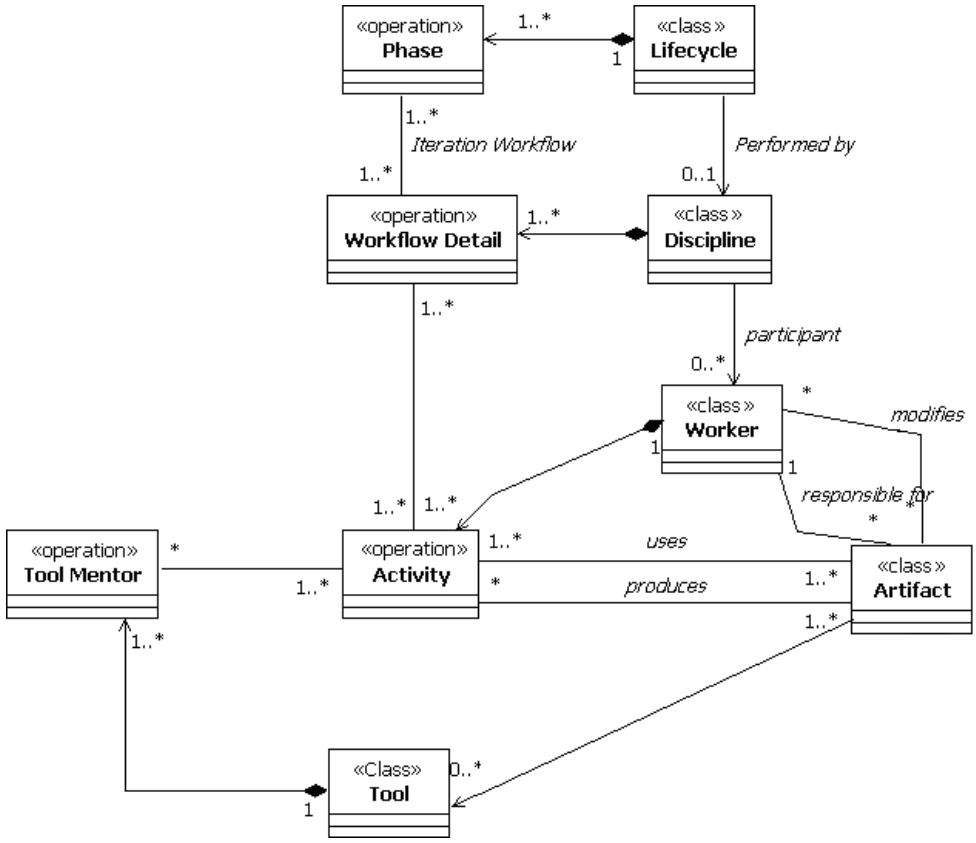


Figura 4 - Metamodelo do RUP (BENCOMO, 2005)

- ***Workflow Detail*:** especifica uma colaboração específica dentro de uma disciplina (*discipline*).
- ***Activity*:** representa uma tarefa do processo, que um papel (*worker*) deve executar.
- ***Artifact*:** são produtos gerados durante as atividades de desenvolvimento do software, tais como: modelos, planos, versões do software, relatórios, etc.
- ***Worker*:** representa os cargos executados por indivíduos em um projeto.
- ***Tool*:** representa as ferramentas de desenvolvimento usadas na organização.

A partir do RUP 7 (SHUJA, 2007), *Workflow Detail* passou a se chamar *Activity*, e esta passou a se chamar *Task.* Esta nomenclatura proposta pelo RUP 7 é utilizada nesse trabalho.

5.1.2 ISO/IEC 24744

ISO/IEC 247444 (ISO/IEC, 2007) é uma norma internacional que define um metamodelo para elaboração de metodologias para o desenvolvimento de software, chamado *Software Engineering Metamodel for Development Methodologies* (SEMDM). A norma cobre os seguintes aspectos de um processo de software (GONZALEZ-PEREZ, 2007):

- ***WorkUnit***: se refere ao aspecto de processo. Descreve o trabalho que deve ser realizado para obter o sistema.
- ***WorkProduct***: se refere aos produtos das metodologias. Descreve artefatos que devem ser usados e/ou criados para obter o sistema.
- ***Producers***: se refere às pessoas. Descreve os papéis, *times* que executam *WorkUnits* e utilizam e/ou criam *WorkProducts.*
- ***Stages***: referente ao aspecto temporal.
- ***ModelUnits***: referente à modelagem das metodologias. Trata aspectos sobre como será modelada a metodologia, por exemplo, linguagem de modelagem a ser utilizada.

A ISO/IEC 24744 aplica uma modelagem de duas camadas, uma para representar elementos de um processo ("*MethodologyElement*") e outra para representar o domínio de aplicação ("*EndeavourElement*"). Por exemplo, o metamodelo proposto nesta norma possui as classes "*WorkUnit*"e "*WorkUnitKind*", sendo que a primeira representa uma "*WorkUnit*" com atributos correspondentes a como ela é executada no domínio de aplicação, por exemplo "*StartTime*" e "*EndTime*", enquanto a segunda representa uma "*WorkUnit*" como ela é documentada em uma metodologia (GONZALEZ-PEREZ, 2007).

A Figura 5 apresenta uma visão geral da estrutura do metamodelo proposto pela ISO/IEC 24744.

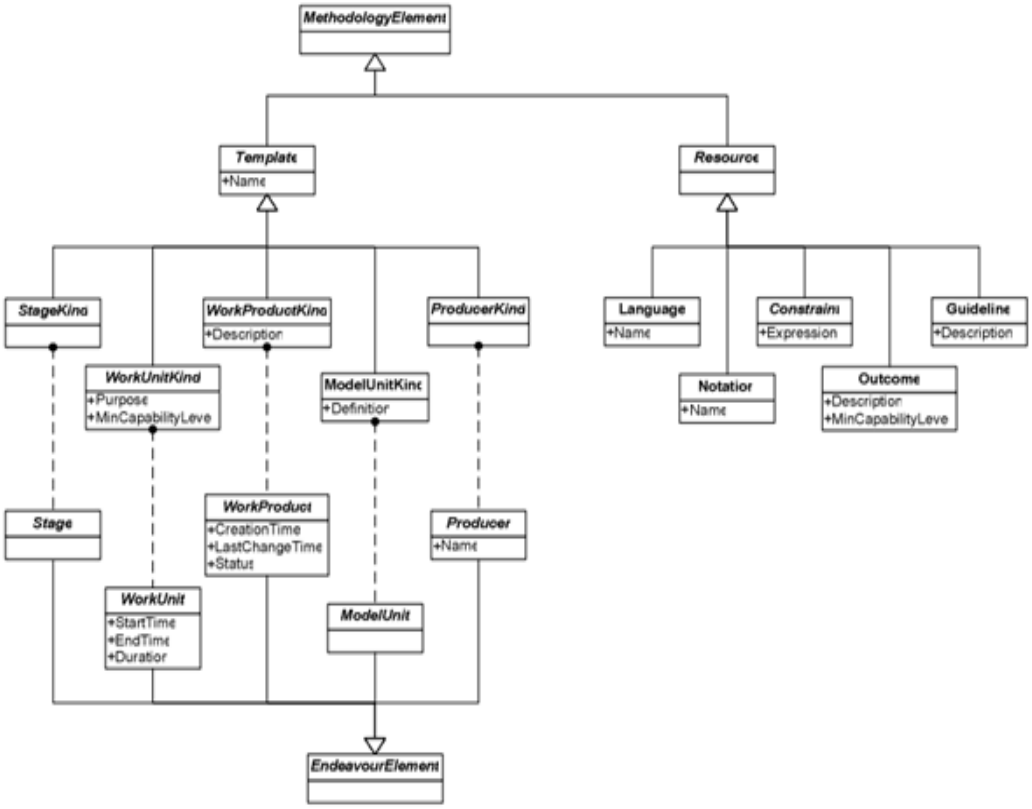


Figura 5 - Visão geral da ISO/IEC 24744 (ISO/IEC 2007)

5.1.3 M^2F – *Metamodel for Method Fragments*

O metamodelo M^2F, publicado em Pereira (2012b), foi desenvolvido a partir da integração do metamodelo do RUP (BENCOMO, 2005) com conceitos da ISO/IEC 24744 (ISO/IEC 2007), permite a instanciação de fragmentos de métodos que sejam utilizados em abordagens baseadas em SME, como a proposta nesta dissertação.

A integração entre estes dois metamodelos foi realizada de acordo com as características desejáveis para os fragmentos de métodos. O RUP escolhido por ser um *framework* de processo bem difundido e compatível com o *Software Process Engineering Metamodel Specification* (SPEM) (OMG, 2008), com elementos bem conhecidos e com características apropriadas ao propósito deste trabalho.

O uso do metamodelo da norma ISO/IEC 24744 se justifica por prover conceitos apropriados para o propósito deste trabalho, como a classe "*Action*" (não presente na Figura 5) para capturar dependências apropriadas entre *process-fragments* e *product-*

fragments como preconiza o conceito de *method fragments*, bem como por prover mecanismos para a recuperação dos fragmentos nas classes do metamodelo (classe "*Guideline*"), usando informações situacionais.

M^2F descreve elementos para a definição de fragmentos de métodos e garante a integridade dos relacionamentos entre os elementos.

O metamodelo M^2F tem objetivo de ser flexível o suficiente para que sejam definidos fragmentos de natureza ágil e planejada, bem como permite a combinação de elementos de diferentes fontes, por exemplo, modelos de processo, modelos de referência e de avaliação de processos e padrões organizacionais.

Outro objetivo do metamodelo M^2F é ser o mais simples possível, representando apenas os conceitos necessários para a definição dos fragmentos de métodos. Por esta razão, nem todas as classes e atributos do metamodelo do RUP e do metamodelo proposto na ISO/IEC 24744 são relevantes para o propósito deste trabalho. A norma ISO/IEC 24744 propõe um metamodelo complexo, pois concebe a modelagem em duas camadas, como citado na Seção 5.1.2, e extenso.

A modelagem de duas camadas não é necessária para o propósito deste trabalho, pois o escopo deste é relacionado apenas com a definição dos fragmentos de métodos, não levando em conta a execução dos mesmos (domínio de aplicação).

A Figura 6 ilustra o metamodelo M^2F, com algumas evoluções em relação ao publicado em Pereira (2012b). Os fragmentos de métodos são representados como instâncias das classes do metamodelo M^2F.

No metamodelo M^2F, a classe "*Activity*" corresponde a um *process-fragment* enquanto a classe "*Artifact*" corresponde a um *product-fragment.*

Os elementos do metamodelo M^2F foram escolhidos de acordo com as características desejáveis para os fragmentos de métodos, em relação à abordagem proposta nesta dissertação.

A integração conceitual é o processo de integrar esquemas conceituais em um esquema global (BATINI, 1992). Este trabalho segue regras propostas por Batini (1992) para desenvolver um metamodelo capaz de representar elementos necessários para modelagem de fragmentos de métodos. Seguindo tais regras para integração de modelos, foi mantida a nomenclatura do RUP (versão 7) (SHUJA, 2007) para classes com conceitos correspondentes. Por exemplo, entre *"Worker*" (RUP) e "*Producer*" (ISO/IEC 24744), foi escolhido o nome "*Worker*"; entre "*Artifact*" (RUP) e "*WorkProduct*" (ISO/IEC 24744) o primeiro foi escolhido.

No metamodelo proposto, as classes extraídas da ISO/IEC 24744 não utilizam o sufixo "*Kind*", apesar de serem semanticamente equivalentes as classes "*Kind*" da norma em questão. Isso foi determinado para facilitar o entendimento do metamodelo através de uma nomenclatura similar com já conhecidas, como o SPEM 2.0 (OMG, 2008). Por exemplo, a classe "*Action*" no metamodelo M^{2}F não utiliza o sufixo "*Kind*" utilizado na ISO/IEC 24744, embora seja semanticamente equivalente.

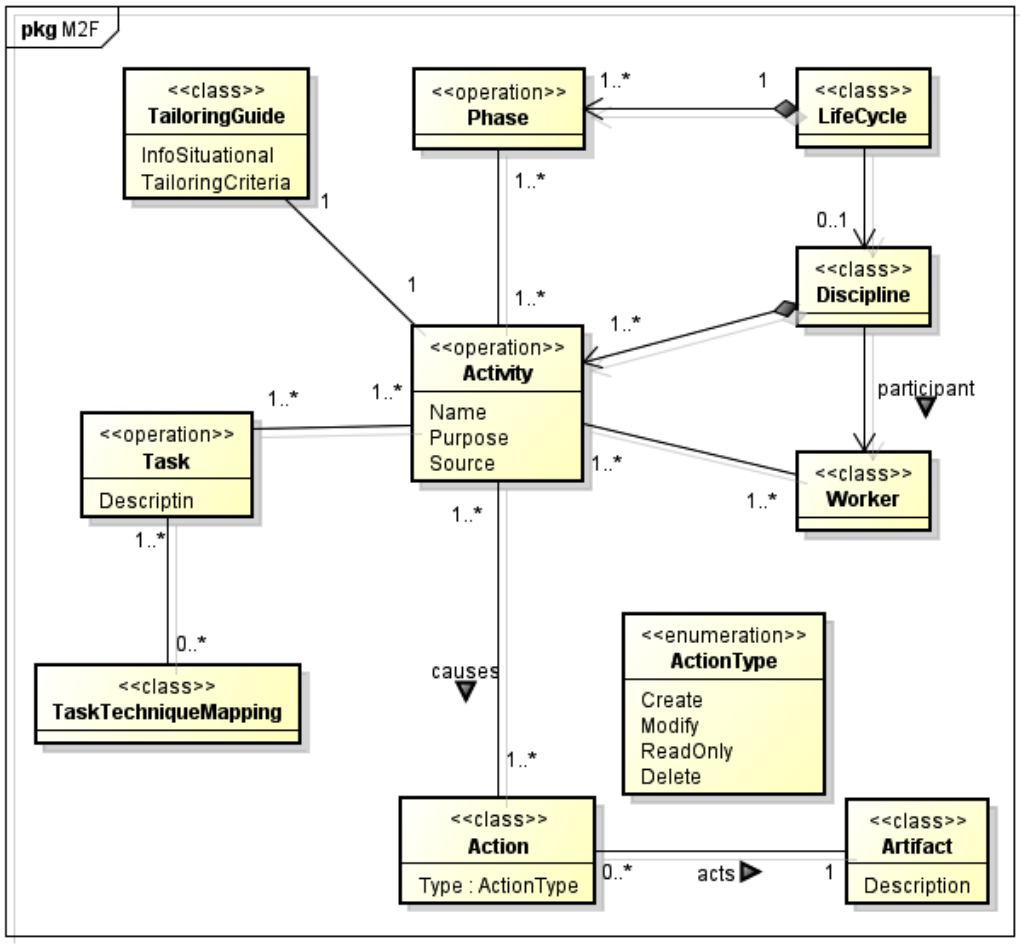


Figura 6 - M^{2}F - Metamodel for method fragments

Entretanto, o metamodelo utiliza conceitos do metamodelo da ISO/IEC 24744, por exemplo, a classe "*Action*" para capturar dependências apropriadas entre *process-fragments* e *product-fragments*, representados pelas classes "*Activity*" e "*Artifact*". Este conceito permite uma relação claramente especificada entre *process-fragments* e *product-fragments*, em vez de tratar a parte produto apenas como entrada/saída, pois

especifica o tipo de relação entre as partes, podendo ser um dos tipos: "*Create*", "*Modify*", "*ReadOnly*" e "*Delete*".

Além disso, o metamodelo provê mecanismos para a recuperação de fragmentos a partir de informações situacionais. A classe "*Guideline*" (ISO/IEC 24744), chamada neste trabalho de "*TailoringGuide*", captura informações sobre como e onde um fragmento de método pode ser usado, uma situação para a qual o uso de tal fragmento é adequado. Neste trabalho, são propostos dois atributos para esta classe, que são: "*InfoSituational*" e "*TailoringCriteria*". O primeiro diz respeito ao contexto de uso dos fragmentos definido de acordo com os fatores do *Octopus Model*, que serve como guia sobre onde o fragmento pode ser empregado com sucesso. O atributo "*TailoringCriteria*" descreve um ou mais critérios de adaptação definido pelo engenheiro de processo ou outro responsável, por exemplo, riscos do projeto que o fragmento pode prevenir através de suas tarefas, requisitos de segurança do projeto que o fragmento pode atender, etc.

As classes "*Discipline*" e "*Lifecycle*" têm como origem o metamodelo do RUP, assim como as classes "*Artifact*", "*Phase*" e "*Worker*" que correspondem as classes "*WorkProductKind*", "*StageKind*" e "*ProducerKind*" da ISO/IEC 24744, respectivamente. A classe "*Activity*" representa a classe "*WorkUnitKind*" da ISO/IEC 24744 e a classe "*WorkflowDetail*" do metamodelo do RUP, porém com nomenclatura atualizada de acordo com o RUP 7. Assim como a classe "*Task*", que representa a classes "*Activity*" do metamodelo do RUP e "*TaskKind*" da ISO/IEC 24744.

O atributo "*Purpose*" tem origem na classe "*WorkUnitKind*" da ISO/IEC 24744 e serve para especificar o propósito do fragmento.

A classe "*TaskTechniqueMapping*", da ISO/IEC 24744, corresponde a guias para a execução das práticas e tarefas relacionadas ao fragmento em questão.

5.2 Fragmentos de métodos definidos a partir do metamodelo M^2F

Nesta seção, são apresentados exemplos de fragmentos de métodos definidos a partir do metamodelo descrito na Seção 5.1.3. Estes são apenas sugestões, pois cada organização pode editar os fragmentos da base de métodos e criar uma base nova de

acordo com suas necessidades. Na Tabela 2[3] pode ser observado um modelo para representação de um fragmento de método, descrevendo cada atributo.

Tabela 2 - Modelo para definição de *process-fragments*

Process-fragment	*Fragment name.*
Purpose	*Here should be the fragment purpose, the fragment goal.*
Source	*Fragment source.*
Discipline	*One discipline related to software development projects.*
Phase(s)	*One or more development phases practiced in the organization.*
Tasks(s)	*Task(s) associated with the fragment.*
Worker(s)	*Roles involved in the fragment implementation.*
TaskTechnique Mapping	*Techniques and guides to execute the fragment tasks.*
Action(s)	*Relations between the process-fragment and product-fragment.*
TailoringGuide	**InfoTailoring:** *Here should be the values for Octopus Model's factors to define the fragment use-context.*
	TailoringCriteria: *Here should be one or more Tailoring Criteria, e.g., risks which the fragment can prevent.*

Na Tabela 3 é possível observar um modelo para representação de um *product-fragment*.

Na Tabela 4 é representado um fragmento exemplo chamado "*Analyze The Problem*" extraído do RUP, enquanto na Tabela 5 é representado o "*Planning Game*" extraído do XP. Na Tabela 6 é mostrado o *product-fragment* exemplo chamado "*Requirements Management Plan*".

Os valores associados aos fatores do *Octopus Model* para definir o contexto de uso do fragmento, em "*InfoTailoring,*" são tratados no Capítulo 6.

[3] Os fragmentos são descritos em inglês devido ao fato de estarem armazenados assim na base de conhecimento da ferramenta usada para a elaboração de estudos de caso e publicação de artigos em inglês.

Tabela 3 - Modelo para definição de *product-fragments*

Product-fragment	*Fragment name.*
Description	*Fragment description.*
Relations	*Relations between the process-fragment and product-fragment.*

Tabela 4 - Fragmento "*Analyze The Problem*"

Fragment	Analyze the Problem	
Purpose	*The purpose is to gain agreement on the problem being solved. Analysis of the problem involves identify the stakeholders…*	
Source	*RUP.*	
Discipline	*Requirements.*	
Phase(s)	*Inception; Elaboration.*	
Tasks	✓ *Capture a Common Vocabulary;* ✓ *Find Actors and Use Cases;* *etc.*	
Roles	✓ *System Analyst ;* ✓ *Customer;* ✓ *Others stakeholders.*	
Task-Technique Mapping	✓ *Brainstorming;* ✓ *Requirements Workshop;* *etc.*	
Action	✓ *Create "Requirements Management Plan";* ✓ *Read "Software Development Plan";* *etc.*	
TailoringGuide	**InfoTailoring:**	***Size:*** *large.* ***Stable Architecture:*** *new.* ***Business Model:*** *large system component.* ***Team Distribution:*** *geographic distribution.* ***Rate of Change:*** *less than 10.* ***Age of System:*** *legacy evolution.* ***Criticity:*** *money loss.* ***Governance:*** *mechanic/formal.*
	TailoringCriteria: *One or more Tailoring Criteria, e.g., risks which the fragment can prevent.*	

Tabela 5 – Fragmento "*Planning Game*"

Process-fragment	*Planning Game*
Purpose	*The purpose of this process-fragment is to determine the scope of the next iteration…*
Source	*Portland Pattern Repository.*
Discipline	*Requirements.*
Phase(s)	*Inception; Elaboration.*
Tasks	✓ *Write User Stories;* ✓ *Define Iteration Scope;* *etc.*
Roles	✓ *Customer;* ✓ *Developer;* ✓ *XP Manager.*
Task-Technique Mapping	✓ *Users present the User Stories and decide the priorities for the next release* *etc.*
Action	✓ *Read "User Stories";* ✓ *Create "Release Plan";*
TailoringGuide	***Infotailoring:*** ***Size:*** *small.* ***Stable Architecture:*** *stable.* ***Business Model:*** *in house or bespoke for a customer.* ***Team Distribution:*** *collocated.* ***Rate of Change:*** *more than 30.* ***System Age:*** *new.* ***Criticity:*** *comfort.* ***Governance:*** *dynamic/flexible.*
	TailoringCriteria: *One or more Tailoring Criteria, e.g., risks which the fragment can prevent.*

A Tabela 4 e a Tabela 5 representam exemplos de fragmentos com contextos de uso extremamente planejado e extremamente ágil, respectivamente.

Tabela 6 - *Product-fragment* "*Requirements Management Plan*"

Product-fragment	*Requirements Management Plan*
Description	*Describes the requirements, specifies information, controls mechanisms for monitoring requirements changes.*
Relations	✓ *Created by "Analyze the Problem".* *etc.*

6. Octopus SME – *Process Tailoring Approach* (OSPTA)

Este trabalho apresenta a abordagem *Octopus SME – Process Tailoring Approach* (OSPTA), publicada em Pereira (2012a). Trata-se de uma abordagem sistemática para construção de processos de software específicos para cada projeto, através da adaptação de processos, baseada em conceitos da abordagem SME. Para tal construção, são utilizados fragmentos de métodos, como definidos e exemplificados no Capítulo 5.

O processo adaptado de software, específico para o projeto, é formado a partir do processo de software padrão da organização, com a inclusão de fragmentos de métodos recuperados da base de métodos de acordo com o critério de adaptação, o qual pode ser entendido como requisitos para o processo adaptado, e priorizados de acordo com sua adequação ao contexto do projeto. Assim, é possível afirmar que a seleção e priorização dos fragmentos se dão através de uma técnica multicritérios, de acordo com a situação ou características situacionais do projeto.

A abordagem OSPTA é dita multicritérios pelo fato de considerar o critério de adaptação definido para recuperar fragmentos do repositório, e utilizar os oito fatores do *Octopus Model* (KRUCHTEN, 2010) como critérios de comparação para priorizar fragmentos de acordo com o contexto do projeto. Tal técnica tem objetivo de guiar os engenheiros de processo na escolha dos melhores fragmentos para serem incluídos no processo adaptado.

Os fragmentos selecionados e priorizados nesta abordagem são *process-fragments*, os quais descrevem tarefas a serem executadas no processo de software. Na construção do processo adaptado, todos os elementos relacionados com cada *process-fragments* são inclusos, inclusive os *product-fragments*.

A seguir é descrito como é tratado o conceito de “contexto” na abordagem OSPTA.

6.1 Contexto na abordagem OSPTA

O contexto do projeto é um aspecto crucial para abordagens baseadas em SME, por isso o nome "*situational*". Na abordagem baseada em SME proposta nesta dissertação, o termo "*situational*", ou características situacionais do projeto, é tratado como sendo o critério de adaptação definido, mais o contexto do projeto caracterizado a partir do *Octopus Model*. Tanto o contexto do projeto quanto o contexto de uso dos fragmentos são caracterizados pelos oito fatores *Octopus Model* (Seção 2.3). A Tabela 7 mostra os fatores do *Octopus Model* e possíveis valores para a definição de contextos, tanto para o projeto quanto para o uso dos fragmentos.

Tabela 7 - Valores para definição de contextos com *Octopus Model*

Fator	**Valores Possíveis**		
	Características ágeis →		Características planejadas →
Tamanho	Pequeno	Médio	Grande
Arquitetura	Estável	Modificada	Nova
Modelo de Negócios	*In house* ou sob medida para um cliente.	Comercial	Componente de um grande sistema.
Distribuição da Equipe	Mesmo local	Equipes diferentes	Distribuição geográfica
Taxa de Mudança (% em um mês)	Mais que 30	Entre 10 e 30	Menos que 10
Idade do Sistema	Novo desenvolvimento	Manutenção	Evolução de sistema legado
Criticidade	Perda de conforto	Perda de dinheiro	Mortes
Controle	Dinâmico/Flexível	Regras simples	Mecânico/Formal

O contexto de uso de um fragmento trata de um contexto no qual é apropriada a utilização de tal fragmento, a execução das tarefas contidas no fragmento.

O *agile sweet spot* (Seção 2.3) de Kruchten (2010), foi adaptado de acordo com as necessidades deste trabalho.

Na Tabela 7 é possível observar, em "Valores Possíveis", que os valores da coluna mais a esquerda representam características para as quais práticas ágeis seriam indicadas, enquanto que os valores da coluna mais a direita representam características para as quais práticas de abordagens planejadas seriam mais indicadas. Há também o meio termo entre os extremos, ágil e planejado. Assim, a partir dos fatores do *Octopus Model* e destes valores, é possível definir contextos de natureza ágil e planejada, seja para contextualizar o projeto ou para definir o contexto de uso dos fragmentos.

6.2 Visão geral da abordagem OSPTA

A abordagem OSPTA, através da adaptação de processos de software, visa à construção de um processo específico para cada projeto, que atenda as características situacionais do mesmo.

Assumindo que já existe um processo padrão da organização definido e que há uma base de conhecimento da organização contendo os elementos necessários, a Figura 7 apresenta uma visão geral da abordagem OSPTA, detalhando as principais atividades que descrevem a abordagem. Há outras atividades relacionadas à abordagem OSPTA, como a definição e modificação de um processo de desenvolvimento padrão para a organização, e a definição de outros elementos de processo como *product-fragments,* papéis, etc.

A seguir são descritas as atividades propostas na abordagem OSPTA.

- ***Definir características situacionais do projeto:*** primeiramente, deve ser definida a situação do projeto, suas características situacionais. Para tal, o engenheiro de processo contextualiza o projeto a partir dos fatores situacionais propostos pelo *Octopus Model*, com valores propostos nesta dissertação (Seção 6.1), e define um ou mais critérios de adaptação ("*TailoringCriteria*") para a construção do PEP. Esta abordagem é genérica, não determinando um critério de adaptação fixo, e assim, possibilitando que

diferentes critérios de adaptação sejam definidos para sua aplicação. Estes podem ser prevenção a riscos, satisfação de requisitos de qualidade ou de segurança, entre outros, de acordo com as necessidades da organização.

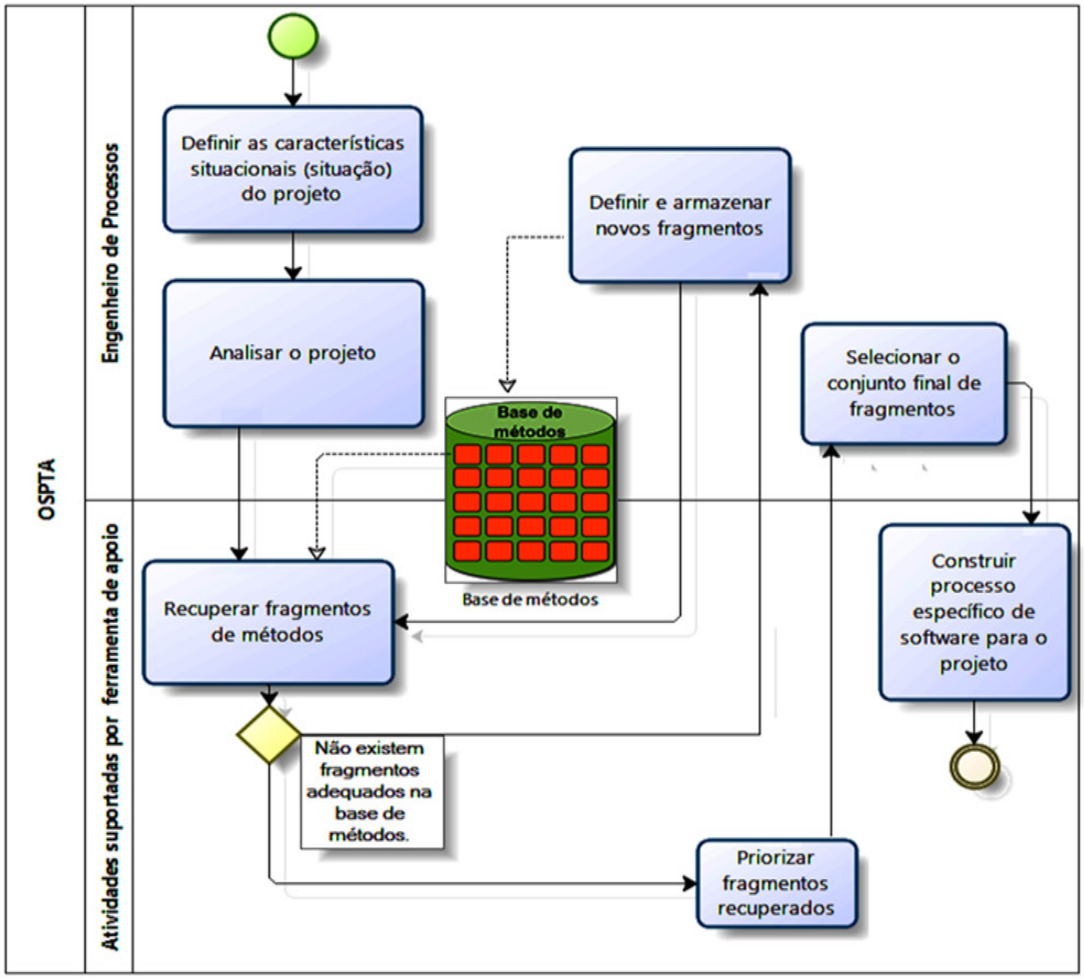


Figura 7 – Visão geral da abordagem OSPTA

- ***Analisar o projeto:*** então o projeto deve ser analisado em relação ao(s) critério(s) de adaptação estabelecido(s). Por exemplo, se o critério de adaptação definido for "Riscos do Projeto", o projeto deve ser analisado com objetivo de identificar quais riscos o projeto está exposto.
- ***Recuperar fragmentos de métodos:*** os fragmentos de métodos (*process-fragments*) adequados em relação a um ou mais critérios de adaptação são recuperados da base de métodos de acordo com o resultado da atividade

anterior. Assumindo "Riscos do Projeto" como critério de adaptação e supondo que foi identificado o risco "Não Entendimento dos Requisitos" para o projeto, todos os fragmentos que visam prevenir este risco que existirem na base de métodos são recuperados. Neste momento, independe se o contexto de uso do fragmento é similar ao contexto do projeto. Por exemplo, de acordo com a base de conhecimento ou base de métodos utilizada nos estudos de caso desta dissertação, seriam recuperados os fragmentos: "*BuildPrototype*", "*ConstantRefactoring*", "*EarlyAndRegularRUP*", entre outros.

- ***Definir e armazenar novos fragmentos:*** deve haver na base de métodos fragmentos previamente associados ao critério de adaptação. Por exemplo, cada fragmento deve estar relacionado a um ou mais riscos que ele pode prevenir. Caso não existam fragmentos no repositório relacionados ao critério de adaptação definido, o responsável deve editar um fragmento existente ou definir novos elementos e armazená-los na base de métodos.
- ***Priorizar fragmentos recuperados:*** a ferramenta de apoio, usando um algoritmo de priorização, classifica os fragmentos previamente recuperados em ordem de relevância em relação ao contexto do projeto para sugerir ao engenheiro de processo os fragmentos mais apropriados para serem inclusos no processo de software adaptado. Essa priorização se dá através de comparações critério a critério do contexto de uso de cada fragmento com o contexto do projeto, através da técnica AHP (SAATY, 1980).
- ***Selecionar o conjunto final de fragmentos:*** após a priorização dos fragmentos pelo algoritmo, o engenheiro de processos pode selecionar o conjunto final de fragmentos de métodos, seguindo a priorização da atividade anterior, que serão inclusos no processo adaptado, e deve fazer esta escolha.
- ***Construir o processo de desenvolvimento de software especifico para o projeto:*** finalmente, o processo adaptado é construído pela ferramenta de apoio e o *web site* do modelo de processo é construído.

6.3 Priorização de fragmentos de métodos

Na abordagem OSPTA, a priorização dos fragmentos de métodos previamente recuperados de acordo com o(s) critério(s) de adaptação é realizada a partir dos fatores do *Octopus Model,* analisando o contexto de uso dos fragmentos em relação ao contexto do projeto.

O mecanismo utilizado nesta dissertação para priorização de fragmentos de métodos em relação ao contexto do projeto é baseado na técnica *Analytic Hierarchy Process* (AHP), tendo como critérios de comparação os oito critérios do *Octopus Model*.

AHP (SAATY, 1980) é uma técnica para tomada de decisão em ambientes complexos, onde muitas variáveis ou critérios devem ser levados em conta para seleção e priorização de alternativas. Trata-se de um modelo matemático para apoio à teoria de decisão.

Um dos pontos críticos de sucesso em abordagens baseadas em SME está relacionado a fazer escolhas certas e consistentes dos fragmentos de métodos, dada uma situação específica. A priorização dos fragmentos de métodos é uma ordenação baseada na análise do contexto de uso do fragmento em relação ao contexto do projeto para guiar os engenheiros de processo na escolha dos melhores fragmentos.

Segundo Vargas (2010), a capacidade de transformar dados empíricos em modelos matemáticos é a principal contribuição da AHP em relação a outras técnicas. Esta característica permite a escolha de alternativas mais corretas e consistentes em ambientes complexos.

AHP trabalha com alternativas e com uma meta global, um objetivo. Em resumo, a probabilidade numérica de cada alternativa é calculada, através dos métodos matemáticos da AHP, e quanto maior for esta probabilidade, maior é a chance de uma alternativa satisfazer o objetivo (VARGAS, 2010).

Na abordagem proposta nesta dissertação, às alternativas correspondem aos fragmentos de métodos armazenados, enquanto a meta global é priorizar tais fragmentos para guiar o engenheiro de processos na escolha dos melhores elementos para serem inclusos no processo adaptado.

A técnica de priorização proposta permite que sejam atribuídos pesos relativos para cada um dos critérios de comparação considerados na priorização, neste caso os

fatores do *Octopus Model,* possibilitando que se atribua uma importância relativa dos fatores em relação à meta global.

Para priorizar os fragmentos, primeiramente, são estabelecidos pelo engenheiro de processo (de acordo com suas experiências), os pesos relativos dos critérios de comparação, visando determinar a importância relativa de cada um. A escala de relativa importância de uma alternativa em relação à outra, proposta por Saaty (2005), é amplamente utilizada, com valores que variam de 1 (igualmente preferido) a 9 (extremamente preferido),. Os valores desta escala, utilizada nesta dissertação para determinar os pesos relativos e, consequentemente, a importância relativa entre os critérios de comparação são mostrados na Tabela 8.

Tabela 8 – Escala de relativa importância

Escala	**Avaliação Numérica**	**Recíproco**
Extremamente preferido	9	1/9
Muito fortemente preferido	7	1/7
Fortemente preferido	5	1/5
Moderadamente preferido	3	1/3
Igualmente preferido	1	1

Fonte: (SAATY, 2005)

O engenheiro de processos deve estabelecer pesos relativos entre os critérios de comparação (fatores do *Octopus Model*) para que seja determinada a importância relativa entre eles. A partir desta escala é criada uma matriz de comparação, como é possível observar na Tabela 9.

A Tabela 9 representa uma matriz comparativa entre dois critérios de comparação, supondo que o Critério 1 é mais relevante que o Critério 2 para a meta global. Uma matriz em que todos os valores fossem “1” significaria todos os critérios com a mesma importância relativa.

Tabela 9 – Matriz comparativa entre dois critérios de comparação

	Critério 1	**Critério 2**
Critério 1	1	Avaliação Numérica
Critério 2	1/Avaliação Numérica (Recíproco)	1

Fonte: (VARGAS, 2010)

No caso desta dissertação, tem-se uma matriz com oito linhas e oito colunas, referentes a oito critérios de comparação, que são os fatores do *Octopus Model*. É possível observar na Figura 8 um exemplo de uma matriz de comparação entre os fatores do *Octopus Model*.

Através da Figura 8 é possível observar que o critério "Tamanho" é fortemente preferido (Tabela 8) em relação ao critério "Controle", para a meta global.

	Tamanho	Criticidade	Modelo de Negócio	Arquitetura Estável	Distribuição da Equipe	Controle	Taxa de Mudanças	Idade do Sistema
Tamanho	1	1	3	3	1	5	1	3
Criticidade	1	1	3	3	1	5	1	3
Modelo de Negócio	1/3	1/3	1	1	1/3	3	1/3	1
Arquitetura Estável	1/3	1/3	1	1	1/3	3	1/3	1
Distribuição da Equipe	1	1	3	3	1	5	1	3
Controle	1/5	1/5	1/3	1/3	1/5	1	1/5	1/3
Taxa de Mudanças	1	1	3	3	1	5	1	3
Idade do Sistema	1/3	1/3	1	1	1/3	3	1/3	1

Figura 8 – Matriz comparativa dos fatores do *Octopus Model*

Para interpretar e dar os pesos relativos a cada critério, a matriz anterior é normalizada. Isto se dá pela divisão de cada valor da matriz pelo somatório dos valores da coluna do valor em questão (VARGAS, 2010).

O resultado da contribuição, importância relativa, de cada critério na meta global é calculada a partir de um vetor de prioridade ou vetor de Eigen. Este apresenta os pesos

relativos entre os critérios, sendo obtido de modo aproximado, através da média aritmética de cada um dos critérios, ou seja, média aritmética dos valores de cada linha da matriz, sendo que cada linha corresponde a um critério (VARGAS, 2010).

Através da Figura 9 é possível observar os valores dos pesos relativos de cada critério de comparação, de acordo com os valores colocados na Figura 8.

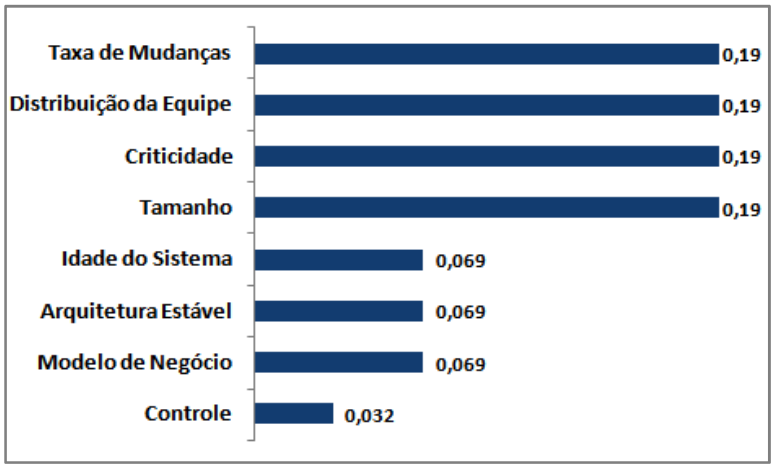


Figura 9 – Pesos relativos dos critérios de comparação a partir da Figura 8

De acordo com a Figura 9, o fator “Tamanho” tem um peso de 19% enquanto “Idade do Sistema” possui peso de 6,9% em relação à meta global. Isso significa dizer que o fator “Tamanho” contribui 2,75 vezes mais do que “Idade do Sistema”.

Através de métodos matemáticos propostos pela técnica AHP, é conferida a consistência dos valores atribuídos. Supondo critérios A, B e C, se o gerente de processo afirmar (ver Figura 8) que A > B e B > C, seria inconsistente afirmar que A < C.

Após as prioridades dos critérios estabelecidas, é possível determinar o comportamento de cada um dos fragmentos previamente selecionados em relação aos critérios estabelecidos. Para tal, os fragmentos candidatos são confrontados dois a dois dentro de cada um dos critérios estabelecidos, ou seja, para cada um dos fatores do *Octopus Model*.

Esta comparação se dá pela avaliação, por parte da ferramenta apresentada no Capítulo 7, do valor do contexto de uso do fragmento com o valor do contexto do

projeto em relação ao critério que está sendo analisado, e a partir daí, valores são atribuídos na matriz. Quanto mais próximo for o valor do contexto de uso do fragmento do contexto do projeto em relação a determinado critério, maior é o valor atribuído. Por exemplo, para o critério "Tamanho", sendo o valor deste critério no contexto do projeto correspondente a "Pequeno", se o valor referente a este critério no contexto de uso de um fragmento candidato for o mesmo, este recebe um valor na matriz comparativa maior do que um fragmento com valor diferente.

A Figura 10 ilustra a descrição do contexto de um projeto, enquanto a Figura 11 ilustra a descrição do contexto de uso de dois fragmentos de métodos (*process-fragments*) distintos.

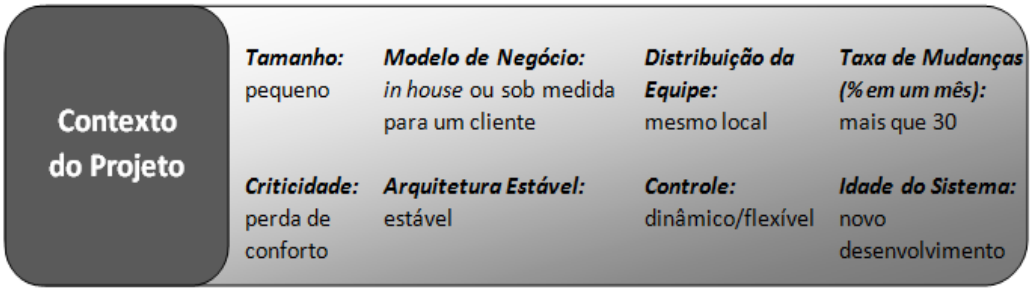


Figura 10 – Contexto de um projeto

A Tabela 10 ilustra a matriz comparativa destes dos fragmentos candidatos mostrados na Figura 11 em relação ao critério "Tamanho". Tal matriz é montada pela ferramenta de apoio, com base no contexto do projeto (Figura 10) e no contexto de uso dos fragmentos (Figura 11), para calcular a probabilidade relativa dos fragmentos em relação a cada critério de comparação.

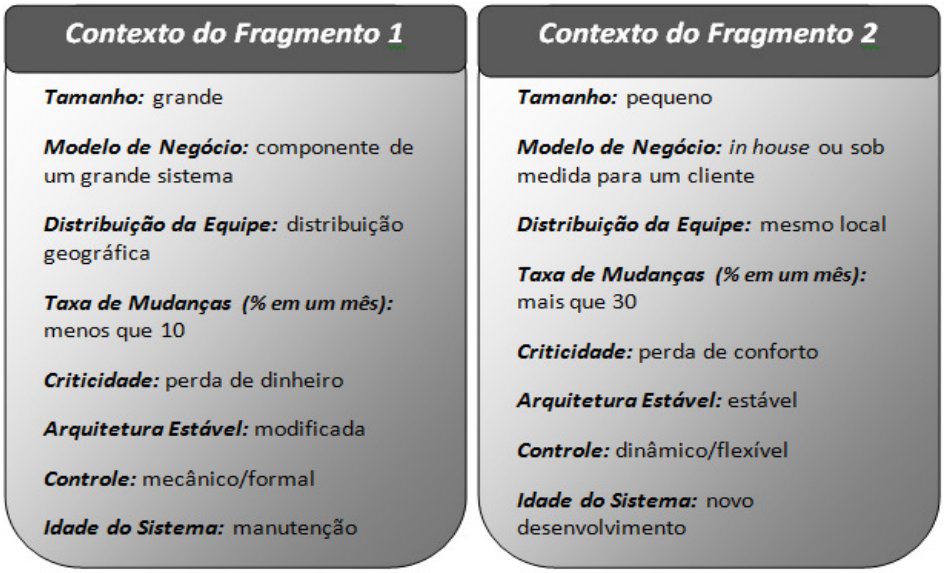


Figura 11 – Contexto de uso de dois fragmentos

Tabela 10 – Matriz comparativa dos fragmentos em relação ao critério "Tamanho"

	Fragmento 1	**Fragmento 2**
Fragmento 1	1	1/9
Fragmento 2	9	1

A matriz indica que o "Fragmento 2" é extremamente preferido (Tabela 8) em relação ao "Fragmento 1" para o contexto do projeto levado em conta, neste caso o contexto mostrado na Figura 10. Em outras palavras, o uso do "Fragmento 2" é nove vezes mais indicado do que o uso do "Fragmento 1" para este contexto.

A partir desta matriz comparativa relacionada a cada critério de comparação, a ferramenta calcula as probabilidades relativas de cada fragmento. Assim, cada fragmento de método terá sua probabilidade relativa para cada fator do *Octopus Model*. Esta probabilidade relativa significa a chance que o fragmento tem de atender com sucesso a meta global em relação a cada critério.

O cálculo das prioridades dos fragmentos para cada critério de comparação acontece através dos métodos matemáticos propostos pela técnica AHP já utilizados anteriormente para definir os pesos relativos dos critérios em relação à meta global.

A Figura 12 apresenta os resultados da priorização dos fragmentos candidatos em relação ao critério tamanho.

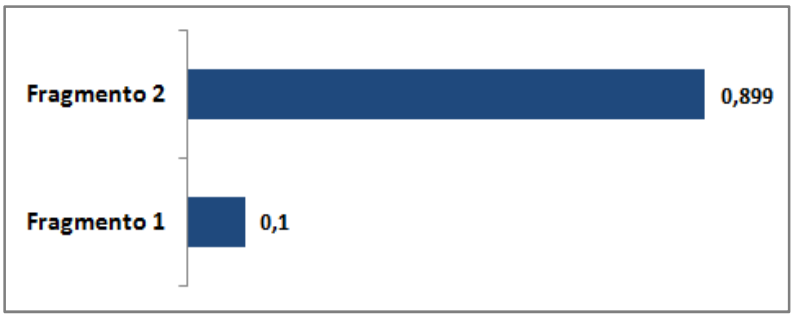


Figura 12 – Fragmentos priorizados em relação ao critério “Tamanho”

Após isso, é calculada a probabilidade final de cada fragmento em relação à meta global. O mecanismo para calcular a prioridade final para o fragmento é a soma dos produtos do peso relativo de cada critério pela probabilidade relativa do fragmento em relação ao mesmo critério.

Na Tabela 11 é possível observar o mecanismo para o cálculo da probabilidade final do “Fragmento 2” em relação a um projeto com contexto mostrado na Figura 10, levando em conta os pesos relativos dos critérios de comparação mostrados na Figura 9.

Para cada instância do(s) critério(s) de adaptação é definida uma lista de fragmentos priorizados.

No caso do exemplo dos riscos do projeto como critério de adaptação, para cada risco identificado para o projeto, são selecionados todos os fragmentos associados a este risco e então é definida uma lista ordenada dos fragmentos, sendo que quanto mais parecido for o contexto de uso do fragmento com o contexto do projeto, maior é sua probabilidade final.

Através dos estudos de caso (Capítulo 8) é possível ter uma visão mais detalhada desta priorização

A Figura 13 apresenta uma visão gera sobre o mecanismo para priorização de fragmentos de métodos.

Tabela 11 – Cálculo da probabilidade final de um fragmento em relação à meta global

Critério de Comparação	**Peso Relativo do Critério**	**Probabilidade Relativa do Fragmento**	**Produto**
Tamanho	0.899	0.190	0.170
Criticidade	0.642	0.190	0.122
Modelo de Negócio	0.899	0.069	0.062
Arquitetura Estável	0.642	0.069	0.044
Distribuição da Equipe	0.899	0.190	0.171
Controle	0.899	0.031	0.028
Taxa de Mudanças	0.899	0.190	0.171
Idade do Sistema	0.642	0.069	0.044
		Probabilidade Final do Fragmento *(x100)*	**81.2** *(aprox.)*

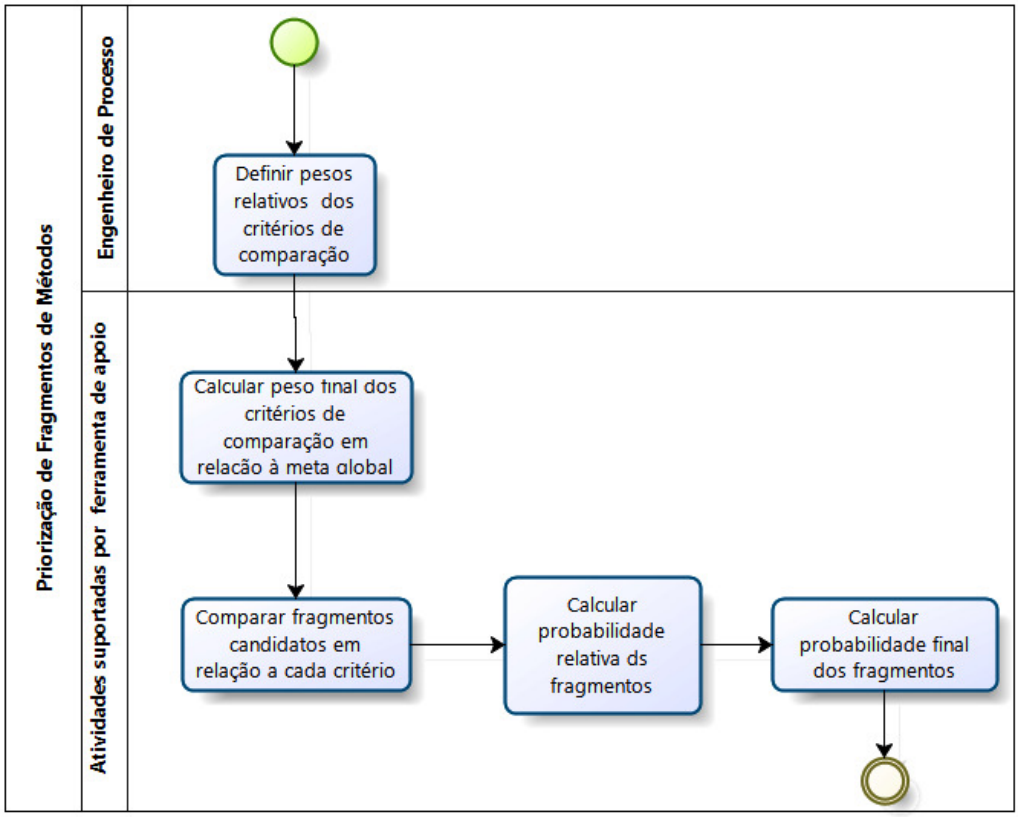


Figura 13- Visão geral do mecanismo para priorização de fragmentos

7. Ferramenta de Suporte para a Abordagem OSPTA

A implementação da ferramenta *Octopus SME Process Tailoring Support Tool* tem como finalidade possibilitar uma ilustração do uso da abordagem proposta.

O ambiente foi desenvolvido utilizando-se a Linguagem Java para especificação das regras de negócio e acesso a dados; *Java Server Pages* (JSP), HTML e JQuery na camada de apresentação, banco de dados MySQL e o servidor de aplicação Tomcat.

Na Figura 14 pode ser visualizado o modelo conceitual contendo as principais classes da ferramenta *Octopus SME Process Tailoring Support Tool*.

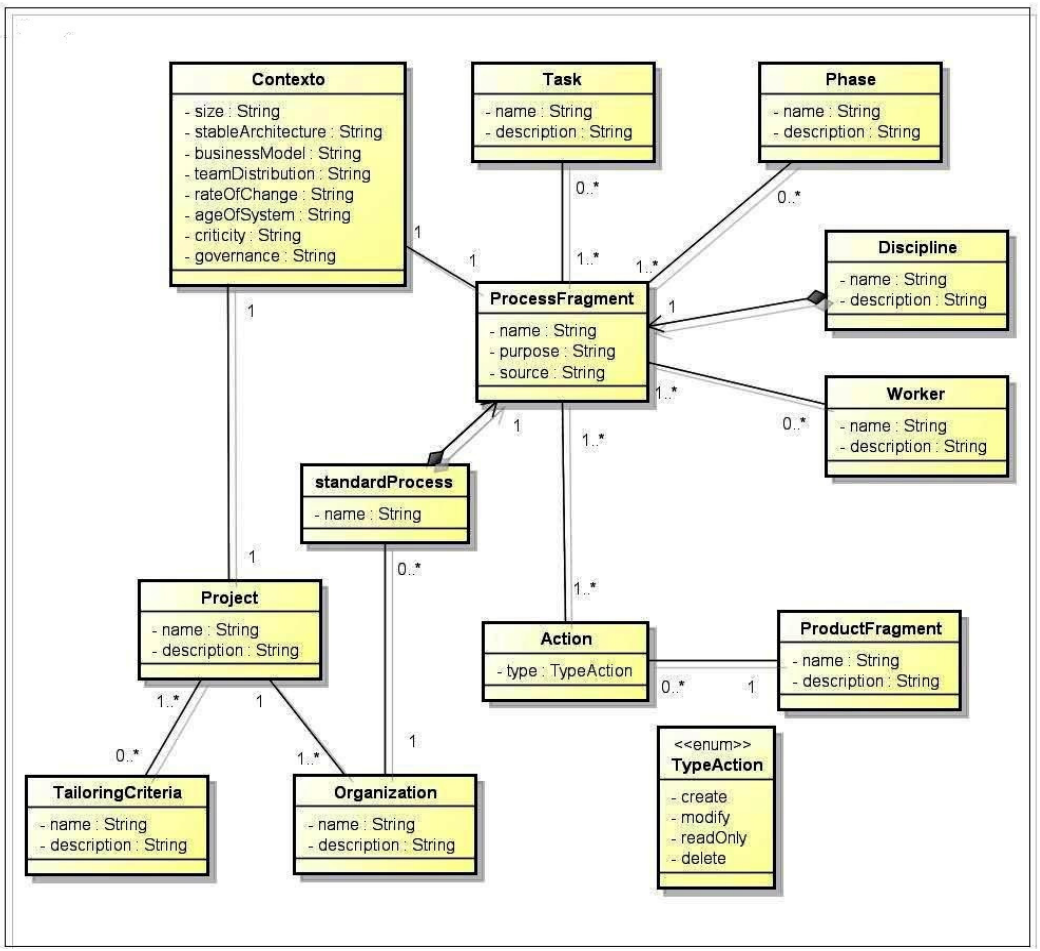


Figura 14- Modelo conceitual - *Octopus SME Process Tailoring Support Tool*

A base de conhecimento utilizada foi adaptada da base existente em PRiMA-Tool (Fontoura, 2006), uma ferramenta desenvolvida para gerenciamento de riscos em projetos de software através de adaptação de processos.

A Seção 7.1 descreve uma visão geral da ferramenta. A Seção 7.2 é ilustrado o uso da ferramenta.

7.1 Visão Geral

A ferramenta para apoiar a abordagem proposta nesta dissertação é responsável pela definição das características situacionais do projeto (critério(s) de adaptação e contexto do projeto), e pela definição, seleção e priorização dos fragmentos de métodos. Além disso, a ferramenta inclui no processo padrão da organização, os fragmentos selecionados em última instância, pelo engenheiro de processos, gerando o processo adaptado específico para o projeto em questão. Ainda, através da ferramenta é possível definir todos os elementos necessários para a definição do processo adaptado de software.

A Figura 15 exibe a tela principal da ferramenta. No menu à esquerda são exibidas as funcionalidades da ferramenta, sendo que estas estão organizadas em três grupos, que são: "*Organization*", "*Project*" e "*Knowledge Base*".

- ***Organization***: descreve as funcionalidades relacionadas à organização, tais como: cadastrar uma nova organização, criar um processo padrão para a organização e editar o processo padrão da organização, definido previamente;
- ***Project***: descreve as funcionalidades relacionadas ao projeto de software, tais como: cadastrar um novo projeto, editar as propriedades de um projeto, e definir as características situacionais do projeto pela definição dos critérios de adaptação para o projeto e do contexto do projeto;
- ***Knowledge Base***: armazena o conhecimento organizacional, sendo dividida em dois grupos, que são:

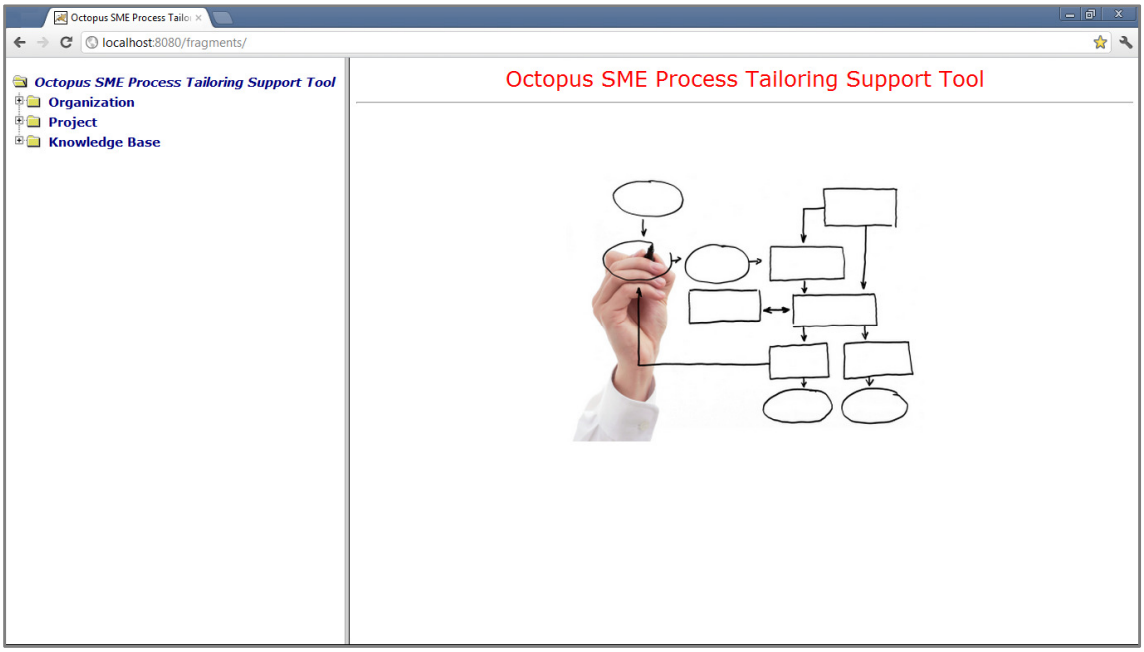


Figura 15 - Tela inicial - *Octopus SME Process Tailoring Support Tool*

Na Figura 16 é exibido o menu da ferramenta, de maneira detalhada.

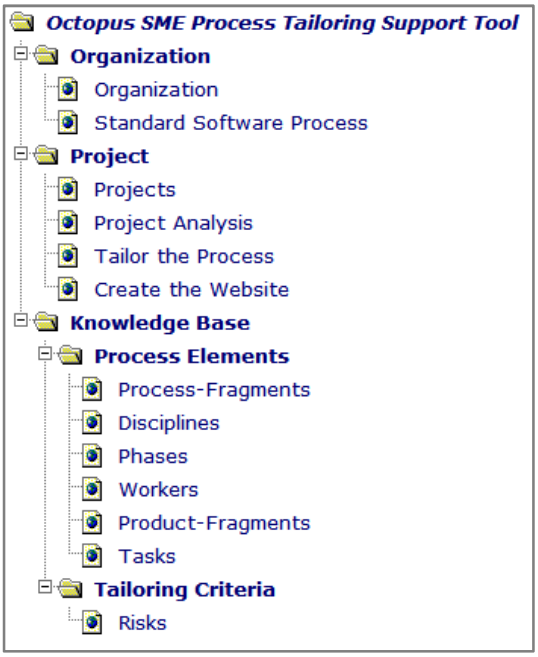


Figura 16 – Funcionalidades da ferramenta *Octopus SME Process Tailoring Support Tool*

Para que a ferramenta seja utilizada de acordo com a abordagem proposta, são necessárias algumas configurações.

Primeiramente, elementos de processos representados segundo os conceitos descritos no metamodelo M^2F devem ser definidos, através das funcionalidades agrupadas em "*Process Elements*". A ferramenta desenvolvida permite que a criação e edição de cada um destes elementos. Como exemplo, a Figura 17 mostra a tela inicial para o cadastro de fragmentos, na qual são cadastradas propriedades do fragmento como nome, origem, propósito, disciplina, entre outras. Além disso, é possível definir o contexto de uso do fragmento.

Em uma segunda etapa, como mostra a Figura 18, são associados papéis, tarefas, e *product-fragments* ao fragmento de método. Além disso, como já existe cadastrado na base de conhecimento o critério de adaptação "Riscos do Projeto", é possível associar ao fragmento riscos que ele pode prevenir.

Além dos elementos de processo e critérios de adaptação, é preciso definir o processo de software padrão da organização. Estas configurações de processo são definidas através das opções "*Organization → Standard Software Process*", com a escolha de elementos de *process-fragments* disponíveis para comporem o processo padrão da organização.

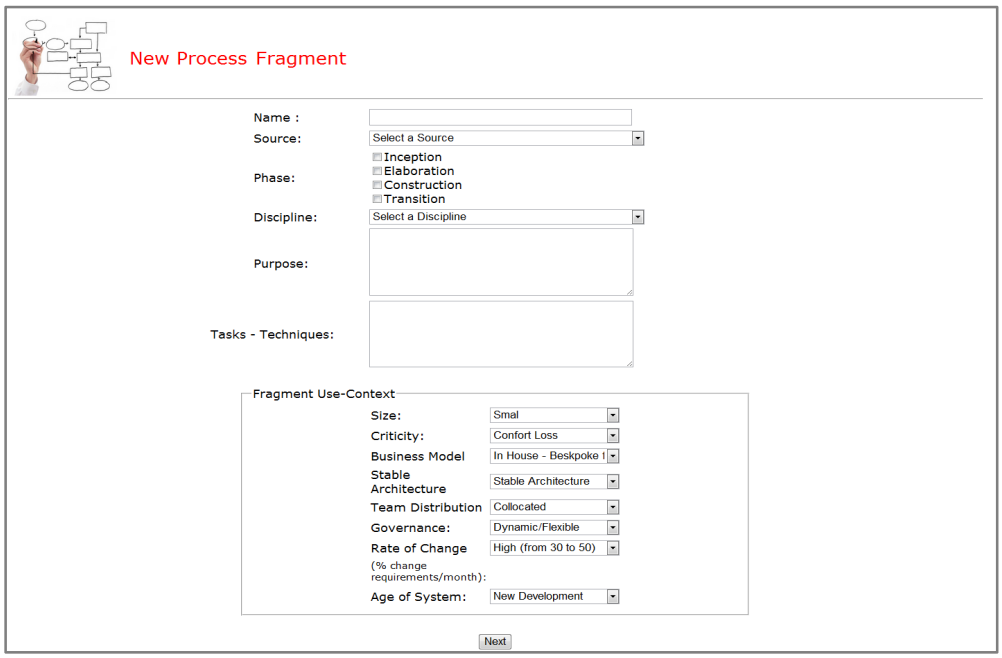


Figura 17 - Tela para criar um novo fragmento – parte 1

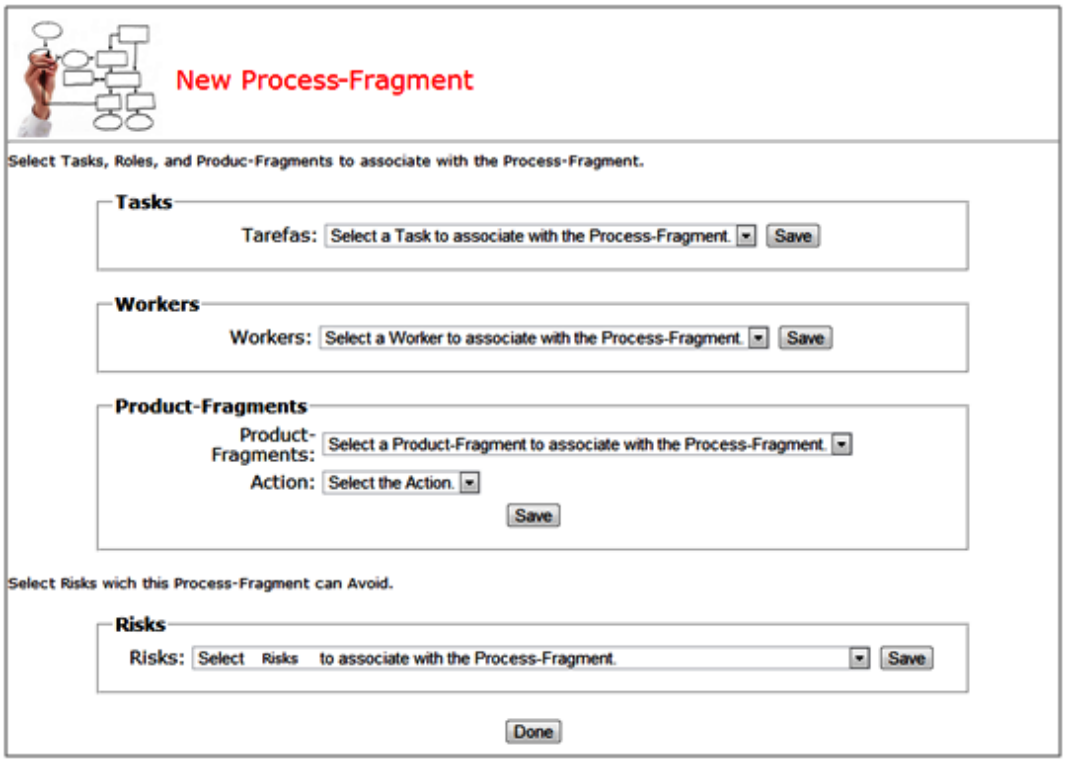


Figura 18 - Tela para criar um novo fragmento – parte 2

7.2 Ilustrando o uso da ferramenta *Octopus SME Process Tailoring Support Tool*

Para seguir a abordagem proposta nesta dissertação através da ferramenta *Octopus SME Process Tailoring Support Tool*, deve existir um processo padrão associado a uma organização. A definição de um projeto é feita através da funcionalidade "*Project*", enquanto o cadastro de uma organização e a definição de um processo de software padrão para a mesma são feitos através das funcionalidades "*Organization*" e "*Standard Software Process*". A Figura 19 apresenta o formulário para cadastrar uma organização, enquanto a Figura 20 apresenta o formulário para associar um modelo de processo padrão à organização.

Havendo uma organização, com um processo padrão definido, é preciso também existir um projeto de software com contexto definido de acordo com os fatores de contextualização do *Octopus Model*. A Figura 21 mostra a o formulário para cadastrar um projeto e definir o contexto do mesmo.

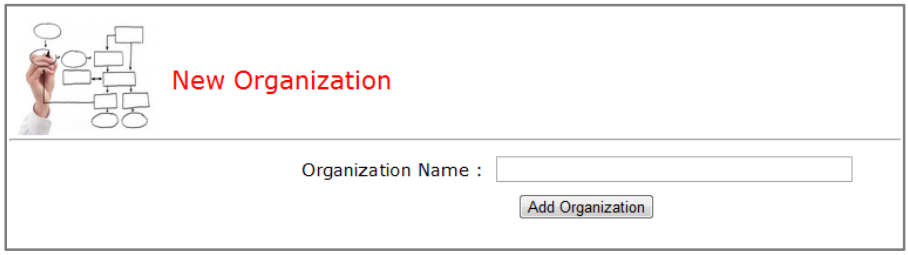


Figura 19 – Formulário para cadastrar uma organização

Figura 20 – Formulário para associar um processo padrão a uma organização

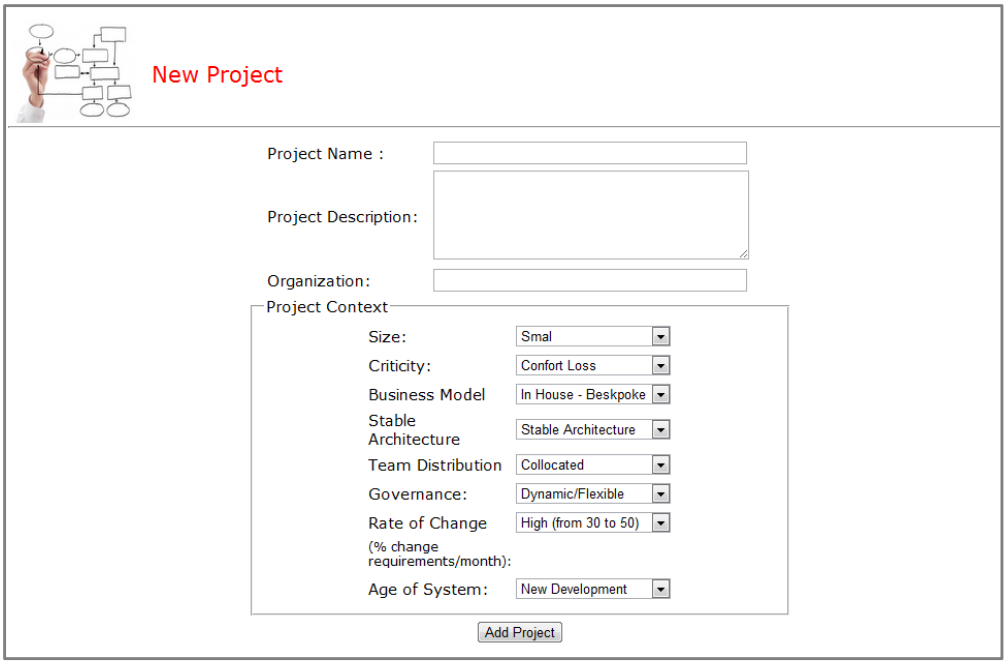


Figura 21 – Formulário para cadastrar um novo projeto

A análise de um projeto é feita para relacionar ao projeto os critérios de adaptação definidos para o mesmo. Seguindo com o exemplo de "Riscos do Projeto" como o critério de adaptação cadastrado na ferramenta.

A Figura 22 ilustra riscos associados ao projeto e o formulário para a associação de novos riscos.

Após isso, através de "*Tailor the Process*", a ferramenta recupera os fragmentos de método adequados para o projeto escolhido e os prioriza de acordo com as características situacionais do projeto. Em outras palavras, os fragmentos são recuperados de acordo com os riscos do projeto e priorizados de acordo com o contexto do mesmo.

A Figura 23 mostra a tela com fragmentos priorizados para que o engenheiro de processos possa fazer a escolha final de quais serão incorporados no processo padrão da organização, compondo o processo adaptado. Estão expostos fragmentos priorizados em relação aos riscos "*Non-realistic schedule and budget*" e "*Misunderstading the requirements*".

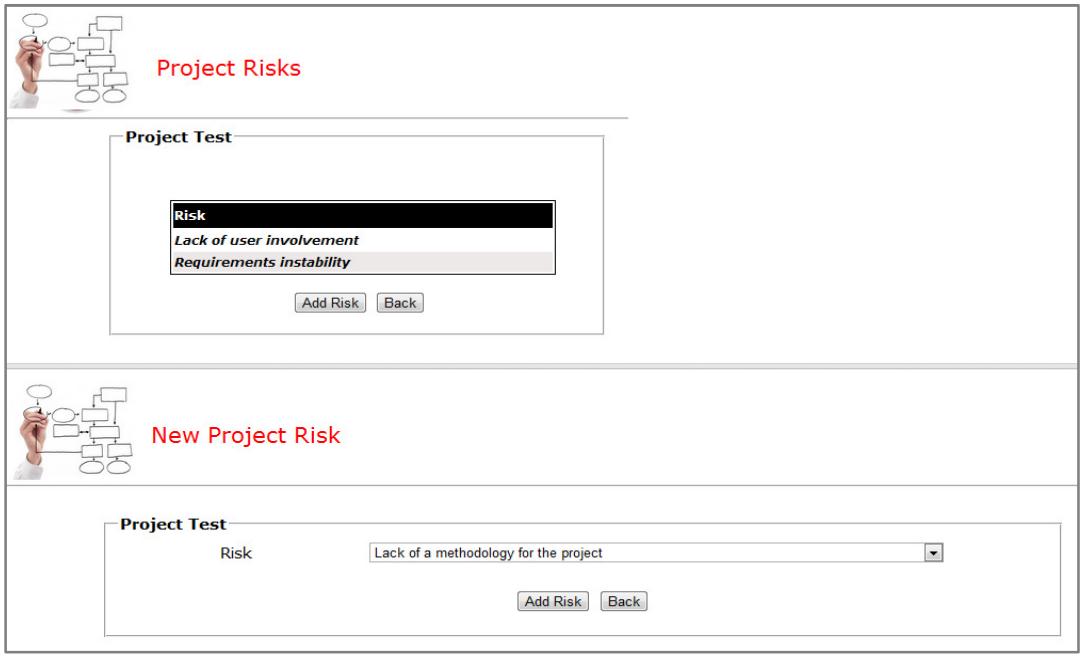
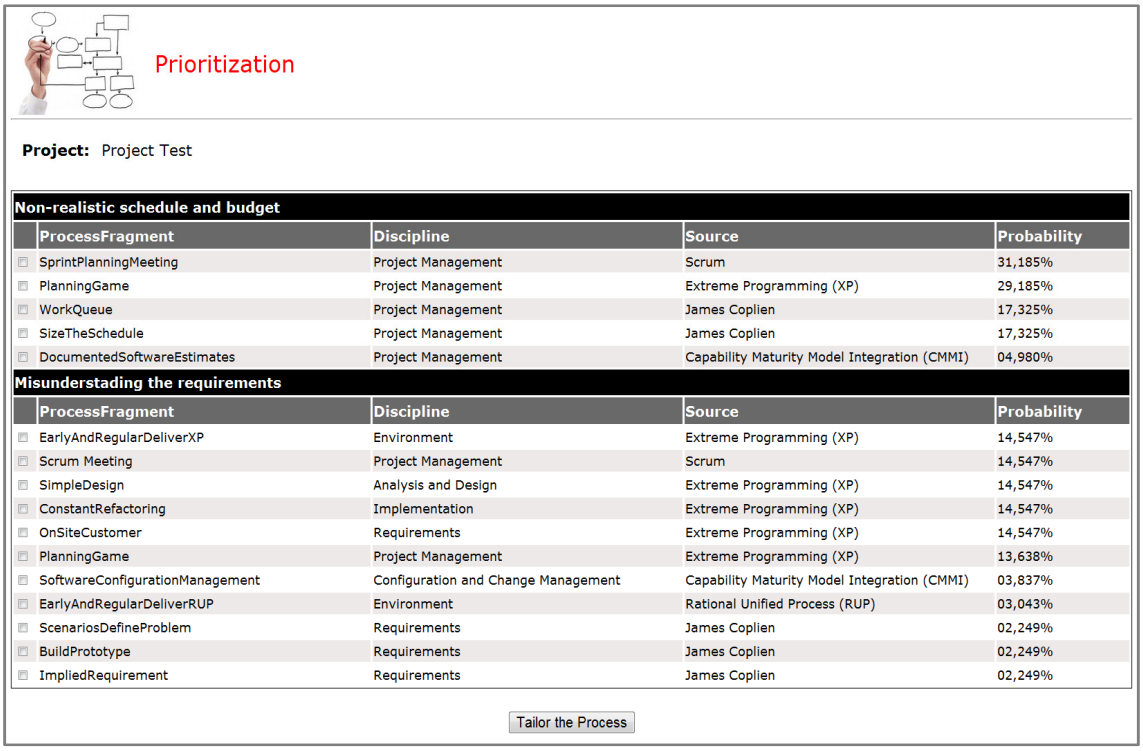


Figura 22 – Relacionando riscos (critério de adaptação) ao projeto



Prioritization

Project: Project Test

Non-realistic schedule and budget

ProcessFragment	Discipline	Source	Probability
SprintPlanningMeeting	Project Management	Scrum	31,185%
PlanningGame	Project Management	Extreme Programming (XP)	29,185%
WorkQueue	Project Management	James Coplien	17,325%
SizeTheSchedule	Project Management	James Coplien	17,325%
DocumentedSoftwareEstimates	Project Management	Capability Maturity Model Integration (CMMI)	04,980%

Misunderstading the requirements

ProcessFragment	Discipline	Source	Probability
EarlyAndRegularDeliverXP	Environment	Extreme Programming (XP)	14,547%
Scrum Meeting	Project Management	Scrum	14,547%
SimpleDesign	Analysis and Design	Extreme Programming (XP)	14,547%
ConstantRefactoring	Implementation	Extreme Programming (XP)	14,547%
OnSiteCustomer	Requirements	Extreme Programming (XP)	14,547%
PlanningGame	Project Management	Extreme Programming (XP)	13,638%
SoftwareConfigurationManagement	Configuration and Change Management	Capability Maturity Model Integration (CMMI)	03,837%
EarlyAndRegularDeliverRUP	Environment	Rational Unified Process (RUP)	03,043%
ScenariosDefineProblem	Requirements	James Coplien	02,249%
BuildPrototype	Requirements	James Coplien	02,249%
ImpliedRequirement	Requirements	James Coplien	02,249%

Tailor the Process

Figura 23 – Priorização de fragmentos de métodos

Através do botão "*Tailor the Process*", os fragmentos selecionados são incluídos no processo padrão da organização, formando o processo adaptado.

Após a adaptação do processo de software, a ferramenta disponibiliza um link para a *web site* que descreve o processo. Tal *web site* pode ser visualizada através da opção "*Process Project Web Site*".

A Figura 24 ilustra a *web site* do processo adaptado para o projeto, sendo que na parte esquerda encontra-se o menu contendo os elementos de processo. A Figura 25 ilustra o menu de navegação do processo adaptado de software.

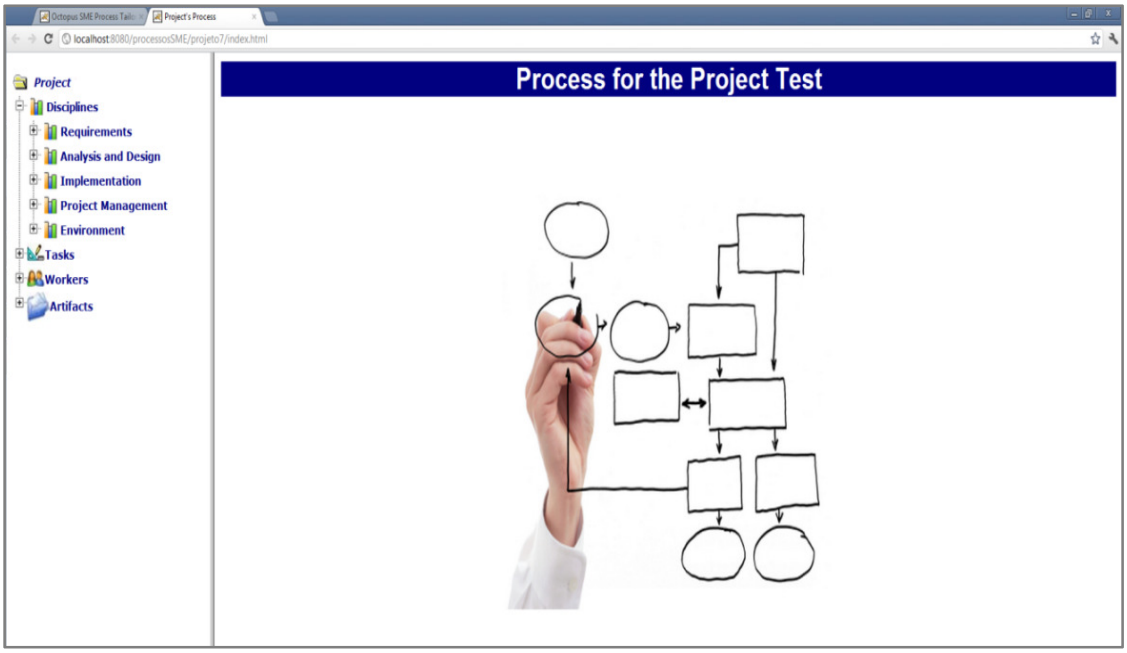


Figura 24 – Web site do processo específico

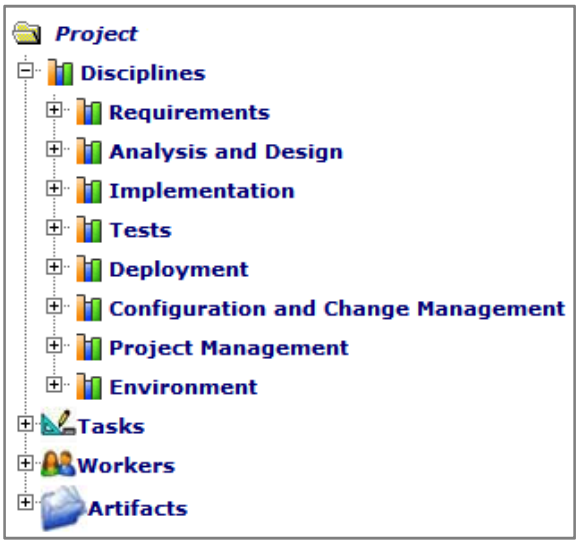


Figura 25 – Menu de navegação do processo adaptado de software

8. Ilustrando a abordagem OSPTA

A ilustração de uso da abordagem proposta nesta dissertação é feita através de casos hipotéticos de uso, que ilustram dois projetos de desenvolvimento de software com características distintas.

8.1 Configuração da validação

Nos estudos de casos ilustrados neste Capítulo, a abordagem proposta foi aplicada com objetivo de construir um processo adaptado específico para um projeto de modo a prevenir riscos identificados para o mesmo. Assim, o requisito do processo específico é a prevenção dos riscos do projeto, ou seja, o critério de adaptação são os riscos do projeto, conforme proposto em Pereira (2011).

Dessa forma, o objetivo é a recuperação de fragmentos apropriados à situação específica do projeto, ou seja, que contenham práticas que levem a prevenção de riscos do projeto de software e que sejam adequados ao contexto do mesmo. Então tais fragmentos formarão o processo adaptado de software a partir do processo de software padrão da organização.

É desejada uma validação da abordagem proposta nesta dissertação, demonstrando que os fragmentos de métodos recuperados da base de métodos de acordo com o critério de adaptação são efetivamente priorizados em relação ao contexto do projeto. Em outras palavras, para projetos com contextos planejados, os fragmentos com contexto de uso desta natureza são considerados mais relevantes em relação a fragmentos com contexto de uso de natureza ágil, e vice-versa. Assim, são incluídos no processo de software adaptado os fragmentos de métodos mais adequados ao contexto do projeto.

Foram atribuídos pesos relativos, para atribuir diferentes importâncias, aos critérios que contextualizam o projeto para a priorização dos fragmentos. Os valores dos pesos relativos considerados são os mesmos mostrados na Figura 8 e na Figura 9.

A base de conhecimento utilizada para esta validação que contém a associação entre riscos do projeto e os fragmentos de métodos, é adaptada de Fontoura (2006). É importante ressaltar que a base de conhecimento utilizada nos estudos de caso trata-se de uma sugestão, podendo ser adaptada de acordo com as necessidades, experiências e práticas de cada organização, e devendo ser atualizada de acordo com o aprendizado de projetos passados.

Os estudos de caso criados para validar a abordagem proposta levam em conta dois projetos com características distintas, pertencentes a uma mesma organização de desenvolvimento de software com um processo de software padrão definido.

Será atribuído o nome "Organização X" para e empresa de desenvolvimento de software fictícia presentes nos estudos de caso. O processo padrão da Organização X é bastante simples, baseado em um pequeno conjunto de fragmentos de métodos extraídos do RUP (RATIONAL, 2003).

A Tabela 12 mostra os fragmentos de método que fazem parte do processo de software padrão da organização.

8.2 Estudo de Caso 1 - SisAcad

O primeiro caso de uso apresenta um projeto para o desenvolvimento de um software para gestão de uma rede de academias de ginástica, chamado neste estudo de "SisAcad".

Determinada rede de academias de ginástica não possui um sistema informatizado para gerenciar informações sobre a gestão dos processos de negócio da academia e sobre as pessoas envolvidas, como professores, alunos e médicos, etc. Assim, toda a gestão de informações é executada utilizando formulários de papel.

Neste contexto, a rede de academias deseja um sistema que controle todos os procedimentos necessários e auxilie na gerência de informações. Assim, foi solicitado para a Organização X um novo sistema web, sob medida de acordo com as necessidades das academias de ginástica.

Tal sistema será desenvolvido por um time pequeno, que trabalha no mesmo local, em um ambiente dinâmico de gestão e trabalho.

Tabela 12 – Processo padrão da Organização X

Disciplina	**Fragmento**
Requirements	*AnalyzeTheProblem*
Analysis and Design	*DefineCandidateArchitecture*
	DesignTheDatabase
Implementation	*ImplementComponents*
	IntegrateTheSystem
Test	*TestAndEvaluate*
	ImproveTestAssets
Deployment	*ProduceDeploymentUnit*
	ProvideAccessToDownloadSite
Configuration and Change Management	*ManageChangeRequests*
Project Management	*PlanTheProject*
	Monitor&ControlProject
Environment	*PrepareEnvironmentForProject*

Tratando-se de um sistema customizado para o cliente, os usuários serão estimulados frequentemente a fornecerem *feedback* sobre o sistema a medida que partes entregáveis do sistema sejam disponibilizadas, o que pode levar a uma taxa média a alta de mudanças nos requisitos do sistema.

A Tabela 13 e a Tabela 14 ilustram situação específica do projeto “SisAcad”. A Tabela 13 mostra os riscos identificados para o projeto, enquanto a Tabela 14 mostra a contextualização do projeto de acordo com o *Octopus Model*.

A Figura 26 mostra os fragmentos recuperados de acordo com os riscos do projeto “SisAcad” e priorizados de acordo com o contexto do mesmo. Os fragmentos marcados com ✓ são selecionados pelo engenheiro de processos para serem incluídos no processo adaptado especifico para o projeto “SisAcad”.

Tabela 13 – Riscos do projeto “SisAcad”

Risco	Categoria
Infeasible Design	Execução
Insufficient/inappropriate staffing	Planejamento
Lack of user involvement	Cliente
Misunderstanding the requirements	Requisitos

Tabela 14 – Contextualização do “SisAcad”

Critério	Valor Estabelecido
Tamanho	Pequeno.
Criticidade	Perda de conforto.
Arquitetura estável	Estável.
Modelo de negócio	*In house* ou sob medida para um cliente.
Distribuição da equipe	Mesmo local.
Taxa de mudanças **(% em um mês)**	Mais que 30.
Idade do sistema	Novo desenvolvimento.
Controle	Flexível/Orgânica.

Através da Figura 26 é possível identificar que os fragmentos sugeridos com maior probabilidade (“*Probability*”) são propostos por abordagens ágeis. Isto se dá, de acordo com a abordagem proposta nesta dissertação, porque tais fragmentos possuem um contexto de uso semelhante ao contexto do projeto, o qual possui um contexto ágil. Os fragmentos com probabilidade mais baixa têm origem de abordagens planejadas e, portanto, é possível afirmar que são pouco aconselhados para serem inclusos no processo adaptado para o projeto em questão.

Quanto maior a probabilidade, mais adequado é o fragmento para o projeto em questão, em outras palavras, mais indicado é o seu uso ou sua inclusão no processo adaptado específico para o projeto.

A Figura 27 mostra os fragmentos contidos no processo adaptado para o projeto "SisAcad". É possível observar que além dos fragmentos pertencentes ao processo padrão da organização (Tabela 12), também fazem parte do processo adaptado os fragmentos selecionados pelo gerente de processo, após a recuperação e priorização dos fragmentos indicados pela ferramenta de apoio (Figura 26). Os fragmentos destacados são aqueles específicos do processo do projeto, os demais são os descritos pelo processo padrão da organização.

Infeasible design			
	ProcessFragment	**Source**	**Probability**
☑	SimpleDesign	Extreme Programming (XP)	29,743%
☑	ConstantRefactoring	Extreme Programming (XP)	29,743%
☑	SystemMetaphor	Extreme Programming (XP)	29,743%
☐	PeerReviews	Capability Maturity Model Integration (CMMI)	05,385%
☐	ArchitectureTeam	James Coplien	05,385%
Insuffient/Inappropriate staffing			
	ProcessFragment	**Source**	**Probability**
☑	DevelopingInPairs	Extreme Programming (XP)	67,785%
☐	ApprenticeShip	James Coplien	10,738%
☐	DayCare	James Coplien	10,738%
☐	GenericsAndSpecifics	James Coplien	10,738%
Lack of user involvement			
	ProcessFragment	**Source**	**Probability**
☑	EarlyAndRegularDeliverXP	Extreme Programming (XP)	23,354%
☑	OnSiteCustomer	Extreme Programming (XP)	23,354%
☑	SprintPlanningMeeting	Scrum	23,354%
☐	EngageCustomer	James Coplien	12,974%
☐	EarlyAndRegularDeliverRUP	Rational Unified Process (RUP)	04,739%
☐	ImpliedRequirement	James Coplien	04,075%
☐	BuildPrototype	James Coplien	04,075%
☐	ScenariosDefineProblem	James Coplien	04,075%
Misunderstading the requirements			
	ProcessFragment	**Source**	**Probability**
☐	OnSiteCustomer	Extreme Programming (XP)	16,681%
☐	ConstantRefactoring	Extreme Programming (XP)	16,681%
☐	SimpleDesign	Extreme Programming (XP)	16,681%
☐	EarlyAndRegularDeliverXP	Extreme Programming (XP)	16,681%
☑	PlanningGame	Extreme Programming (XP)	16,103%
☐	SoftwareConfigurationManagement	Capability Maturity Model Integration (CMMI)	04,870%
☐	EarlyAndRegularDeliverRUP	Rational Unified Process (RUP)	03,445%
☐	ScenariosDefineProblem	James Coplien	02,953%
☐	ImpliedRequirement	James Coplien	02,953%
☐	BuildPrototype	James Coplien	02,953%

Figura 26 – Fragmentos recuperados e priorizados para o projeto "SisAcad"

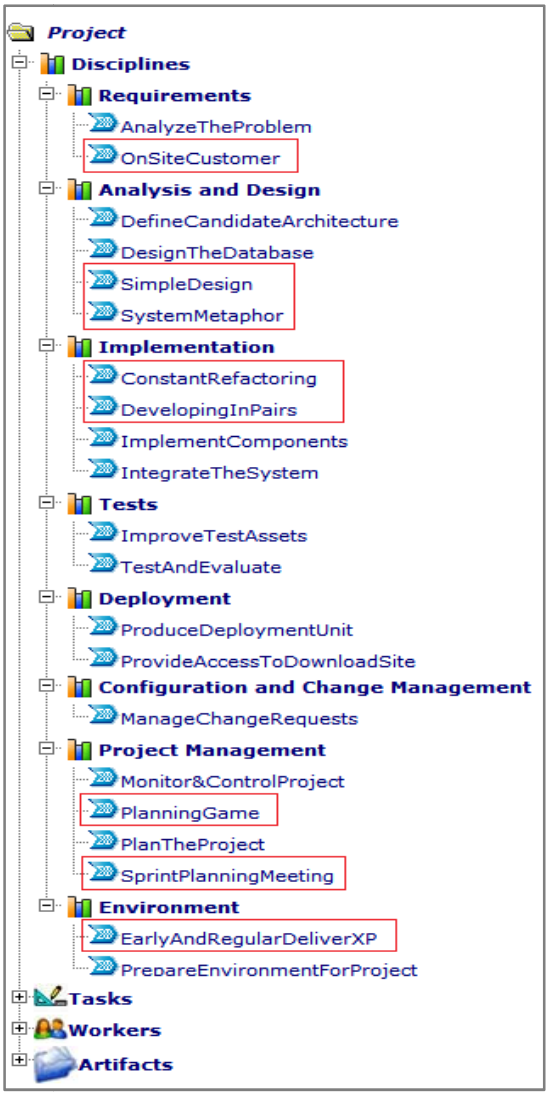


Figura 27 – Processo específico para o "SisAcad"

Sendo o "SisAcad" um projeto com contexto ágil segundo o modelo utilizado para contextualizar os projetos na abordagem proposta (*Octopus Model*), a ferramenta de apoio indica ao engenheiro de processo o uso de fragmentos de métodos de origem ágil e, portanto, mais adequados ao projeto. Assim, incluindo tais fragmentos no processo adaptado, este passa a possuir características ágeis desejáveis.

Devido à falta de clareza dos requisitos (risco "*Misunderstanding the requirements*"), foram incluídos os fragmentos "*PlanningGame*", "*OnSiteCustomer*", "*SprintPlanningMeeting*" e "*EarlyAndRegularDeliverXP*". Os três últimos tratam também da falta de envolvimento do cliente com o projeto (risco "*Lack of User Involvement*").

Considerando o risco "*Insufficient/inappropriate staffing*", foi incluído o fragmento "*DevelopingInPairs*" para tratar a falta de habilidade de membros da equipe.

Para prevenir o risco "*Infeasible Design*", relacionado a inviabilidades do projeto, foram incluídos os fragmentos "*SimpleDesign*", "*ConstatntRefactoring*" e "*SystemMetaphor*". O fragmento "*ConstantRefactoring*" também previne o risco "*Misunderstanding the requirements*".

O Apêndice A apresenta uma descrição sobre os fragmentos incluídos no processo específico do projeto "SisAcad", de acordo com Fontoura (2006).

8.3 Estudo de Caso 2 - SisEventos

A Organização X é proprietária de um software para empresas promotoras e administradoras de eventos como shows, teatros e cinemas, etc. Este é um sistema *desktop* que apenas reserva lugares nos eventos, de acordo com o desejo dos clientes no momento da compra, que deve ser feita em locais autorizados. Portanto, a compra de ingressos não pode ser feita via web pelos clientes. Além disso, a gestão dos negócios das empresas não é feita via sistema. Este será chamado de "SisEventos".

Neste contexto, a Organização X vislumbra evoluir o sistema de maneira a proporcionar uma maior comodidade às empresas promotoras de eventos e seus clientes compradores de ingressos, possibilitando a compra de bilhetes eletrônicos via web e meios para as empresas que utilizam o sistema para gerenciarem seus processos de negócio através do sistema.

O sistema será desenvolvido por equipes distribuídas, em diferentes escritórios da Organização X, levando a necessidade de regras formais para comunicação dos times. Além disso, não haverá envolvimento freqüente e direto dos usuários durante o desenvolvimento, gerando uma baixa taxa de mudanças

Sendo este um sistema crítico, devido ao fato de envolver transações financeiras e possível perda substancial de dinheiro, regras formais para o gerenciamento das equipes e do projeto são adotadas pela Organização X.

A Tabela 15 e a Tabela 16 ilustram situação específica do projeto "SisEventos". A Tabela 15 mostra os riscos identificados para o projeto, enquanto a Tabela 16 mostra a contextualização do projeto de acordo com o *Octopus Model.*

Tabela 15 – Riscos do projeto “SisEventos”

Risco	Categoria
Insufficient/inappropriate staffing	Planejamento
Lack of a methodology for the project	Planejamento
Lack of top management commitment to the project	Planejamento
Requirements instability	Requisitos
Non-realistic schedule and budget	Requisitos
Wrong development of functions or user interfaces	Execução

Tabela 16 – Contextualização do projeto “SisEventos”

Critério	Valor Estabelecido
Tamanho	Grande.
Criticidade	Perda de dinheiro.
Arquitetura estável	Modificada.
Modelo de negócio	Comercial.
Distribuição da equipe	Distribuição geográfica.
Taxa de mudanças **(% em um mês)**	Menos que 10.
Idade do sistema	Evolução de sistema legado.
Controle	Mecânico/formal.

A Figura 28 mostra os fragmentos recuperados de acordo com os riscos do projeto “SisEventos” e priorizados de acordo com o contexto do mesmo. Os fragmentos marcados com ✓ são selecionados pelo engenheiro de processo para serem incluídos no processo adaptado especifico para o projeto “SisEventos”.

Insuffient/Inappropriate staffing			
	ProcessFragment	**Source**	**Probability**
☑	ApprenticeShip	James Coplien	30,421%
☑	DayCare	James Coplien	30,421%
☐	GenericsAndSpecifics	James Coplien	30,421%
☐	DevelopingInPairs	Extreme Programming (XP)	08,737%
Lack of a methodology for the project			
	ProcessFragment	**Source**	**Probability**
☑	ProjectProcessIsDefined	Capability Maturity Model Integration (CMMI)	39,167%
☑	SoftwareLifeCycleIsDefined	Capability Maturity Model Integration (CMMI)	39,167%
☐	DefinedProcess	Extreme Programming (XP)	21.665%
Lack of top management commitment to the project			
	ProcessFragment	**Source**	**Probability**
☑	SeniorManagementReview	Capability Maturity Model Integration (CMMI)	66,992%
☐	SprintPlanningMeeting	Scrum	16,504%
☐	IterationPlanning	Extreme Programming (XP)	16,504%
Requirements instability			
	ProcessFragment	**Source**	**Probability**
☑	EarlyAndRegularDeliverRUP	Rational Unified Process (RUP)	16,746%
☑	ImpliedRequirement	James Coplien	16,254%
☑	BuildPrototype	James Coplien	16,254%
☑	ScenariosDefineProblem	James Coplien	16,254%
☑	SoftwareConfigurationManagement	Capability Maturity Model Integration (CMMI)	14,004%
☐	PlanningGame	Extreme Programming (XP)	04,491%
☐	EarlyAndRegularDeliverXP	Extreme Programming (XP)	03,999%
☐	OnSiteCustomer	Extreme Programming (XP)	03,999%
☐	ConstantRefactoring	Extreme Programming (XP)	03,999%
☐	SimpleDesign	Extreme Programming (XP)	03,999%
Non-realistic schedule and budget			
	ProcessFragment	**Source**	**Probability**
☑	DocumentedSoftwareEstimates	Capability Maturity Model Integration (CMMI)	35,502%
☑	SizeTheSchedule	James Coplien	24,173%
☐	WorkQueue	James Coplien	24,173%
☐	PlanningGame	Extreme Programming (XP)	08,435%
☐	SprintPlanningMeeting	Scrum	07,717%
Wrong development of functions or user interfaces			
	ProcessFragment	**Source**	**Probability**
☑	PeerReviews	Capability Maturity Model Integration (CMMI)	68,749%
☐	AcceptanceTests	Extreme Programming (XP)	15,625%
☐	DevelopingInPairs	Extreme Programming (XP)	15,625%

Figura 28 – Fragmentos recuperados e priorizados para o projeto “SisEventos”

Através da Figura 28 é possível identificar que os fragmentos sugeridos com maior probabilidade (“*Probability*”) são propostos por abordagens planejadas. Isto se dá, de acordo com a abordagem proposta nesta dissertação, porque tais fragmentos

possuem um contexto de uso semelhante ao contexto do projeto, o qual possui um contexto planejado. Os fragmentos com probabilidade mais baixa têm origem de abordagens ágeis e, portanto, é possível afirmar que são pouco aconselhados para serem inclusos no processo adaptado para o projeto em questão.

Quanto maior a probabilidade, mais adequado é o fragmento para o projeto em questão, em outras palavras, mais indicado é o seu uso ou sua inclusão no processo adaptado específico para o projeto.

A Figura 29 mostra os fragmentos contidos no processo adaptado específico para o projeto "SisEventos". É possível observar que além dos fragmentos pertencentes ao processo padrão da organização (Tabela 12), também fazem parte do processo adaptado os fragmentos selecionados pelo engenheiro de processo, após a recuperação e priorização dos fragmentos indicados pela ferramenta de apoio (Figura 28). Os fragmentos destacados são aqueles específicos do processo do projeto, os demais são os descritos pelo processo padrão da organização.

Sendo o "SisEventos" um projeto com contexto planejado segundo o modelo utilizado para contextualizar os projetos na abordagem proposta (*Octopus Model*), a ferramenta de apoio indica ao gerente de processo o uso de fragmentos de métodos de origem planejada e, portanto, mais adequados ao projeto. Assim, incluindo tais fragmentos no processo adaptado, este passa a possuir as características planejadas desejáveis.

Considerando o risco "*Insufficient/inappropriate staffing*", foram incluídos os fragmentos "*DayCare*" e "*ApprenticeShip*" para tratar deficiências da equipe.

Para prevenir o risco "*Lack of a methodology for the project*" foram incluídos os fragmentos "*ProjectProcessIsDefined*" e "*SoftwareLifeCycleIsDefined*".

Para tratar a falta de envolvimento da alta gerencia com o projeto (risco "*Lack of top management commitment to the project*"), foi incluído o fragmento "*SeniorManagementReview*".

Devido à instabilidade dos requisitos identificada para o projeto (risco "*Misunderstanding the requirements*"), foram incluídos os fragmentos "*EarlyAndRegularDeliverRup*", "*ImpliedRequirement*", "*BuildPrototype*" e "*ScenariosDefineProblem*" e "*SoftwareConfigurationManagement*".

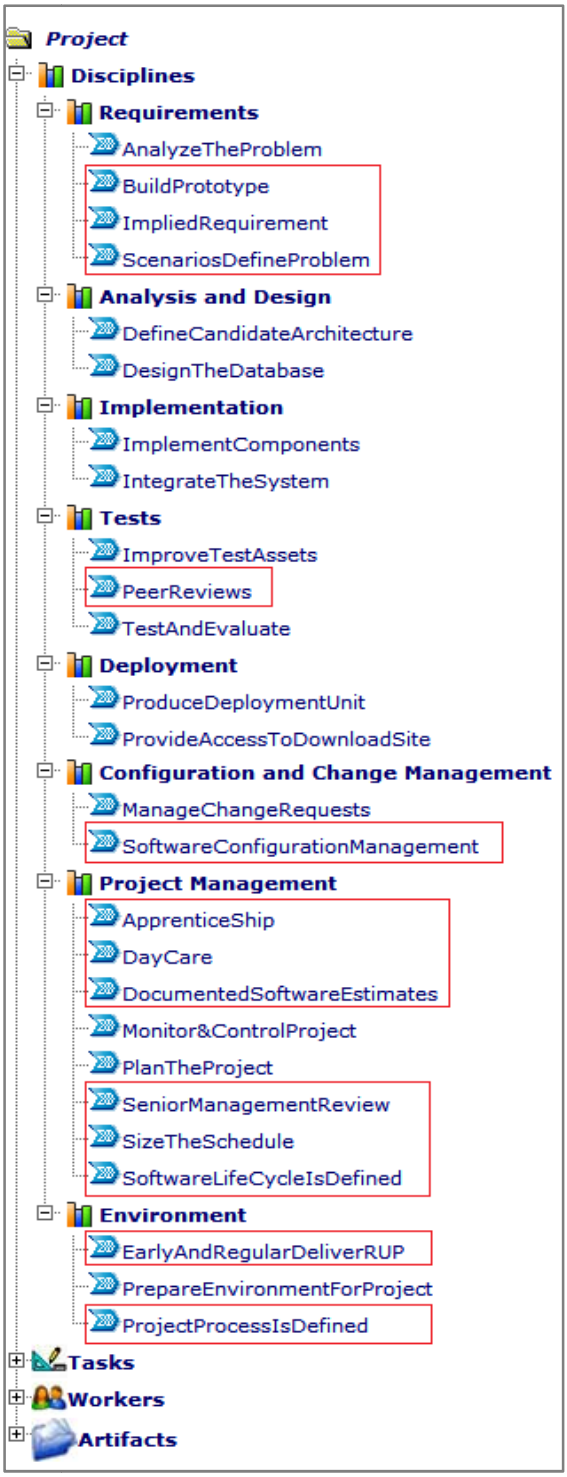


Figura 29 – Processo específico para o projeto “SisEventos”

Para tratar questões relacionadas a orçamento e cronograma do projeto (risco “*Non-realistic Schedule and budget*”), foram incluídos os fragmentos “*DocumentedSoftwareEstimates*” e “*SizeTheSchedule*”.

O fragmento "*PeerReviews*" foi incluído para prevenir o risco "*Wrong development of functions or user interfaces*", relacionado ao mau desenvolvimento de funções ou interfaces do sistema.

O Apêndice A apresenta uma descrição sobre os fragmentos incluídos no processo específico do projeto "SisEventos", de acordo com Fontoura (2006).

8.4 Análise da validação

Os estudos de caso foram realizados com dois projetos de características bastante distintas, gerando processos específicos também distintos. Em ambos os casos, primeiramente, os fragmentos apropriados para prevenir os riscos do projeto foram recuperados da base de métodos ou base de conhecimento da organização.

No projeto para desenvolvimento do sistema "SisAcad", com contexto ágil de acordo com os fatores do *Octopus Model* e, portanto, com características que propiciam o uso de metodologias ágeis, os fragmentos de mais alta relevância sugeridos pela abordagem OSPTA são fragmentos que visam proporcionar maior agilidade ao processo padrão da organização, baseado no RUP.

No desenvolvimento do sistema "SisEventos", o projeto apresenta um contexto planejado de acordo com os fatores do *Octopus Model*. Assim, os fragmentos de mais alta relevância sugeridos pela abordagem OSPTA são fragmentos que visam proporcionar mais planejamento e documentação ao processo de software.

Conclui-se que o mecanismo de seleção e priorização dos fragmentos não elimina o papel do engenheiro de processos, apenas facilita a sua atividade, identificando os fragmentos de mais alta relevância, de acordo com os riscos e contexto do projeto.

A base de conhecimento utilizada nesta dissertação é uma sugestão. A organização deve adaptar tal base de acordo com suas necessidades e experiências com projetos passados para aperfeiçoar a eficácia do mecanismo de seleção e priorização de fragmentos de métodos.

A abordagem proposta não trata o fluxo de execução dos fragmentos no processo específico, assim, a tarefa de ajustar o processo definido cabe ao projetista. Os fragmentos são inseridos no processo adaptado de acordo com a disciplina do mesmo.

9. Considerações finais e Trabalhos futuros

A literatura mostra que projetos de software possuem características específicas, sendo distintos em relação a diversos fatores que afetam o desenvolvimento de software. Assim, cada projeto de software requer estratégias particulares de desenvolvimento. Métodos/processos de desenvolvimento devem ser configurados de acordo com o contexto dos projetos, situações específicas dos projetos de software.

Também é possível observar que a adaptação de processos é uma tarefa que requer alto grau de conhecimento e frequentemente é executada de maneira *ad-hoc,* com base apenas nas experiências dos engenheiros de processo, e sem levar em conta características específicas do projeto.

Este trabalho propõe uma abordagem sistemática para construção de processos de software de software especifico para o projeto, através da adaptação de processos, chamada OSPTA – *Octopus SME Process Tailoring Approach.* Esta é uma abordagem multicritérios, baseada em conceitos de *Situational Method Engineering* (SME).

A abordagem proposta considera um ou mais critérios de adaptação definidos pelo engenheiro de processo e mais oito fatores que caracterizam o contexto de um projeto de software, definidos pelo *Octopus Model* proposto por Kruchten (2010). Juntos, todos estes critérios definem as características situacionais do projeto.

A abordagem OSPTA é genérica, pois pode ser customizada para considerar diferentes critérios de adaptação de acordo com as necessidades da organização. Por exemplo, o critério de adaptação pode estar relacionado a riscos do projeto, para que se obtenha um processo adaptado que inclua atividades com capacidade para prevenir os riscos identificados para o projeto. Por outro lado, o critério de adaptação pode estar relacionado a metas de qualidade, para que seja construído um processo de software com atividades capazes de contribuir para garantia que tais metas serão alcançadas.

Em OSPTA a construção de processos específicos para um projeto acontece através da adaptação do processo padrão da organização (PSPO), incluindo fragmentos adequados ao contexto do projeto. Seguindo conceitos da abordagem SME, a construção se dá a partir de fragmentos de métodos armazenados em um repositório

chamado base de métodos, levando em conta as características situacionais, ou seja, o contexto do projeto.

Os fragmentos de métodos utilizados neste trabalho são baseados nos conceitos de *method fragments* (Henderson-Sellers, 2008), pois, entre outras razões, segundo Hederson-Sellers são mais adequados a situações reais, de acordo com experiências com abordagens baseadas SME na indústria.

Este trabalho propõe uma definição para fragmentos de métodos para serem utilizados na abordagem proposta. Um metamodelo para a definição destes fragmentos também é proposta nesta dissertação.

O metamodelo proposto, chamado M^2F, desenvolvido através de uma integração do metamodelo do RUP (BENCOMO, 2005) e da ISO/IEC 24744 (ISO/IEC 2007) seguindo regras de integração propostas por Batini (1992), apresenta uma estrutura de simples entendimento e nomenclatura bem conhecida.

O M^2F representa os elementos de processos necessários para a definição dos fragmentos de métodos, garante a integridade das relações entre tais elementos. Além disso, provê meios para auxiliar a recuperação e adaptação dos fragmentos em abordagens SME, principalmente através da classe "*TailoringGuide*" a qual inclui o(s) critério(s) de adaptação associados ao fragmento e o contexto de uso do mesmo.

Além de simples, o metamodelo proposto é flexível no sentido de que através dele podem ser definidos fragmentos de métodos oriundos de diferentes fontes, por exemplo, modelos de processo, modelos de referencia, padrões de processo, boas práticas, bem como a combinação de elementos de origens distintas. Além disso, é possível definir fragmentos de métodos com contextos ágeis e planejados. Em outras palavras, através deste metamodelo, a organização pode definir fragmentos e armazená-los de acordo com as suas necessidades. O metamodelo em si também pode ser adaptado de acordo com as necessidades e experiências da organização, por exemplo com a inclusão de outros elementos de processo para a definição dos fragmentos.

Através do *Octopus Model* e do conceito do *agile sweet spot,* é possível definir contextos com características de abordagens ágeis e planejadas, tanto para o contexto de uso dos fragmentos quanto para o contexto do projeto.

A abordagem OSPTA utiliza uma abordagem multicritérios para seleção e priorização de fragmentos de métodos. Em um primeiro momento considerando apenas o(s) critério(s) de adaptação definidos pelo engenheiro de processo. Após, utilizando os

oito fatores do *Octopus Model* como critérios para priorizar os fragmentos em ordem de relevância em relação ao contexto do projeto.

Este mecanismo de priorização é baseado em um modelo matemático de apoio a tomada de decisões: AHP (*Analytical Hierarchy Process*). Por se tratar de um mecanismo baseado em métodos matemáticos, trata-se de um mecanismo sistemático, e evitando a seleção de fragmentos *ad-hoc,* baseada apenas nas experiências do engenheiro de processos.

Todos os fragmentos selecionados são comparados, critério por critério, com o contexto do projeto e, através desta priorização, quanto mais o contexto de uso do fragmento se encaixar ao contexto do projeto, mais indicado ele será. Assim, a abordagem OSPTA garante que em nenhuma hipótese as características situacionais ou contextos do projeto deixem de ser considerados em uma atividade de adaptação de processos.

Foi desenvolvida uma ferramenta de apoio a OSPTA. Através desta, após ser definida uma base de métodos, a atividade de criação torna-se menos demorada e dispendiosa para a organização.

A validação da abordagem proposta foi desenvolvida em um ambiente hipotético de gerenciamento de riscos em projetos de software, sendo “Riscos do Projeto” o critério de adaptação. Em cada um dos estudos de casos, a ferramenta propôs para o engenheiro de processo a inclusão no PSPO dos fragmentos mais apropriados em relação ao contexto do projeto, para cada risco do projeto identificado. Assim, levando em conta um ou mais critérios de adaptação somados ao contexto do projeto definido pelo *Octopus Model*, pode-se afirmar que o processo é adaptado de acordo com as características situacionais do projeto.

Concluiu-se, ainda, que a partir de um PSPO é possível criar processos específicos para o projeto com base em diferentes critérios de adaptação e para diferentes contextos.

9.1 Contribuições

Esta dissertação apresenta como principal contribuição a proposta de uma abordagem sistemática para a construção de um modelo de processo de software específico para um projeto de acordo com as suas características, através de uma análise multicritérios para seleção e priorização de fragmentos de métodos armazenados em um repositório, os quais podem ser unidos ao PSPO para formar o PEP.

Outras contribuições do trabalho incluem:

- Um metamodelo para definição de fragmentos de métodos, o qual permite a instanciação dos fragmentos de métodos de natureza ágil e planejada, bem como representa os conceitos e relacionamentos dos elementos de processo relacionados aos fragmentos;
- Associação de fragmentos de métodos com diferentes critérios de adaptação e contextualização dos mesmos através do *Octopus Model;*
- Utilização do *Octopus Model* para caracterização de situações específicas em um projeto de software, para uso na abordagem de adaptação de processos baseada em SME;
- Uma base de conhecimento inicial, como sugestão, com fragmentos de métodos associados aos fatores do *Octopus Model,* indicando o contexto ou situação adequada para que tais fragmentos sejam utilizados;
- Um mecanismo multicritérios para seleção e priorização de fragmentos de métodos, utilizando o modelo matemático AHP, para priorização dos fragmentos de acordo com o contexto do projeto.

9.2 Trabalhos Futuros

Como trabalhos futuros a este, propõe-se:

- A melhoria da base de conhecimento existente, com novos fragmentos, relacionando os fragmentos a outros critérios de adaptação, além de riscos do projeto;

- Enriquecer os critérios de comparação para contextualização de projetos e contexto de uso de fragmentos, usados nesta abordagem com o estudo de mais critérios que afetam o desenvolvimento de software, e de que maneira afetam;
- Explorar fragmentos com contexto de uso híbrido, com intuito de mesclar práticas ágeis e planejadas, quando for adequado;
- Definir um mecanismo para que o processo padrão da organização também se ajuste ao contexto do projeto, mais ágil ou mais planejado, através de práticas equivalentes;
- A abordagem proposta não trata o fluxo de execução dos fragmentos no processo específico, assim, a tarefa de ajustar o processo definido cabe ao projetista. Portanto, a evolução da abordagem OSPTA para tratar o aspecto dinâmico, fluxo do processo adaptado, é apontada como um relevante trabalho futuro a este.

9.3 Publicações

- **IADIS Applied Computing**: Pereira, G. e Fontoura, L. "*An Approach to Risk Prevention Based on Situational Method Engineering*" Em: IADIS International Conference Applied Computing. Rio de Janeiro, Brasil, 2011.
- **XV Congresso Ibero-Americano em Engenharia de Software (CibSE):** Pereira, G. e Fontoura, L. "*A Risk Management Approach Based on Situational Method Engineering*" Em: XV Congresso Ibero-Americano em Engenharia de Software. Buenos Aires, Argentina, 2012.
- **Simpósio Brasileiro de Sistemas de Informação (SBSI):** Pereira, G. e Fontoura, L. "*Defining Agile and Planned Method Fragments for Situational Method Engineering*" Em: VIII Simpósio Brasileiro de Sistemas de Informação. Fortaleza, Brasil, 2012.

Referências

ABAD, Z. S. H., SADI M. H. AND RAMSIN, R. **Towards Tool Support for Situational Engineering of Agile Methodologies**. In: Proceedings of the 2010 Asia Pacific Software Engineering Conference (APSEC '10), IEEE Computer Society, USA, p. 326-335, 2010.

AGERFALK, P. J. et al. **Modularization Constructs in Method Engineering Towards Common Ground**. In: Ralyté J. et al. (eds.) Situational Method Engineering Fundamentals and Experiences, p. 359-368. Boston: Springer, 2007

AHARONI, A.; REINHARTZ-BERGER, I. **A Domain Engineering Approach for Situational Method Engineering**. In: Proceedings of the 27th International Conference on Software and System Process (ICSSP '11), USA**,** p. 43-52, 2008.

ALEGRÍA, J. A. H. et al. **An MDE approach to software process tailoring**. In: Proceedings of the 2011 International Conference on Software and Systems Process, p. 43-52, 2011.

AMBLER, S. W. **Agile Modeling: Best Practices for the Unified Process and Extreme Programming**. New York: John Wiley & Sons, 2002.

BATINI, C.; CERI, S.; NAVATHE, S. B. Conceptual Database Design Na ENtity-Relationship Approach. USA: The Benjamin/Cummings Publications, 1992.

BECK, K. **Programação Extrema (XP) Explicada: Acolha as Mudanças**. Ed. Bookman: Porto Alegre. 2004.

BENCOMO, **A. Extending the RUP, Part 1: Process Modeling**. IBM, 2005. Disponível em: <http://www-128.ibm.com/developerworks/rational/library/05/r-3320/>. Acesso em: out. 2010.

BOEHM, B. Get Ready for Agile Methods, with Care. **Computer**, New York, v.35, n.1, p. 64-69, 2002

BOEHM, B.; TURNER, R. Using Risk to Balance Agile and Plan-Driven Methods. **Computer**, New York, v. 36, n. 6, p. 39-43, 2003.

BÖRNER, R. et al. **Fragment Selection and Context Factors in Situational Methods for Service Identification**. In: Proceedings of the Fifth International Conference on Research Challenges in Information Science (RCIS), France, p. 01-08, 2011.

BUCHER, T. et al. **Situational Method Engineering – On the Differentiation of "Context" and "Project Type"**. In: Ralyté J. et al. (eds.) Situational Method Engineering Fundamentals and Experiences, p. 33-48. Boston: Springer, 2007.

COAD, P. et al. **Java Modelling in Color**. Englewood Cliffs: Prentice Hall,1999.

COCKBURN, A. **Crystal Clear: A Human-Powered Software Development Methodology for Small Teams**. Reading: Addison-Wesley, 2001.

CONBOY, K. Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. **Information Systems Research**, v. 20, n. 3, p. 329-354, 2009.

COPLIEN, J. O. **Sofware Patterns**. New York: SIGS Books and Multimedia, 1996.

Dai, F.; Li, T. **Tailoring Software Evolution Process**. In: Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007, v. 2, p. 782-787, 2007.

FONTOURA, M. L. "***PRiMA: Project Risk Management Approach***". Tese (Doutorado em Ciência da Computação). Universidade Federal do Rio Grande do Sul – UFRGS. Porto Alegre, Brasil. 2006.

FOWLER, M. **The New Methodology**. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html>. Acesso em: ago. 2010.

FUGGETA, A. **Software Process: A Roadmap**. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 22., 2000, New York. **Proceedings...** New York: ACM Press, 2000.

GERICKE, A. et al. **Situational Method Engineering for Governance, Risk and Compliance Information Systems**. In: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology (DESRIST '09). ACM, New York, 2009.

GINSBERG, M.; P.; QUINN, L. H. **Process Tailoring and the Software Capability Maturity Model**. [S.l.]: Software Engineering Institute, Carnegie Mellon University, 1995. (CMU/SEI-94-TR-024).

GONZALEZ-PEREZ, C. **Supporting Situational Method Engineering with ISO/IEC 24744 and the Work Product Approach**. In: Ralyté J. et al. (eds.) Situational Method Engineering Fundamentals and Experiences, p. 33-48. Boston: Springer, 2007

HENDERSON-SELLERS, B. Object-Oriented Methods and Processes. In: Proceedings of the Software Methods and Tools Conference, Wollongong: University of Wollongong, 2000.

HENDERSON-SELLERS, B.; GONZALEZ-PEREZ, C., RALYTÉ, J. **Comparison of Method Chunks and Method Fragments for Situational Method Engineering**. In: Proceedings of the 19th Australian Conference on Software Engineering, Washington, USA, p. 479-488, 2008

HENDERSON-SELLERS, B.; RALYTÉ, J. Situational Method Engineering: State-of-the-Art Review. **Journal of Universal Computer Science.** V. 16, n. 3, p. 424-478, 2010.

HULL, M.E.C. et al. Software Development Processes - an Introduction. **Information and Software Technology**, Amsterdam, v. 44, n. 1, p. 1-12, 2002.

HUMPHREY, W. S. **Managing the software process**. Massachusetts: Addison-Wesley, 1990.

IBM Corporation.**IBM Rational Unified Process v7.0**. 2007.

ISO/IEC. **ISO/IEC 24744:2007**. Software Engineering – Metamodel for Development Methodologies. 2007.

ISO/IEC. **ISO/IEC 12207:2008**. Systems and software engineering -- Software life cycle processes. 2008.

KAMEL, M.; BEDIWI, I.; AL-RASHOUD, M. Planned Methodologies vs. Agile Methodologies under the Pressure of Dynamic Market. **JKAU: Eng. Sci**, v. 21, n. 1, p. 19-35, 2010.

KORNYSHOVA, E.; DENECKÈRE, R.; SALINESI, C. **Method Chunks Selection by Multicriteria Techniques: An Extension of the Assembly-based Approach**. In: Ralyté J. et al. (eds.) Situational Method Engineering Fundamentals and Experiences, p. 64-78. Boston: Springer, 2007.

KRUCHTEN, P. **Contextualizing Agile Software Development**. In: Proceedings of the EuroSPI Conference,p. 1-12, 2010.

LAPLANTE, P. A.; Neill, C. J. Opinion: The Demise of the Waterfall Model is Imminent. **ACM Queue**, 1(10), p. 10-15, 2004.

LYCETT, M. et al. Migrating Agile Methods to Standardized Development Practice. **Computer**, New York, v. 36, n. 6, p.79-85, 2003.

MAGDALENO, A.M. **An Optimization-based Approach to Software Development Process Tailoring**. In: Second International Symposium on Search Based Software Engineering (SSBSE), p. 40-43, 2010.

NICHOLS, R.; CONNAUGHTON, C. **Software Process Improvement Journey:** IBM Australia Application Management Services. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 2005. (CMU/SEI-2005-TR-02).

OBJECT MANAGEMENT GROUP (OMG). **Software & System Process Engineering Metamodel Specification**, Version 2.0. Massachusetts: Object Management Group, Inc. 2008.

PEDREIRA, O. et al. A Systematic review of software process tailoring. In: **SIGSOFT Softw. Eng. Notes**, v.32, n. 3, p. 1-6, 2007.

PEREIRA, G.; FONTOURA, L. **An Approach to Risk Prevention Based on Situational Method Engineering**. In: Proceedings of the IADIS International Conference Applied Computing. Rio de Janeiro, Brasil, 2011.

PEREIRA, G.; FONTOURA, L. **A Risk Management Approach Based on Situational Method Engineering**. In: Proceedings of the XV XV Ibero-American Conference on Software Engineering (CIBSE). Buenos Aires, Argentina, 2012a

PEREIRA, G.; FONTOURA, L. **Defining Agile and Planned Method Fragments for Situational Method Engineering**. Em: VIII Simpósio Brasileiro de Sistemas de Informação. Fortaleza, Brasil, 2012b.

POPPENDIECK, M. Lean programming. **Software Development Magazine**. p. 71–75, 2001.

PUVIANI, M.; CABRI, G.; LEONARDI, L. **Experiences in Applying Situational Method Engineering in AOSE**. In: Proceedings of the 35th Eutomicro Conference on Software Engineering and Advanced Applications (SEAA '09), Washington, USA, p. 353-359, 2009.

RATIONAL SOFTWARE CORPORATION. **Rational Unified Process**: Version 2003.06.12. Cupertino, 2003.

RUI H.; HAO W.; ZHIQING L. **A Software Process Tailoring Approach Using a Unified Lifecycle Template**. In: Computational Intelligence and Software Engineering. Beijing, China, p. 1-7, 2009.

STAPLETON, J. **DSDM: Dynamic Systems Development Method**. Harlow: Addison-Wesley, 1997.

SAATY, T. L. **The Analytic Hierarchy Process**. New York: McGraw-Hill International, 1980.

SAATY, T. L. **Theory and Applications of the Analytic Network Process: Decision Making with Benefits, Opportunities, Costs, and Risks**. Pittsburgh: RWS Publications, 2005.

SCHWABER, K.; BEEDLE, M. **Agile Software Development with Scrum**. Upper Saddle River: Prentice Hall, 2002.

SHUJA, A., KREBS, J. **RUP Reference and Certification Guide: Solution Designer (RUP).** USA: IBM Pres, 2007.

SOFTWARE ENGINEERING INSTITUTE. **Capability Maturity Model Integration:** (CMMI-DEV), Version 1.3, Continuous Representation. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 2010. (CMU/SEI-2010-TR-033).

SOMMERVILLE I. **Engenharia de software**. São Paulo: Pearson Addison – Wesley, 2006.

SONIA, S.S.; BEN GHEZALA, H.; KRAIEM, N. **Toward Web oriented Situational Methods**. In: Proceedings of the IEEE International Conference on Information Reuse and Integration, pp.30-35, 2007.

VARGAS, R**. Using the Analytic Hierarchy Process (AHP) to Select and Prioritize Projects in a Portifolio**. In: PMI Global Congress, Washington, USA, 2010.

XU, P. **Knowledge Support in Software Process Tailoring**. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, Washington, USA, p. 87c., 2005.

XU, P.; RAMESH, B. Using Process Tailoring to Manage Software Development Challenges. **IT Professional**, v.10, n.4, p.39-45, 2008.

APÊNDICE A – Descrição de Fragmentos de Métodos

A tabela Apêndice A. Tabela 1 e a tabela Apêndice A. Tabela 2 apresentam uma breve descrição, segundo Fontoura (2006), sobre os fragmentos incluídos nos processos específicos dos projetos "SisAcad" e "SisEventos", respectivamente.

Apêndice A. Tabela 1 – Descrição dos fragmentos incluídos no processo específico para o projeto "SisAcad".

Fragmento	Descrição
OnSiteCustomer	O cliente deve estar disponível quando os desenvolvedores precisam solucionar dúvidas, definir prioridades, e resolver pendências. Se possível, o cliente deve trabalhar na mesma sala dos desenvolvedores.
SimpleDesign	Implemente os requisitos atuais da forma mais simples possível. XP apresenta diretrizes com essa finalidade, como: considere a implementação mais simples que pode funcionar, implemente somente o código necessário para suportar os requisitos.
SystemMetaphor	A arquitetura do sistema é descrita como uma metáfora. Uma metáfora é uma história compartilhada entre a equipe de como o sistema funciona.
ConstantRefactoring	Melhorar continuamente a estrutura do código sem modificar a funcionalidade.
DevelopingInPairs	Todo o código de produção deve ser desenvolvido por duas pessoas trabalhando juntas no mesmo computador. Enquanto um dos programadores digita o código o outro pensa nas funcionalidades, fica atento a *bugs* e cuida para que a codificação esteja no rumo certo.
PlanningGame	O objetivo dessa prática é determinar o escopo da iteração seguinte. Os usuários decidem escopo e prioridades do próximo release. A equipe técnica, por outro lado, decide estimativas, conseqüências técnicas, processo de desenvolvimento e detalha cronograma.
SprintPlanningMeeting	Em processos Scrum ocorre uma reunião de planejamento a cada 30 dias, quando se inicia um novo *Sprint*. Nessa reunião se reúnem os clientes, os usuários, o gerenciamento, o dono do produto e a equipe Scrum para determinar os objetivos e as funcionalidades do próximo *Sprint.*
EarlyAndRegularDeliverXP	Desenvolvimento incremental, onde versões do software são entregues freqüentemente ao cliente.

Apêndice A. Tabela 2 – Descrição dos fragmentos incluídos no processo específico para o projeto “SisEventos”.

Fragmento	Descrição
BuildPrototype	Elaborar e validar um protótipo do sistema junto ao cliente para entender, validar e explorar o custo e benefícios de decisões de projeto, visando definir os requisitos do sistema.
ImpliedRequirement	Conjuntos de funcionalidades relacionadas que representam os requisitos devem ser nomeados.
ScenariosDefineProblem	Os requisitos funcionais devem ser capturados como casos de uso.
PeerReviews	Executar revisões por pares, com um exame metodológico de produtos de trabalho de software para identificar defeitos e áreas onde mudanças são necessárias.
ApprenticeShip	Treine os novos contratados, por meio de um programa de treinamento, para estes se tornarem especialistas.
DayCare	Quando os especialistas da equipe estão gastando muito tempo auxiliando os novatos; dedique um dos especialistas para auxiliar os novatos, e deixe os outros desenvolverem o sistema.
DocumentedSoftwareEstimates	Estimativas de tamanho, custo, tempo, recursos críticos e fundos devem ser documentadas em uma base histórica.
SeniorManagementReview	O gerente sênior deve participar das reuniões de acompanhamento nos marcos dos projetos e ser informado sobre a execução das atividades dos projetos periodicamente.
SizeTheSchedule	O trabalho a ser realizado é definido e o tamanho do projeto é estimado. Negocie o cronograma do projeto com a equipe de desenvolvimento.
SoftwareLifeCycleIsDefined	Definir um ciclo de vida com estágios pré-definidos e de tamanho gerenciável para o projeto.
EarlyAndRegularDeliverRup	Desenvolvimento incremental, onde versões do software são entregues freqüentemente ao cliente.
ProjectProcessIsDefined	Para cada projeto deve existir uma definição operacional do processo que será usado no projeto. O processo de software definido do projeto é descrito em termos de padrões, procedimentos, procedimentos, ferramentas e métodos.