# Relatório Técnico

# A Criação de um Exemplo para Alimentar a Abordagem COMPOOTIM de Composição e Otimização de Processos de Software

Andréa Magalhães Magdaleno
(andrea@cos.ufrj.br)

Cláudia Maria Lima Werner
(werner@cos.ufrj.br)

Renata Mendes de Araujo
(renata.araujo@uniriotec.br)

PESC
Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ

Rio de Janeiro, Dezembro de 2011

# A Criação de um Exemplo para Alimentar a Abordagem COMPOOTIM de Composição e Otimização de Processos de Software

Andréa Magalhães Magdaleno[1]
Cláudia Maria Lima Werner[1]
Renata Mendes de Araujo[2]

[1]Programa de Engenharia de Sistemas e Computação (PESC) – COPPE/UFRJ
Caixa Postal 68.511 – 21945-970 – Rio de Janeiro – RJ – Brasil

[2]Programa de Pós Graduação em Informática (PPGI) – Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec) – Universidade Federal do Estado do Rio de Janeiro (UNIRIO) – 22290-240 – Rio de Janeiro – RJ – Brasil

{andrea, werner}@cos.ufrj.br, renata.araujo@uniriotec.br

## RESUMO

As organizações de software buscam continuamente melhorar a qualidade dos seus produtos de software para atender às necessidades dos seus clientes. Esta busca pela melhoria da qualidade tem motivado muitas organizações a investirem na definição de processos de software específicos para os projetos. Porém, as iniciativas de definição de processos de software geralmente são desafiadas pela diversidade de projetos, modelos de desenvolvimento, processos e pessoas existentes. Estes fatores contribuem para tornar a definição de processos uma atividade complexa, demorada e sujeita a erros. Tal atividade pode ser facilitada por uma estratégia de reutilização de processos. Esta estratégia pode ser implementada através de uma abordagem de Linha de Processos Baseada em Contexto (LPBC). Com base nos dados desta abordagem e seguindo a sua sistemática, este trabalho apresenta uma proposta para a composição e otimização de processos de software (COMPOOTIM). O objetivo da COMPOOTIM é apoiar o gerente de projeto na tomada de decisão sobre a composição de um processo de software específico para o projeto. A composição dos componentes do processo é automatizada e usa uma técnica de otimização para sugerir processos com base no contexto do projeto. O foco deste trabalho é apresentar um exemplo de criação de uma LPBC que gera os dados necessários para o funcionamento da COMPOOTIM.

# ABSTRACT

Software organizations continuously seek to improve the quality of their software products to meet the needs of customers. This search for quality improvement has motivated many organizations to invest in the definition of software processes specific to projects. However, the definition of software processes is usually challenged by diversity of projects, development models, processes, and people. These factors contribute to make process definition a complex, time consuming and error prone activity. Such activity can be facilitated by a process reuse strategy. This strategy can be implemented through a context-based process line approach. Based on the data from this approach and following its systematic, this work presents a proposal to software process composition and optimization (COMPOOTIM). The purpose of COMPOOTIM is to support the project manager's decisions about the composition of a software process specific to a project. Process components composition is automated and uses an optimization technique to suggest processes based on the context of a particular project. The aim of this work is to present an example of creating a LPBC that generate data necessary for COMPOOTIM operation.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# 1.Introdução

As organizações de software são continuamente desafiadas pela necessidade de melhorar a qualidade dos produtos de software gerados e atender às necessidades dos seus clientes (HERBSLEB *et al.*, 2005). Neste cenário, a premissa de que a qualidade do produto de software depende do processo de desenvolvimento adotado para construí-lo e mantê-lo (CHRISSIS *et al.*, 2006, CUGOLA e GHEZZI, 1998, FUGGETTA, 2000, OSTERWEIL, 1987, PAULK, 2009, PRESSMAN, 2001, RAMAN, 2000), indica que muitos dos problemas associados ao desenvolvimento de produtos de software podem ser resolvidos aperfeiçoando-se este processo.

Esta busca pela melhoria da qualidade tem motivado muitas organizações a investirem na definição de seus processos de software[1]. A razão para se definir processos de software é melhorar a forma pela qual o trabalho de desenvolvimento do software é realizado. Ao pensar no processo de forma organizada, é possível antecipar problemas e antever maneiras de prevenir ou resolvê-los (HUMPHREY, 1989).

A definição de processos de software pode ser feita em diferentes níveis de abstração. Primeiro, um processo de software padrão pode ser definido para a organização. Baseado nesse processo organizacional, processos padrão especializados podem ser definidos considerando, por exemplo, paradigma, tecnologia ou domínio de aplicação específicos. Finalmente, processos específicos de projetos podem ser definidos a partir de processos padrão (especializados ou não) (MACHADO, 2000, ROCHA *et al.*, 2001). Este trabalho foca especificamente na definição de processos de software específicos para projetos (que será tratada daqui em diante simplesmente como definição de processos).

Dentro das organizações, as iniciativas de definição de processos de software específicos para projetos são geralmente desafiadas pela **diversidade** de projetos, modelos de desenvolvimento, processos e pessoas existentes.

A **diversidade de projetos** faz com que os contextos onde os processos serão utilizados sejam muito diversificados. Tendo em mente que as organizações são diferentes, e ainda que dois projetos dentro da mesma organização também podem ser diferentes, não existe um processo de software, por mais bem definido que seja, que possa ser genericamente aplicado a todos os projetos. Dependendo das características do projeto, um processo aplicado com sucesso em um projeto pode ser um fracasso em outro (BERGER, 2003, CUGOLA e GHEZZI, 1998, HENDERSON-SELLERS, 2002, MACHADO, 2000, MARTÍNEZ-RUIZ *et al.*, 2008,

[1] Um processo de software pode ser definido como "um conjunto de atividades, métodos, práticas e transformações que as pessoas usam para desenvolver e manter software e seus produtos associados" (PAULK, 2009).

PEDREIRA *et al.*, 2007). Isso torna clara a seguinte afirmação de Humphrey (1989): "Assim como não existem dois projetos de software idênticos, também não existem dois processos de software idênticos no mundo".

Como o universo dos projetos de software é extenso, um único modelo de desenvolvimento[2] não consegue atender aos requisitos de todos os projetos, pois eles têm objetivos, características e necessidades variados (FEI DAI e TONG LI, 2007, MACHADO, 2000). Esta **diversidade de modelos de desenvolvimento** também traz desafios para a definição de processos de software. Os modelos de desenvolvimento de software orientado ao planejamento[3], ágil (BECK *et al.*, 2001) e livre (RAYMOND, 2001) têm o mesmo objetivo: melhorar o desenvolvimento de software; mas adotam enfoques distintos. Enquanto no desenvolvimento orientado ao planejamento busca-se previsibilidade, estabilidade e confiabilidade (CHRISSIS *et al.*, 2006), o desenvolvimento ágil tenta agregar valor ao negócio rapidamente e se adaptar às mudanças de mercado, tecnologia e ambiente (COCKBURN, 2001). Por outro lado, no desenvolvimento de software livre, o objetivo principal é garantir as liberdades básicas dos usuários e desenvolvedores para executar, estudar, adaptar, melhorar e distribuir o código do programa (FSF, 2008). Estes três modelos tiveram, na última década, um enorme impacto e a sua perspectiva de evolução futura é igualmente promissora (EBERT, 2007, TAURION, 2004, THEUNISSEN *et al.*, 2008). Como cada um representa um universo de desenvolvimento com características muito singulares, a pesquisa na área tem discutido como conciliar as particularidades de cada modelo para a definição de processos de software mais eficazes (BARNETT, 2004, BOEHM e TURNER, 2003, GLASS, 2001, GLAZER *et al.*, 2008, MAGDALENO *et al.*, 2011, PAULK, 2001, TURK *et al.*, 2002, TURNER e JAIN, 2002, WARSTA e ABRAHAMSSON, 2003).

Por sua vez, a **diversidade de processos** (LINDVALL e RUS, 2000) ocorre quando um projeto é executado aplicando diferentes processos: (i) ao mesmo tempo dentro do mesmo projeto (diversidade *latitudinal* - observada em projetos com múltiplas equipes trabalhando em paralelo); (ii) ao longo do tempo (diversidade *longitudinal* - adoção de processos diferentes ao longo do ciclo de vida de um projeto, observado, por exemplo, na transição do desenvolvimento para a fase de manutenção) (SIEBEL *et al.*, 2003).

Por fim, um processo não pode ser definido sem levar em consideração as pessoas que vão executá-lo (como os funcionários da organização) ou interagir com

---

[2] Um modelo de desenvolvimento de software define em um alto nível de abstração um conjunto de práticas recomendadas para o desenvolvimento de software (SANTOS, 2009, SOMMERVILLE, 2004).

[3] O modelo de desenvolvimento orientado ao planejamento (também conhecido na literatura como tradicional, disciplinado ou rigoroso) é tipicamente exemplificado pelos modelos de maturidade e normas de qualidade, tais como o CMMI-DEV (CHRISSIS *et al.*, 2006), a ISO/IEC 12207 (ISO/IEC, 1995) e o MR-MPS (SOFTEX, 2011).

ele (como os clientes e fornecedores). É preciso conectar o processo com as pessoas e a forma como elas trabalham e interagem (SWENSON *et al.*, 2011). Porém, a **diversidade de pessoas** envolvidas (com diferentes motivações, experiências, conhecimentos e etc.) e as interações sociais entre elas, acrescenta mais um desafio à tarefa de definição de processos de software.

Para lidar com toda esta diversidade, há uma crescente necessidade por parte da indústria de software pela rápida e efetiva definição de processos de software (ALEIXO *et al.*, 2010, HANSSON *et al.*, 2006, PATEL *et al.*, 2006). Isto envolve a adequação à realidade dos projetos e organizações e o reúso de experiências passadas na definição de processos de software com o objetivo de aumentar a produtividade durante a realização de tal atividade.

Apesar de ser uma das principais tarefas executadas pelo gerente do projeto, a definição do processo específico para o projeto não é simples (TERNITE, 2009). Esta tarefa envolve o conhecimento de muitos aspectos da Engenharia de Software e requer a harmonização de muitos fatores do contexto da equipe, do projeto ou da organização (BARRETO *et al.*, 2010). Compreender esses fatores e como eles poderão afetar o processo é um desafio para o gerente de projeto (XU e RAMESH, 2008), principalmente para aqueles com menos experiência.

Devido a esta complexidade, geralmente, o gerente de projeto não é capaz de avaliar todas as opções disponíveis e acaba fazendo esta definição de processos de forma *ad-hoc*, baseado na sua experiência e conhecimento. Este conhecimento, geralmente, é limitado a alguns modelos de desenvolvimento apenas, pois não é de se esperar que o gerente esteja familiarizado com cada um deles, dada a sua diversidade.

Além disso, as decisões de definição do processo para o projeto costumam ser tomadas pelo gerente no início do projeto e registradas no plano de projeto. No entanto, reconhece-se que é impossível prever neste momento um processo totalmente adequado, pois o contexto do projeto muda ao longo da sua execução e o processo precisa acompanhar essas mudanças de contexto.

Como resultado, o processo definido pode não ser a melhor alternativa para o projeto em questão. Desta forma, observa-se que a definição de processos de software por si só não é uma garantia de qualidade, já que processos podem ser definidos de forma equivocada (COSTA, 2010).

Como o orçamento do projeto, o prazo de desenvolvimento e a qualidade do produto dependem diretamente da qualidade do processo de software, a definição de processos precisa ser feita corretamente para evitar que as seguintes possíveis consequências negativas se materializem: um processo de software ruim pode envolver atividades desnecessárias que levam a perda de tempo e dinheiro; e a

omissão de atividades necessárias pode prejudicar a qualidade do produto (PEDREIRA *et al.*, 2007).

Com o propósito de facilitar a definição de processos, é possível seguir duas abordagens diferentes (CHRISSIS *et al.*, 2006): a adaptação de um processo específico para o projeto a partir do processo padrão da organização (PEDREIRA *et al.*, 2007); ou a **composição** de um processo para o projeto baseada em unidades menores e reutilizáveis.

A composição de processos foi a abordagem adotada neste trabalho, como forma de promover a reutilização do conhecimento relacionado a processos de software. Segundo Costa (2010), definir mecanismos para prover ou auxiliar a composição de processos pode reduzir esforços de modelagem, garantir a qualidade dos processos e impedir que erros já resolvidos em projetos passados tornem a trazer problemas em novos projetos.

Este trabalho adota uma estratégia de reutilização de processos. Esta estratégia é implementada através de uma abordagem de composição e otimização de processos de software (COMPOOTIM). A abordagem COMPOOTIM funciona com base nos dados oriundos de uma Linha de Processos Baseada em Contexto (LPBC) (NUNES *et al.*, 2010). Portanto, o objetivo deste trabalho é apresentar um exemplo de criação de uma LPBC que gera os dados necessários para alimentar a abordagem COMPOOTIM.

O restante deste documento está organizado da seguinte forma: a Seção 2 apresenta a abordagem COMPOOTIM. A Seção 3 resume a abordagem de Linha de Processos Baseada em Contexto. Na Seção 4, é descrita a abordagem ArchToDSSA, que, apesar de ter sido originalmente criada para a definição de arquiteturas de referência, foi adaptada neste trabalho para apoiar a criação da linha de processos. A Seção 5 detalha o exemplo de criação da LPBC e a Seção 6 ilustra um possível uso dessa LPBC. Por fim, a Seção 7 conclui este trabalho.

# 2. COMPOOTIM: Composição e Otimização de Processos de Software

A abordagem COMPOOTIM (Figura 1) tem como objetivo apoiar o gerente de projeto na composição do processo de software específico para um determinado projeto. A ideia é automatizar alguns passos, possivelmente diminuindo o esforço necessário para a execução desta atividade e melhorando a qualidade e a adequação do processo resultante.

A abordagem COMPOOTIM se propõe a otimizar a composição de processos aderentes ao contexto atual do projeto de desenvolvimento. Usando métodos de otimização, que têm como finalidade buscar soluções que atendam um conjunto de restrições impostas, para maximizar ou minimizar um determinado fator

(PAPADIMITRIOU e STEIGLITZ, 1998), é feita uma busca entre as diversas alternativas existentes no espaço de solução, selecionando uma solução boa e viável para compor um processo que satisfaça ao contexto do projeto.

A definição inicial e modelagem computacional deste problema de composição de processos de software já foram apresentadas em trabalhos anteriores (MAGDALENO *et al.*, 2010, MAGDALENO, 2010a) como um problema Engenharia de Software Baseada em Buscas (*Search Based Software Engineering – SBSE*) (CLARKE *et al.*, 2003, HARMAN e JONES, 2001).

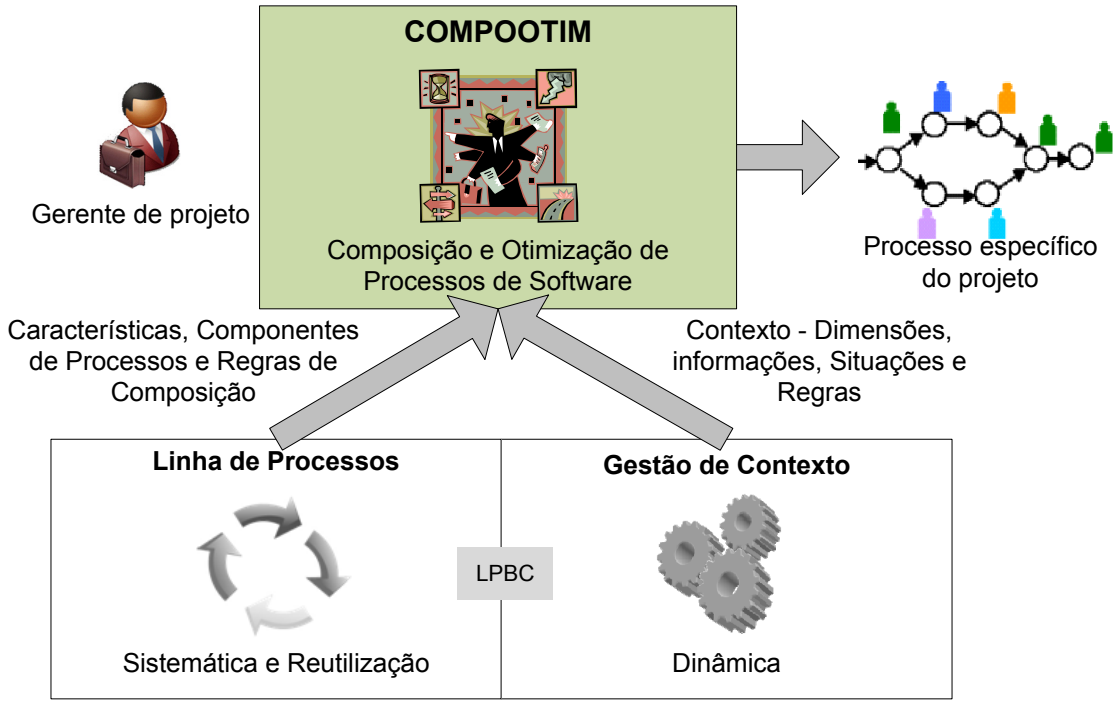


**Figura 1** – Abordagem COMPOOTIM de Composição e Otimização de Processos de Software

Para que a abordagem possa funcionar ela precisa receber alguns insumos. Estes insumos são conjuntos de dados sobre o processo e o contexto do projeto, da organização e da equipe. Estes dados são fornecidos pela abordagem de Linha de Processos Baseada em Contexto (LPBC) (NUNES *et al.*, 2010), descrita na próxima seção. Todos esses dados recebidos são combinados e analisados pela abordagem COMPOOTIM em busca de uma solução (Figura 1). A criação destes conjuntos de dados é o foco deste trabalho e será detalhada na Seção 5.

Desta forma, a abordagem COMPOOTIM oferece ao gerente de projeto um apoio para a tomada de decisão sobre a melhor forma de compor um processo específico para o projeto, uma vez que possibilita que seja considerado um universo maior de fatores e alternativas. Seguindo essa ideia, é possível propor ao gerente de projeto algumas opções de processos que satisfaçam as restrições e que otimize alguns dos fatores envolvidos no problema. Porém, a decisão final sobre o processo que será efetivamente adotado continua sendo dele.

Além disso, é possível reexecutar a otimização do processo a qualquer

momento e fazer mudanças durante a execução do processo, pois não basta que a composição de processos seja feita uma única vez no início do projeto, visto que requisitos e o ambiente do projeto mudam. Portanto, ao reexecutar a otimização do processo, é possível considerar o contexto atual de execução do processo e manter o alinhamento do processo com as necessidades atuais do projeto (BRINKKEMPER, 1996, FITZGERALD *et al.*, 2003, MAGDALENO, 2010b, XU e RAMESH, 2008).

# 3.Linha de Processos Baseada em Contexto

Os processos de software – apesar de também serem variáveis entre os projetos de software – ainda não são gerenciados da mesma forma sistemática que os produtos de software (ROMBACH, 2006). Esta sistemática permitiria que estes processos também fossem organizados de acordo com as suas similaridades e diferenças, facilitando e guiando a sua composição de acordo com as necessidades e objetivos específicos dos projetos.

Neste trabalho, esta sistemática é estabelecida valendo-se de uma estrutura de reutilização de processos baseada em linha de processos. Uma **linha de processos** corresponde à aplicação do conceito de linha de produtos[4] (NORTHROP, 2002) para o contexto de processos. Simplificadamente, uma linha de processos (ALEIXO *et al.*, 2010, BARRETO *et al.*, 2010, JAUFMAN e MUNCH, 2005, ROMBACH, 2006, TEIXEIRA, 2011, TERNITE, 2009, WASHIZAKI, 2006) é uma linha de produtos onde os produtos são processos.

Washizaki (2006) define uma linha de processos como "um conjunto de processos de um determinado domínio de problema ou com um determinado propósito, que tem características em comum e é construído baseado em ativos reutilizáveis de processos". Diferentemente da linha de produtos, a abordagem de linha de processos é recente e não está ainda totalmente consolidada na literatura. Como as definições propostas para linha de processos ainda não estão totalmente claras ou bem-definidas, neste trabalho adota-se, a seguinte definição para linha de processos (MAGDALENO, 2010c, NUNES *et al.*, 2010, TEIXEIRA, 2011):

> Um conjunto de elementos de processos que compartilham características comuns e variáveis dentro de um domínio específico e são desenvolvidas a partir de artefatos que podem ser reutilizados e combinados entre si, segundo regras de composição e recorte, para compor e adaptar processos.

Os objetivos de uso de linhas de processos são: aumentar a produtividade da atividade de composição de processos, diminuindo o esforço necessário para realizá-la; aumentar a qualidade e adequação dos processos gerados (reutilização

[4] Uma linha de produtos é um "conjunto de produtos de software que compartilham um conjunto de características comuns e controladas que satisfazem necessidades de um segmento de mercado em particular, e são desenvolvidos a partir de artefatos, de forma predefinida" (NORTHROP, 2002).

do conhecimento de especialistas e de dados sobre utilização); representar variabilidades e semelhanças entre processos para potencializar a reutilização; e diminuir os riscos de uma composição inadequada do processo (BARRETO *et al.*, 2010, JAUFMAN e MUNCH, 2005, PEDREIRA *et al.*, 2007, ROMBACH, 2006).

Em um trabalho anterior de nosso grupo de pesquisa (NUNES *et al.*, 2010), foi proposta a utilização de informações de **contexto** para apoiar a composição de processo, e, assim, melhorar a utilização e gestão da linha de processos. Contexto é "uma descrição complexa do conhecimento compartilhado sobre circunstâncias físicas, sociais, históricas e outras dentro das quais ações ou eventos ocorrem" (BREZILLON, 1999). A gestão de contexto permite identificar modificações em um processo em tempo de execução, a fim de adaptá-lo às novas situações. Portanto, a composição de processo deve ser continuamente realizada à medida que mais informações estão disponíveis durante a realização do projeto, ou seja, considerando o contexto de execução do processo.

Assim, foi criada a abordagem de **Linha de Processos Baseada em Contexto (LPBC)**. A LPBC oferece a habilidade de adaptar o comportamento do processo às mudanças no contexto. De forma flexível, ela permite ativar ou desativar alternativas de componentes de processos na linha dependendo do contexto. Isso ajuda a garantir que a instanciação do processo satisfaça às necessidades atuais.

A abordagem LPBC estabelece a sequência de passos para a criação da linha de processos e para a instanciação de processos a partir da linha. De forma análoga a engenharia de linha de produtos, a LPBC também compreende duas grandes fases (MONTERO *et al.*, 2007, ROMBACH, 2006, SCHNIEDERS e PUHLMANN, 2006): Engenharia de Domínio de Processos de Software (EDPS) e Engenharia de Aplicação de Processos de Software (EAPS) (Figura 2).

A EDPS é iniciada quando existe a necessidade de estabelecer uma linha de processos, ou quando os requisitos, necessidades e objetivos da organização mudam, promovendo a evolução da linha de processos. Em ambos os cenários, os modelos de processos da organização e/ou modelos de referência existentes, servem como entrada para esta fase.

A EDPS cria a linha de processos com um conjunto de processos que capturam as similaridades e variabilidades do domínio. A variabilidade pode ser entendida como "a habilidade que um artefato possui de ser alterado, customizado, ou configurado para um contexto em particular" (BOSCH, 2004). No contexto de processos de software, a variabilidade é importante para explicitar os pontos onde os processos se assemelham e, portanto, podem ser reutilizados, e os pontos onde eles diferem entre si, devendo receber tratamento específico. Desta forma, a

variabilidade oferece flexibilidade para lidar com a diversidade.

A fase da EDPS provê um conjunto de artefatos e infraestrutura para a derivação de múltiplos processos na próxima fase de EAPS. Os principais objetivos da EDPS são: (i) definir o escopo da linha de processos; (ii) definir as similaridades e variabilidades da linha de processos; (iii) definir e construir artefatos reutilizáveis que atendam a variabilidade desejada. Para alcançar estes objetivos, a EDPS é composta pelas atividades de análise, projeto e implementação do domínio do processo (Figura 2).

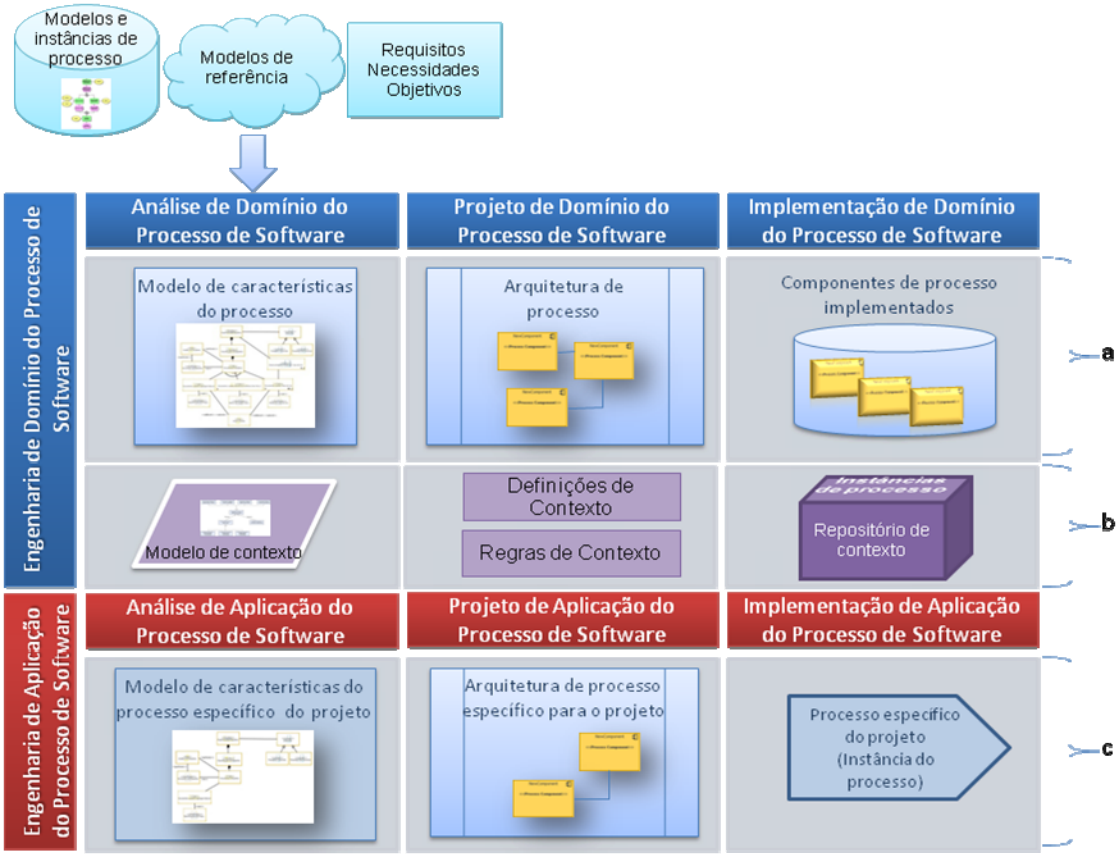


**Figura 2** – Linha de processos baseada em contexto

Na EDPS, a etapa de Análise tem como objetivo identificar o conhecimento de domínio através de um **modelo de características** (KANG *et al.*, 1990) (que representa as características de domínio, com as similaridades e variabilidades de um processo) e **regras de composição** (que definem as dependências ou relações de mútua exclusividade entre as características). Na etapa de Projeto, a **arquitetura de processos** é especificada para definir o "esqueleto" que um processo deve ter, estabelecendo os principais componentes e como eles se relacionam entre si. As características e os componentes do processo estão relacionados uma vez que cada característica tem um conjunto de componentes que a implementa. Por fim, a Implementação visa gerar **modelos executáveis** da arquitetura de processos (Figura 2a).

Com relação ao **contexto**, durante a Análise, também é possível identificar as informações contextuais consideradas relevantes para descrever o processo. O modelo de contexto fornece uma representação explícita de conhecimento contextual e é composto por **entidades/dimensões** de contexto e **informações** de contexto (FERNANDES *et al.*, 2008). Na fase de Projeto, este modelo de contexto é refinado através de **definições** de contexto (circunstâncias contextuais ou situações que podem acontecer com base na combinação de informações de contexto) e **regras** de contexto (definir uma sugestão sobre os recursos a serem selecionados com base em situações que acontecem no momento) (FERNANDES *et al.*, 2008). A Implementação de domínio envolve a criação do **repositório** de contexto (Figura 2b).

Após criada a linha de processos de software na fase de EDPS, em cada projeto, em vez de definir um processo a partir do zero, os gerentes podem fazer uso da infraestrutura da linha de processo baseada em contexto para compor um novo processo específico para o projeto. Isso significa que o gerente de projeto deverá tomar decisões associadas aos pontos de variação – pontos no processo onde é possível escolher quais variantes, entre as disponíveis, devem ser incluídas no processo para serem executadas. Entretanto, a experiência com linhas de produtos tem mostrado que essa não é uma tarefa trivial e que pode ser cara, custosa e sujeita a falhas (DEELSTRA *et al.*, 2005). O mesmo é esperado que aconteça no caso da composição de processos de software. A abordagem COMPOOTIM pode ajudar a reduzir a complexidade dessa configuração.

A fase de EAPS apresenta atividades semelhantes as da EDPS (Figura 2c). Em primeiro lugar, na Análise, o resultado é o **modelo de características recortado**, contendo apenas as características selecionadas para o novo processo a ser composto. Durante a etapa de Projeto, os **componentes do processo** são selecionados para serem usados dentro da arquitetura de processos específica do projeto. A partir do modelo de características gerado, das regras de composição, e do contexto existente, o novo **processo** é **composto**. Finalmente, a etapa de Implementação corresponde a uma análise de ferramentas de apoio para serem utilizadas para implementar o processo. Como resultado, o **processo** é efetivamente **implementado** e está pronto para ser executado, associado ao contexto de execução, e para ser adaptado em tempo de execução.

# 4.Abordagem ArchToDSSA

Em busca de uma abordagem para a criação da linha de processos baseada em contexto, identificou-se que a ArchToDSSA (KÜMMEL, 2007) tem um propósito similar, apesar de ter sido originalmente criada para a definição de arquiteturas de

referência. Assim, o propósito desta seção é explicar essa abordagem que será aplicada no contexto deste trabalho na Seção 5.

A ArchToDSSA visa à identificação de elementos arquiteturais mandatórios, opcionais e pontos de variação no domínio, procurando apoiar o engenheiro de domínio na tarefa de comparação de arquiteturas existentes no domínio, tais como as recuperadas a partir dos sistemas legados, e criação de uma arquitetura de referência, a partir da análise das informações disponíveis nas arquiteturas existentes.

Um apoio automatizado é oferecido através da ferramenta ArchToDSSATool, implementada no contexto do ambiente de desenvolvimento e reutilização Odyssey (ODYSSEY, 2011). Uma vez que a ArchToDSSA está inserida no contexto do ambiente Odyssey, é adotada a notação Odyssey-FEX (OLIVEIRA, 2006). A notação Odyssey-FEX contempla a representação de elementos mandatórios e opcionais, pontos de variação, variantes e invariantes em um domínio. Ao longo das fases da abordagem ArchToDSSA, faz-se necessário identificar tais elementos.

## 4.1.Requisitos para a abordagem

A abordagem foi definida a partir dos seguintes requisitos (KÜMMEL, 2007):

- A comparação entre elementos arquiteturais deve levar em consideração o fato de que elementos diferentes, em diferentes arquiteturas, podem ter sido criados com sintaxes diferentes e mesma semântica, isto é, dois ou mais elementos que tenham o mesmo significado podem ter sido criados com nomenclaturas diferentes. Portanto, é preciso ter mecanismos para auxiliar no processo de comparação;
- Devem ser permitidas ao engenheiro de domínio certas escolhas, tais como: o número mínimo de arquiteturas envolvidas para se considerar equivalência entre elementos, sua opcionalidade, pontos de variação e suas variantes;
- Pode ser definido se todos os elementos poderão ser comparados entre si ou se somente elementos do mesmo tipo serão comparados (ex. comparação apenas entre classes, ou entre classes e pacotes);
- A arquitetura de referência deve ser especificada de forma inteligível, em uma representação que seja familiar ao engenheiro de domínio, onde possam ser identificados com clareza tanto os elementos arquiteturais que compõem a arquitetura de referência, quanto seus atributos de opcionalidade e variabilidade;
- Deve ser previsto um apoio ferramental, no intuito de melhorar a produtividade do engenheiro de domínio e reduzir o esforço humano.

Após uma análise, concluí-se que todos estes requisitos também são desejáveis na criação da linha de processos de referência. Portanto, como a

abordagem ArchToDSSA foi criada para satisfazer estes requisitos, em linhas gerais a abordagem é adequada as necessidades deste trabalho.

## 4.2.Visão Geral da Abordagem

O processo sugerido (KÜMMEL, 2007) contempla várias tarefas divididas em três fases distintas. O resultado obtido em cada fase serve de entrada para a fase seguinte, como ilustra a Figura 3.

Na Fase 1 – Detecção de Equivalências e Opcionalidades, o objetivo é identificar as similaridades (equivalências) e diferenças entre os elementos das arquiteturas analisadas. Dessa forma, é possível identificar elementos mandatórios e/ou opcionais no domínio.

Na Fase 2 – Detecção das Variabilidades, as equivalências obtidas na Fase 1 são analisadas para que se possa definir as "Variabilidades" do domínio, isto é, pontos de variação e suas respectivas variantes.

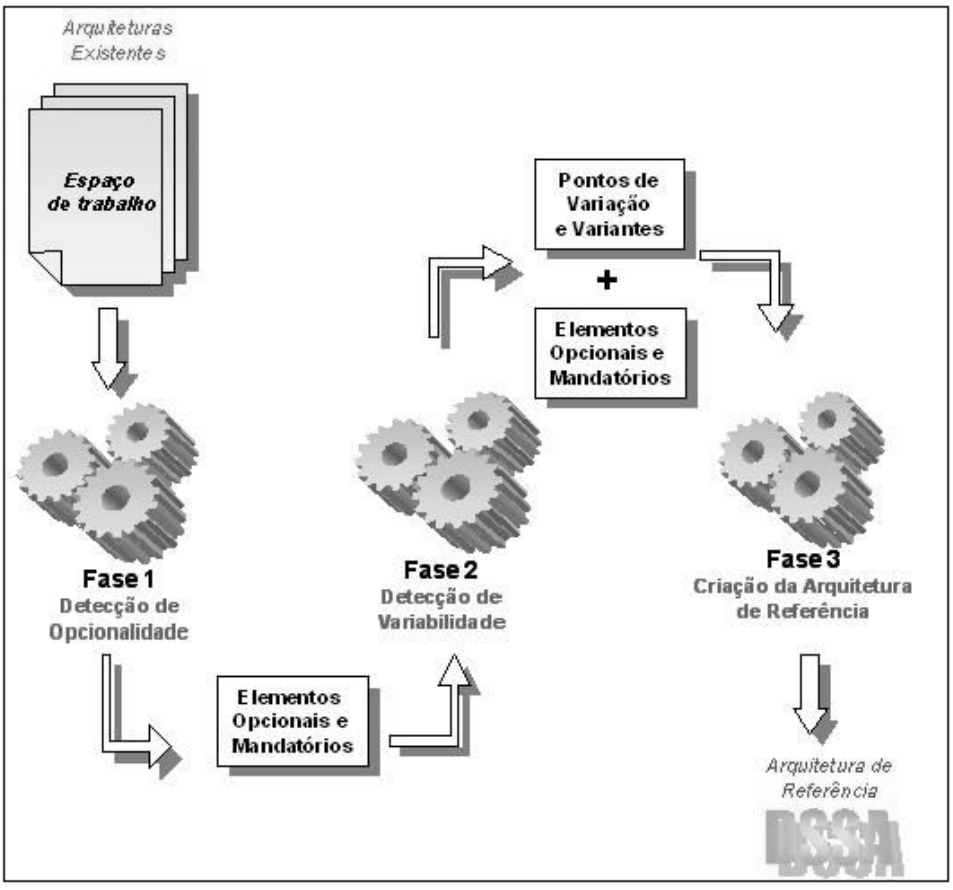


**Figura 3** – Abordagem ArchToDSSA **(KÜMMEL, 2007)**

Finalmente, na última fase (Fase 3 – Criação da Arquitetura de Referência), são definidos elementos arquiteturais, oriundos de arquiteturas disponíveis no conjunto de trabalho, que deverão compor uma possível arquitetura de referência para o domínio.

As atividades de cada uma destas fases são apresentadas nas próximas

seções.

## 4.3.Fase 1: Detecção de Equivalências e Opcionalidades

As atividades descritas na Fase 1 são retratadas na Figura 4. Nesta fase, arquiteturas existentes são comparadas, e, seguindo alguns critérios de comparação, as equivalências são identificadas. Em seguida, mediante parâmetros de opcionalidade aplicados sobre as equivalências identificadas, são determinados os elementos opcionais e mandatórios que poderão compor uma arquitetura de referência de domínio.

Uma equivalência é um elemento identificado unicamente, que representa um elo de ligação entre elementos arquiteturais equivalentes, pertencentes a uma mesma arquitetura ou a arquiteturas diferentes. Neste contexto, o termo "equivalente" é usado com a finalidade de identificar elementos arquiteturais distintos que possuam o mesmo significado. Em relação a uma equivalência, assume-se, ainda que (KÜMMEL, 2007):

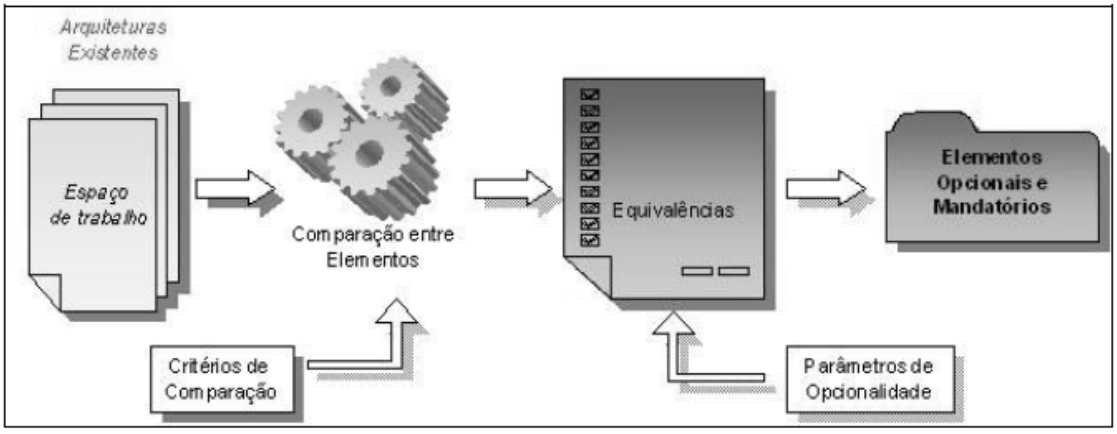


**Figura 4** – Abordagem ArchToDSSA – Fase 1 **(KÜMMEL, 2007)**

a) o número de elementos pertencentes a uma equivalência está compreendido no intervalo (2,...,n);

b) uma vez pertencente a uma equivalência, um elemento não pode pertencer a outra;

c) uma equivalência possui reciprocidade, isto é, se a1 é equivalente a b1, então b1 é equivalente a a1;

d) uma equivalência possui transitividade, isto é, se a1 é equivalente a b1, e b1 equivalente a c1, então a1 é equivalente a c1.

Na busca por elementos equivalentes, comparar aplicações distintas em um mesmo domínio pode vir a ser uma tarefa mais árdua do que comparar versões diferentes de uma mesma aplicação. Isso porque, embora um conjunto de trabalho contenha aplicações de um mesmo domínio, podem existir casos onde elementos arquiteturais equivalentes estejam descritos de forma diferente. Tal fato ocorre porque, apesar das aplicações analisadas fazerem parte de um mesmo domínio,

elas podem não ter sido necessariamente implementadas por uma mesma equipe, em um mesmo período, ou dentro de uma mesma empresa. Isso pode ocasionar uma grande variedade de elementos equivalentes, porém com nomes distintos, nas diferentes aplicações analisadas. Além disso, à medida que cresce o número de aplicações a serem comparadas e/ou o tamanho das arquiteturas extraídas, cresce também o trabalho envolvido na atividade de comparação dessas arquiteturas.

A identificação de equivalências deve ser realizada analisando-se elementos arquiteturais nas diferentes arquiteturas, e para tanto, assume-se a utilização da comparação dos nomes dos elementos arquiteturais. Dois ou mais elementos arquiteturais de mesmo nome seriam fortes candidatos a originar uma equivalência. No entanto, quando os nomes não são idênticos, determinar a equivalência de elementos arquiteturais passa a depender muito do conhecimento de um engenheiro de domínio. Sendo assim, assume-se na abordagem, além da equivalência por nomes idênticos, a utilização de alguns critérios de comparação.

**Critério 1 – Utilização do Dicionário de Sinônimos**

O dicionário de sinônimos consiste de uma lista ligando palavras com um mesmo significado, tal como em um dicionário comum. Essa lista pode ser criada e alimentada pelo engenheiro de domínio com base em seu conhecimento sobre o domínio. As palavras do dicionário podem não representar necessariamente o nome de elementos arquiteturais, mas assim como em qualquer dicionário, ligar cadeia de caracteres equivalentes.

**Critério 2 – Utilização de Lista de Palavras Ignoradas**

Assim como no dicionário de sinônimos, o conhecimento do engenheiro de domínio é usado para detectar a existência de palavras a serem ignoradas no contexto da comparação de nomes. Cria-se, portanto, uma lista de palavras a serem ignoradas.

**Critério 3 - Divisão de Nomes em Partes**

Quando dois nomes (nome1 e nome2) são comparados, o primeiro passo é decidir se cada nome será tratado como um todo, ou se será dividido em partes. Caso o nome seja tratado como um todo, cada nome será comparado integralmente. Seguindo esse critério, um nome é dividido em partes de tal forma que, quando uma letra maiúscula, hífen, ou underscore for encontrado no meio da palavra, uma nova parte é definida.

**Critério 4 - Comparação de Elementos do Mesmo Tipo**

Na comparação entre elementos arquiteturais, o engenheiro pode decidir se um elemento de uma arquitetura poderá ser comparado com qualquer elemento de outra arquitetura (ex: classes com pacotes), ou se deverão ser comparados somente elementos do mesmo tipo (ex: pacotes com pacotes, classes com classes).

**Critério 5 - Número Mínimo de Arquiteturas Equivalentes**

Neste critério, o engenheiro de domínio pode determinar qual o número mínimo de arquiteturas às quais elementos equivalentes devem pertencer para ser considerada realmente uma equivalência.

Levando em conta esses cinco critérios, um algoritmo é sugerido na abordagem ArchToDSSA (KÜMMEL, 2007) para a identificação de equivalências. As equivalências identificadas nesta atividade serão usadas na fase de detecção de Variabilidades, portanto, quanto mais preciso for o resultado desta etapa, melhor será o resultado na fase subsequente.

Em seguida, é previsto, nesta fase, que um elemento seja classificado como um mandatório ou opcional. Para tanto, o critério Número mínimo de arquiteturas equivalentes volta a ser considerado, indicando que um elemento é mandatório no domínio, quando ele estiver presente em um certo número de arquiteturas distintas, igual ou maior, ao critério estabelecido pelo engenheiro. Caso contrário, o elemento será considerado opcional.

## 4.4.Fase 2: Detecção das Variabilidades

De posse das equivalências identificadas na Fase 1 da abordagem, dá-se início à segunda fase, de Detecção das Variabilidades, como exemplificado na Figura 5. O objetivo desta fase é identificar pontos de variação e suas respectivas variantes.

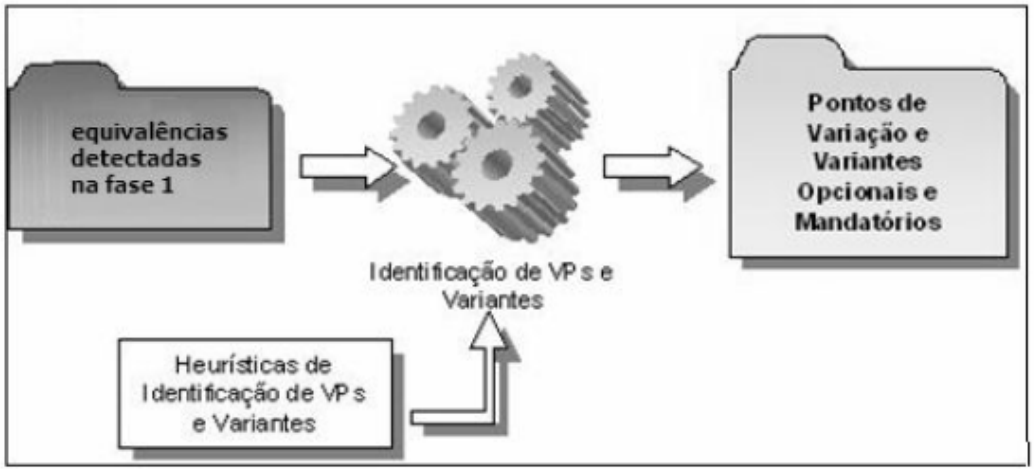


**Figura 5** – Abordagem ArchToDSSA – Fase 2 **(KÜMMEL, 2007)**

Assim como na identificação de equivalências, para nortear o processo de identificação de pontos de variação, existe um critério que define o número mínimo de arquiteturas equivalentes para se considerar um elemento como um ponto de variação. Este critério indica em quantas arquiteturas um dado elemento deve possuir equivalência para que seja considerado como um ponto de variação.

Para a identificação de pontos de variação e variantes, é definido um algoritmo que utiliza heurísticas para o mapeamento de pontos de variação e suas variantes com interfaces e classes que participam de um relacionamento de

herança (KÜMMEL, 2007). Além disso, também é permitido que o engenheiro de domínio defina os pontos de variação como melhor lhe convier, não fazendo restrições quanto às suas escolhas. Na prática, isto significa que o engenheiro pode determinar os pontos de variação com base unicamente no seu conhecimento do domínio, sem seguir qualquer critério sugerido pela abordagem proposta. Isto se deve ao fato de que o principal objetivo da abordagem é auxiliar e em alguns casos sugerir ao engenheiro opções para a criação de uma arquitetura de referência, e não impor restrições ao seu trabalho.

Vale ressaltar que devido a sua especificidade este algoritmo de identificação de variabilidades não foi utilizado no presente trabalho, onde optou-se por deixar o engenheiro de domínio definir livremente os pontos de variação e suas variantes.

## 4.5.Fase 3: Criação de uma Arquitetura de Referência

A terceira e última fase da abordagem (Figura 6) tem como objetivo selecionar elementos para a criação de uma arquitetura de referência de domínio. Os elementos selecionados nas fases anteriores irão compor a arquitetura de referência sugerida. Porém, não só os elementos são selecionados, mas também os relacionamentos entre eles.

Nesta fase é proposta a definição de uma arquitetura-base, ou seja, a escolha de uma das arquiteturas presentes no conjunto de trabalho como arquitetura-base. Uma vez escolhida pelo engenheiro, a arquitetura-base terá todos os seus elementos e relacionamentos incondicionalmente presentes na arquitetura de referência.

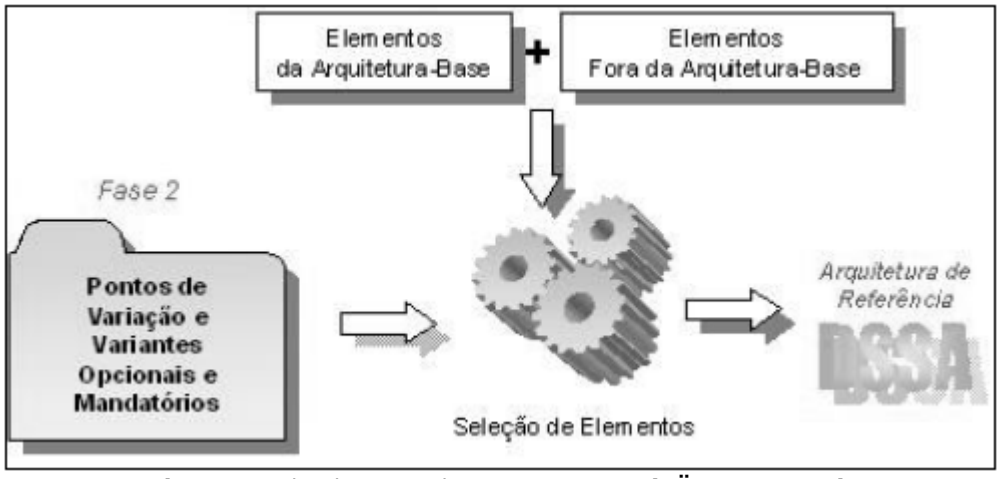


**Figura 6** – Abordagem ArchToDSSA – Fase 3 **(KÜMMEL, 2007)**

Além da arquitetura-base, o engenheiro pode selecionar elementos das outras arquiteturas para a criação da arquitetura de referência. Para auxiliar o engenheiro na atividade de seleção de elementos para a arquitetura de referência, são previstos alguns critérios:

- Caso o elemento selecionado já possua um elemento equivalente na arquitetura-base, o engenheiro não deve incluí-lo na arquitetura de referência;
- As variantes das arquiteturas não pertencentes à arquitetura-base, identificadas na Fase 2, devem ser selecionadas. Estas variantes deverão ser inseridas no ponto de variação equivalente ao seu ponto de variação na arquitetura-base e deverão manter o mesmo tipo de relacionamento existente com o seu ponto de variação.

# 5.Criação da linha de processos baseada em contexto

Para a criação da linha de processos baseada em contexto, o primeiro passo é definir a estratégia de captura de conhecimento que será adotada, conforme apresentado na próxima seção. Em seguida, é preciso popular todos os conjuntos de dados previstos na criação da linha. A construção de cada um destes conjuntos de dados será apresentada nas seções subsequentes.

## 5.1.Estratégia de Captura de Conhecimento para Linha de Processos

Existem duas estratégias de captura de conhecimento para a criação de uma linha de processos: *bottom-up* e *top-down* (ROMBACH, 2006, WASHIZAKI, 2006) que se diferenciam basicamente pelos insumos utilizados. Na estratégia *bottom-up,* também adotada nas propostas de Washizaki (2006) e Barreto *et al*. (2010), o conhecimento existente nos modelos de processos da organização ou nas instâncias executadas dos projetos de desenvolvimento de software é recuperado, através de análise ou mineração dos processos (*process mining*) (AALST e GIINTHER, 2007), e utilizado para criar a linha de processos. Porém, nesta estratégia enfrenta-se o risco da memória organizacional sobre o processo não estar totalmente estruturada ou acessível.

Por sua vez, a estratégia *top-down* inicia-se a partir de modelos de referência, onde processos ou partes deles são adquiridos de forma a serem combinados para formar novos processos. Esta estratégia é mais adequada quando não existe um conjunto de processo previamente definidos na organização e os requisitos/objetivos para a criação da linha de processos estão claros (BARRETO *et al.*, 2010). Entretanto, Washizaki (2006) alerta para o risco de se realizar a análise do domínio, a partir destes modelos de referência, e não conseguir garantir a sua completude devido a sua extensão. Segundo Jaufman e Munch (2005), nesta estratégia deve-se manter um alto nível de adaptabilidade do processo, uma vez que inicialmente a linha de processo criada é menos madura.

Neste trabalho, foi adotada inicialmente a estratégia *top-down*, utilizando

como base modelos de referência publicados livremente, tais como o Scrum[5] (SCHWABER, 2004), XP (*Extreme Programming*)[6] (BECK, 2004), OpenUP[7] e RUP (*Rational Unified Process*) (IBM, 2009). Os modelos de referência agregam solução aos problemas do ponto de vista do domínio e carregam um conjunto de conhecimento bastante amadurecido.

## 5.2.Modelagem de Características de Processo

O primeiro conjunto de dados que precisa ser criado é o de características do processo. Inicialmente, os modelos do Scrum, XP, OpenUP e RUP foram analisados individualmente. Cada um destes modelos foi representado através de um diagrama de características próprio. Estes diagramas foram construídos utilizando a notação *OdysseyProcess-FEX* (TEIXEIRA, 2011), mas foram enfatizados os conceitos necessários para a posterior composição do processo: disciplinas, atividades, tarefas e produtos de trabalho. Estes diagramas de características foram criados na ferramenta Odyssey (ODYSSEY, 2011).

O modelo do Scrum (Figura 7) é o menor de todos, com apenas 16 características sendo 1 disciplina, 2 atividades, 7 tarefas e 6 produtos. A existência de uma única disciplina é explicada pelo fato deste modelo de referência ser exclusivamente voltado para a gerência de projetos de software e não incluir questões técnicas de desenvolvimento. Uma atividade "representa o agrupamento de unidades de trabalhos menores, representadas por tarefas" (TEIXEIRA, 2011). Então, para organizar e agrupar as 7 tarefas existentes no Scrum, foram criadas 2 atividades (Planejamento de Projeto e Acompanhamento de Projeto) que não existiam no modelo original do Scrum.

O modelo do XP (Figura 8) é composto por 40 características sendo 4 disciplinas, 4 atividades, 23 tarefas e 9 produtos. A disciplina de Gerência de Projeto também está presente no XP, mas desta vez é acompanhada pelas disciplinas de Requisitos, Implementação e Teste. Neste caso, as atividades também foram criadas para organizar o conjunto de tarefas distribuídas entre as disciplinas.

No modelo do OpenUP (Figura 9) existe um total de 42 características, sendo 5 disciplinas, 3 atividades, 18 tarefas e 16 produtos. Este modelo tem as mesmas disciplinas do XP, somadas a disciplina de Arquitetura.

[5] Site: http://epf.eclipse.org/wikis/scrumpt
[6] Site: http://epf.eclipse.org/wikis/xp
[7] Site: http://epf.eclipse.org/wikis/openup

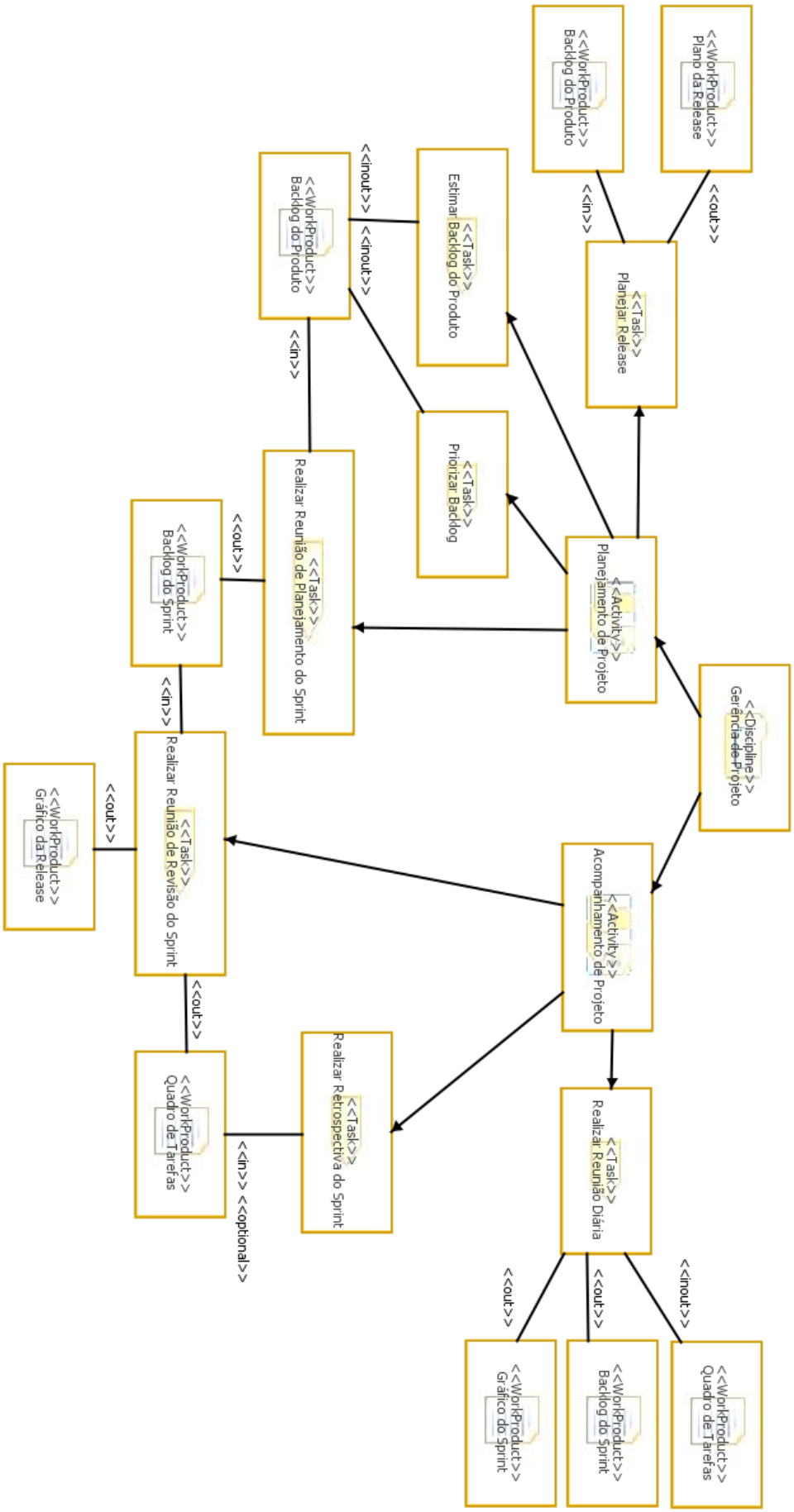


**Figura 7** – Modelo de Características Scrum

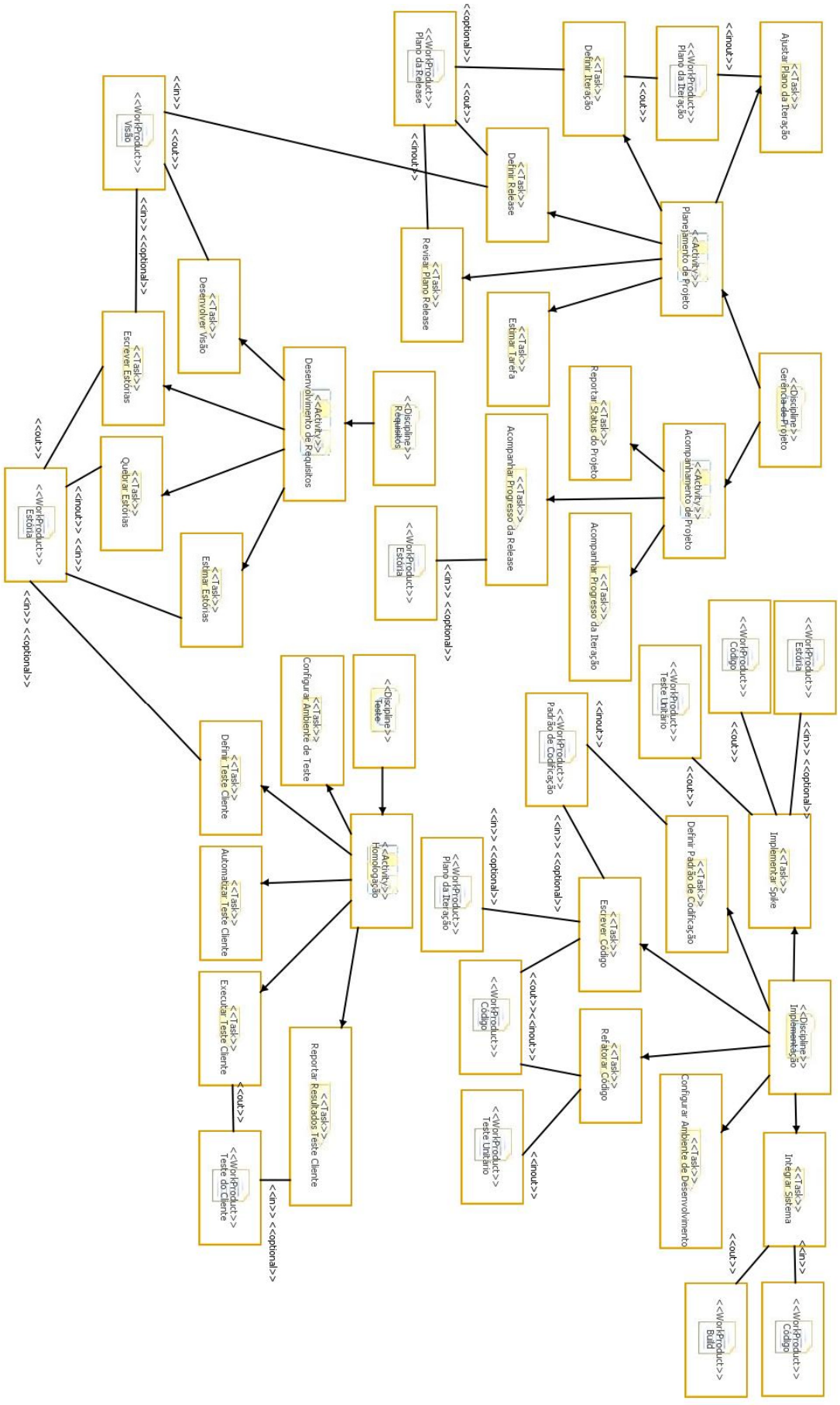


**Figura 8** – Modelo de Características XP

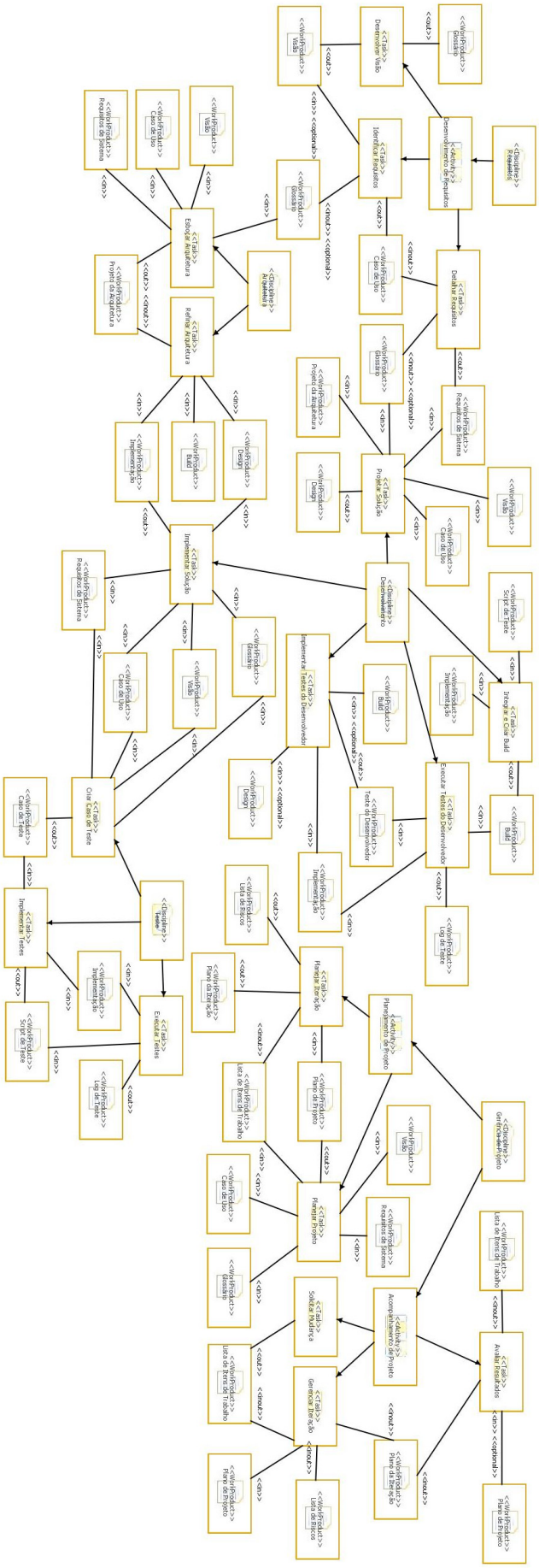

**Figura 9** – Modelo de Características OpenUP

Por fim, o modelo do RUP (Figura 10) é o mais completo. Originalmente, este modelo possui 9 disciplinas. Porém, foram modeladas e detalhadas apenas as 4 disciplinas também existentes nos outros modelos. Nestas 4 disciplinas existem: 25 atividades, 73 tarefas e 41 produtos, totalizando 143 características. Neste caso, as atividades modeladas já estavam presentes no modelo original.

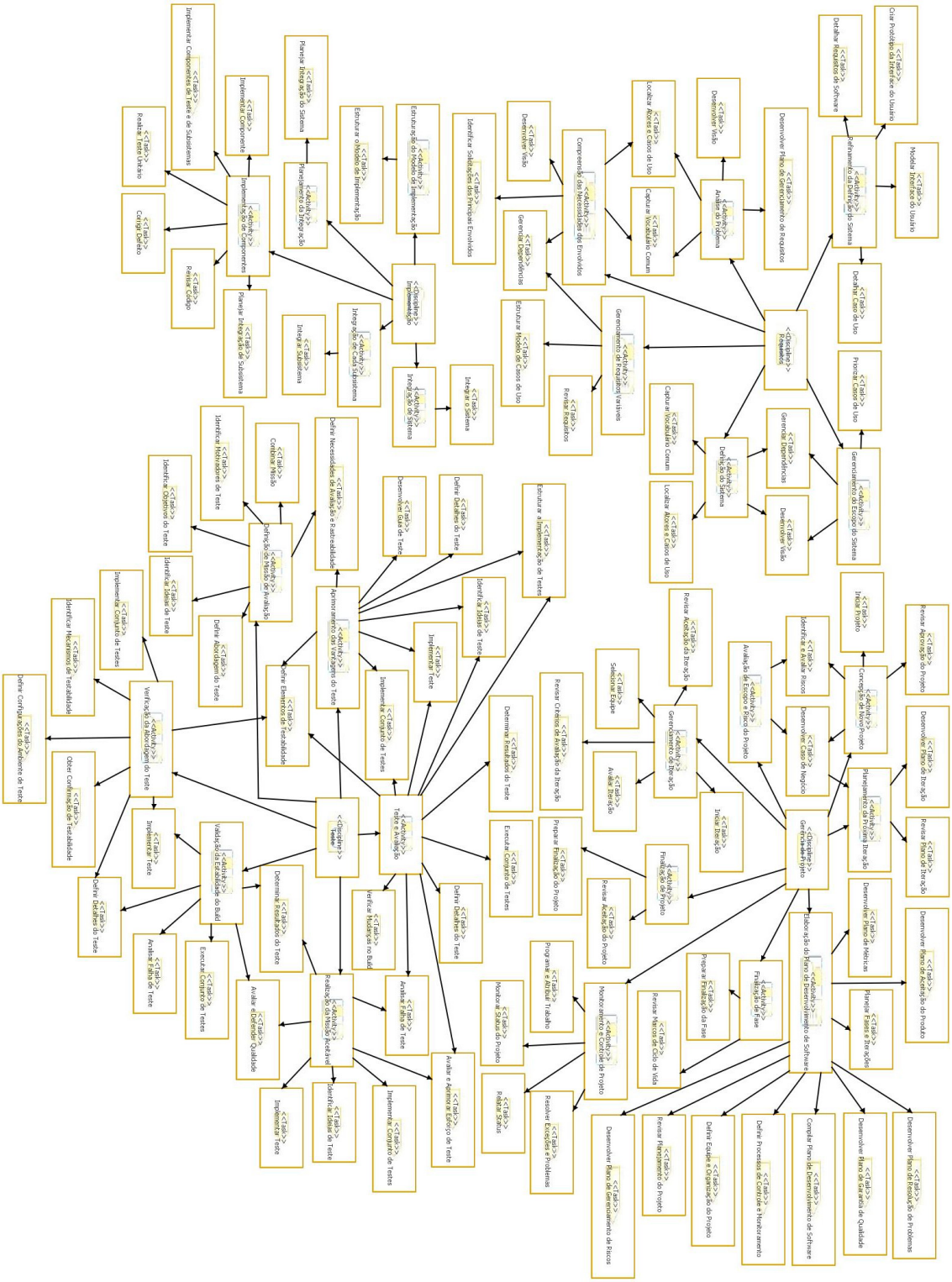

**Figura 10** – Modelo de Características RUP

A Tabela 1 resume o total de 241 características modeladas nestes diagramas de características individuais. Para a criação de um exemplo, foi escolhida a disciplina Gerência de Projeto, presente em todos os 4 modelos. A gerência de projeto tem por objetivo criar planos de projeto instanciá-los, monitorá-los e controlar sua execução. É geralmente composta por duas fases principais planejamento e controle do projeto (FEILER e HUMPHREY, 1992). Apesar da Gerência de Projeto ser uma disciplina importante para garantir o sucesso de um projeto de software, ela pode ter uma grande variabilidade entre as diferentes organizações, modelos e projetos.

**Tabela 1** – Total de Características

| Modelo | Característica | Quantidade | Total |
|---|---|---|---|
| Scrum | Disciplina | 1 | 16 |
| | Atividade | 2 | |
| | Tarefa | 7 | |
| | Produto | 6 | |
| XP | Disciplina | 4 | 40 |
| | Atividade | 4 | |
| | Tarefa | 23 | |
| | Produto | 9 | |
| OpenUP | Disciplina | 5 | 42 |
| | Atividade | 3 | |
| | Tarefa | 18 | |
| | Produto | 16 | |
| RUP | Disciplina | 4 | 143 |
| | Atividade | 25 | |
| | Tarefa | 73 | |
| | Produto | 41 | |

A partir dos modelos isolados, o próximo passo é a criação de uma linha de processos de referência, que integre todos esses diferentes modelos. Esta criação da linha de referência requer um grande esforço de generalização e de identificação de opcionalidades e variabilidades. A abordagem usada para a criação desta linha de processos de referência é apresentada na próxima seção.

## 5.3.Aplicação da abordagem ArchToDSSA para processos

A análise e comparação dos modelos de características isolados ajudam na tarefa de identificação dos principais elementos do domínio. Mesmo sabendo que os

modelos de um mesmo domínio apresentam elementos comuns, a identificação destas similaridades e diferenças não é trivial. Em primeiro lugar, existe um problema de uso de diferentes nomenclaturas para o mesmo elemento semântico. Além do problema de nomenclatura, existe ainda o problema de organização e composição dos elementos nos diferentes modelos. O mesmo elemento pode estar em diferentes locais nos modelos e apresentar diferentes conexões. Na medida em que a complexidade e o número de modelos comparados crescem, maiores são as dificuldades para se obter estas informações de forma precisa.

Conforme mencionado anteriormente, os requisitos nos quais a abordagem ArchToDSSA se baseou também são pertinentes para o caso de linha de processos. Assim, devido a similaridade com a tarefa em questão, a abordagem de Kümmel (2007) foi adaptada para apoiar a criação da linha de processos de referência que será usada como exemplo neste trabalho. Portanto, as mesmas fases da abordagem de Kümmel (2007) foram realizadas (Figura 11), conforme apresentado nas próximas seções.

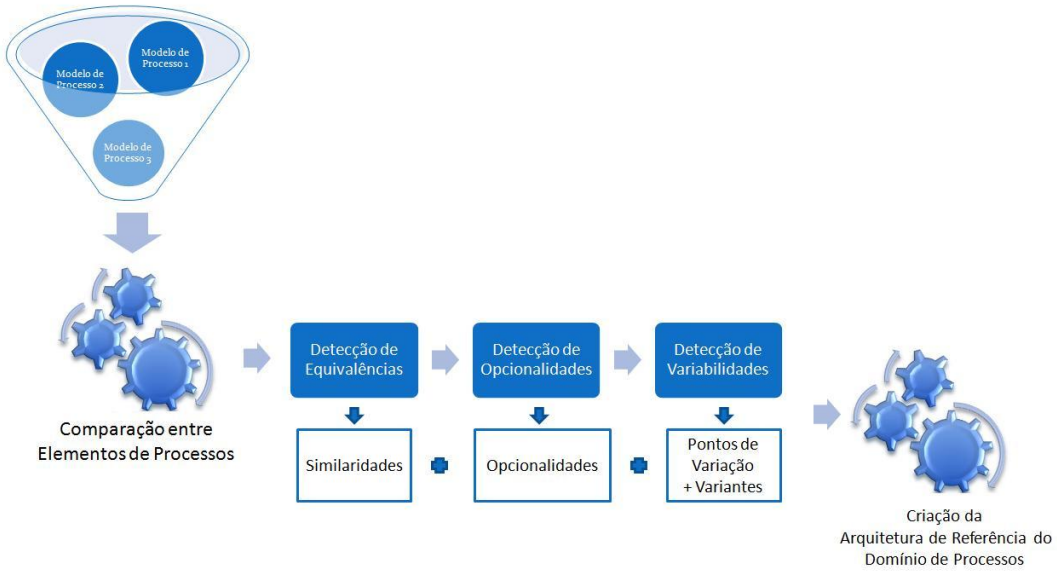


**Figura 11** – Abordagem para análise de similaridades e variabilidades no domínio de processos de software (TEIXEIRA, 2011)

### *5.3.1. Detecção de Equivalências e Opcionalidades*

A identificação de equivalências foi realizada analisando-se os elementos dos modelos de características individuais do Scrum, XP, OpenUP e RUP. O primeiro passo foi a comparação dos nomes dos elementos, visto que dois ou mais elementos com o mesmo nome seriam fortes candidatos a originar uma equivalência. As equivalências que foram identificadas neste primeiro momento são apresentadas na Tabela 2.

**Tabela 2** – Equivalências por nomes idênticos

| Id | Tipo | Nome | Modelos |
|---|---|---|---|
| e1 | Disciplina | Gerência de Projeto | Scrum, XP, OpenUP e RUP |
| e2 | Disciplina | Requisitos | XP, OpenUP e RUP |
| e3 | Disciplina | Teste | XP, OpenUP e RUP |
| e4 | Atividade | Planejamento de Projeto | Scrum, XP e OpenUP |
| e5 | Atividade | Acompanhamento de Projeto | Scrum, XP e OpenUP |
| e6 | Produto | Plano da Release | Scrum e XP |
| e7 | Produto | Plano da Iteração | XP, OpenUP e RUP |
| e8 | Produto | Visão | OpenUP e RUP |

Como a disciplina de Gerência de Projeto era a única presente em todos os modelos (Tabela 2), optou-se por focar este exemplo nesta disciplina. Portanto, as demais equivalências bem como o restante do exemplo daqui em diante, se concentram apenas nesta disciplina.

Quando os nomes não são idênticos, assume-se na abordagem a utilização dos critérios de comparação. A utilização dos critérios definidos por (KÜMMEL, 2007) para a criação da linha de processos de referência é explicada abaixo.

**Critério 1 – Utilização do Dicionário de Sinônimos**

O dicionário de sinônimos consiste de uma lista ligando palavras com um mesmo significado, tal como em um dicionário comum. O seguinte dicionário de sinônimos foi adotado para a criação da linha de processos de referência para gerência de projetos:

- Planejar = Definir
- Retrospectiva = Avaliação
- Realizar avaliação = Avaliar
- Próxima Iteração = Iteração
- Iteração = Sprint
- Acompanhar progresso = Gerenciar
- Gerenciar = Gerenciamento
- Relatar = Reportar
- Planejar = Planejamento = Desenvolver plano
- Plano de desenvolvimento de software = Plano de projeto
- Ajustar = Revisar
- Revisão de iteração = Avaliação de iteração
- Avaliar resultados = Avaliar iteração

- Backlog = Itens de trabalho

**Critério 2 – Utilização de Lista de Palavras Ignoradas**

Este critério define uma lista de palavras a serem ignoradas no contexto da comparação de nomes. Neste trabalho, não foram identificadas palavras a serem ignoradas.

**Critério 3 - Divisão de Nomes em Partes**

Quando dois nomes (nome1 e nome2) são comparados, o primeiro passo é decidir se cada nome será tratado como um todo, ou se será dividido em partes. Caso o nome seja tratado como um todo, cada nome será comparado integralmente. Neste trabalho, como a nomenclatura dos objetos não envolvia hífen ou underscore, não foi necessária a divisão dos nomes em partes.

**Critério 4 - Comparação de Elementos do Mesmo Tipo**

Na comparação entre elementos arquiteturais, o engenheiro pode decidir se um elemento de uma arquitetura poderá ser comparado com qualquer elemento de outra arquitetura ou se deverão ser comparados somente elementos do mesmo tipo. Para a criação da linha de processos de referência, optou-se pela comparação apenas de elementos do mesmo tipo para Disciplinas e Produtos. Porém, no caso de Atividades e Tarefas foi permitida a comparação de elementos de tipos diferentes, pois alguns modelos colocam esses elementos com granularidades diferentes.

**Critério 5 - Número Mínimo de Modelos Equivalentes**

Neste critério, o engenheiro de domínio pode determinar qual o número mínimo de arquiteturas às quais elementos equivalentes devem pertencer para ser considerada realmente uma equivalência. Neste exemplo, como a quantidade de modelos disponíveis é reduzida, foi estabelecido o número mínimo de 2 modelos para ser considerada uma equivalência.

Ao término desta fase, após a aplicação dos critérios 1, 4 e 5 acima descritos, a seguinte lista de equivalências por sinônimos foi obtida (Tabela 3):

**Tabela 3** – Equivalências por sinônimos

| Id | Tipo | Nome | Sinônimos | Modelos |
|---|---|---|---|---|
| e9 | Tarefa | Planejar Release | • Definir Release | Scrum, XP e OpenUP |
| e10 | Tarefa | Avaliar iteração | • Realizar Retrospectiva do Sprint<br>• Realizar Reunião de Revisão do Sprint | Scrum, OpenUP e RUP |
| e11 | Tarefa | Desenvolver Plano da Iteração | • Definir Iteração<br>• Realizar Reunião de Planejamento do Sprint<br>• Planejar Iteração | Scrum, XP, OpenUP e RUP |

| Id | Tipo | Nome | Sinônimos | Modelos |
|---|---|---|---|---|
| e12 | Atividade /Tarefa | Gerenciar Iteração / Acompanhar Progresso da Iteração | • Realizar Reunião Diária<br>• Gerenciar Iteração | Scrum, OpenUP e RUP |
| e13 | Tarefa | Revisar Plano da Iteração | • Ajustar Plano da Iteração | XP e RUP |
| e14 | Tarefa | Reportar Status do Projeto | • Relatar Status | XP e RUP |
| e15 | Tarefa | Desenvolver Plano de Projeto | • Planejar projeto<br>• Desenvolver Plano de Desenvolvimento de Software | OpenUP e RUP |
| e16 | Produto | Plano de Projeto | • Plano de Desenvolvimento de Software | OpenUP e RUP |
| e17 | Produto | Lista de Itens de Trabalho | • Backlog do Produto | Scrum e OpenUP |

Somando-se a lista de 8 equivalências por nomes idênticos com a lista de 9 equivalências por sinônimos, ao final desta fase foi obtido um total de 17 equivalências. Essa lista de equivalências foi usada para determinar os elementos mandatórios na linha de processos de referência para gerência de projeto. Todos os demais elementos serão opcionais (Figura 12).

### *5.3.2. Detecção das Variabilidades*

Para a identificação de pontos de variação e variantes, é definido um algoritmo que utiliza heurísticas para o mapeamento de pontos de variação e suas variantes com interfaces e classes que participam de um relacionamento de herança. Além disso, também é permitido que o engenheiro de domínio defina os pontos de variação com base unicamente no seu conhecimento do domínio (KÜMMEL, 2007).

Devido a sua especificidade, este algoritmo de identificação de variabilidades não foi utilizado no presente trabalho, onde optou-se por deixar o engenheiro de domínio definir livremente os pontos de variação e suas variantes. Assim, os pontos de variação e variantes propostos são apresentados na Figura 12.

### *5.3.3. Criação da linha de processos de referência*

O RUP foi escolhido como a arquitetura-base para a criação da linha de processos de referência por oferecer uma quantidade maior de atividades, o que corresponde também a um conjunto maior de opções para o Gerente de Projeto. Desta forma, as atividades, tarefas e produtos do RUP que não estavam presentes nos outros modelos foram colocados como opcionais na linha de processos de referência, com algumas exceções:

- A atividade Finalizar fase com as respectivas tarefas Preparar finalização da fase e Revisar Marcos de Ciclo de Vida não foram incluídas, pois os demais modelos

não têm este conceito de fase e isto poderia gerar confusão;

- Pelo mesmo motivo, também não foram incluídas as seguintes tarefas: Desenvolver Caso de negócio; Planejar fases e iterações;
- Para fins de simplificação, atividades com muitas aprovações e revisões também foram retiradas: Revisar aprovação do projeto; Revisar critérios de Aceitação da Iteração; Revisar aceitação do Projeto; e Revisar Aceitação da Iteração.

O diagrama de características criado com a representação da linha de processos de referência para Gerência de Projeto é apresentado na Figura 12. Ao final, foi obtido o seguinte conjunto de características (Tabela 4):

**Tabela 4** – Características da Gerência de Projeto

| Característica | Quantidade | Total |
|---|---|---|
| Disciplina | 1 | 57 |
| Atividade | 7 | |
| Tarefa | 35 | |
| Produto | 14 | |

Este modelo de características tem duas atividades principais: Planejamento e Monitoramento do Projeto, ambas obrigatórias. A atividade Planejamento de Projeto é composta por Concepção do Projeto, Desenvolvimento de Plano de Desenvolvimento de Software e Planejamento de Iteração. A atividade de Concepção do Projeto é opcional, o que significa que nem todo projeto precisa de um início formal. Sobre as atividades Desenvolvimento de Plano de Desenvolvimento de Software e Planejamento da Iteração, vale ressaltar que geralmente os modelos analisados trabalham com dois níveis de granularidade para o planejamento do projeto: projeto (considera o projeto como um todo) e iteração (foco no trabalho que está sendo feito no momento). Além disso, a atividade Desenvolvimento de Plano de Desenvolvimento de Software tem um alto nível de variabilidade - a tarefa Estimativa do Projeto pode trabalhar com estimativa de esforço, tamanho ou custos. A tarefa Desenvolver Plano de Projeto pode incluir a criação de diferentes planos, tais como Resolução de Problemas, Controle de Qualidade , Mátricas, Gestão de Riscos e Aceitação do Produto. A mesma ideia de diferentes tipos de estimativas também está presente na tarefa Estimativa de Trabalho. Nestes casos, um gerente de projeto pode escolher as melhores opções, considerando as necessidades do projeto, ao compor o processo. Finalmente, em Gestão de Iteração, a monitoração e avaliação da iteração recebem grande atenção. Esse monitoramento pode ser feito em uma base diária ou mensal e pode ser mais (como um relatório) ou menos (como uma reunião de standup) formal.

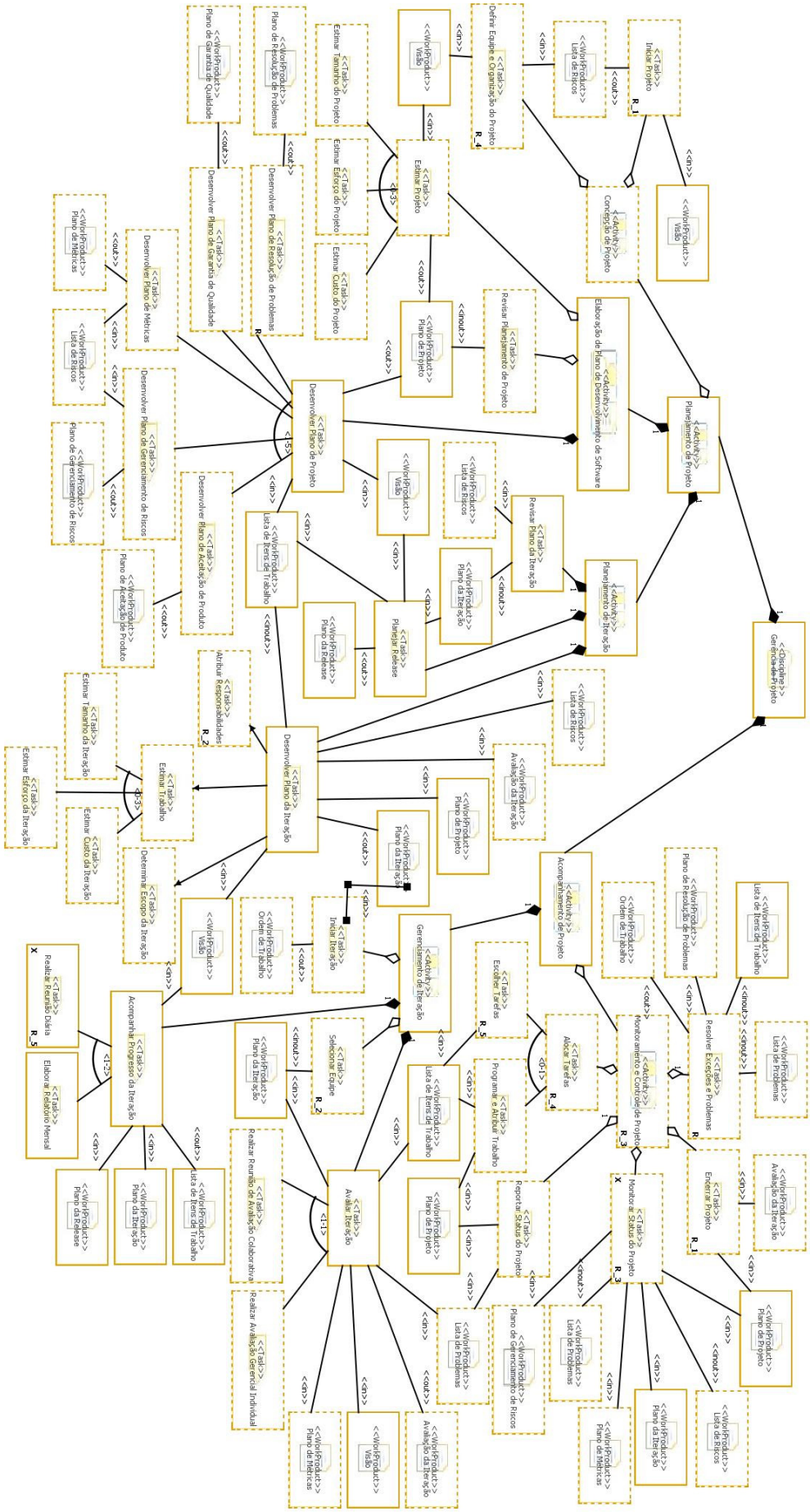

**Figura 12** – Exemplo – Modelo de Características de Gerência de Projeto

Estas características foram cadastradas na ferramenta COMPOOTIM (Figura 13) para compor o primeiro conjunto de dados necessário.

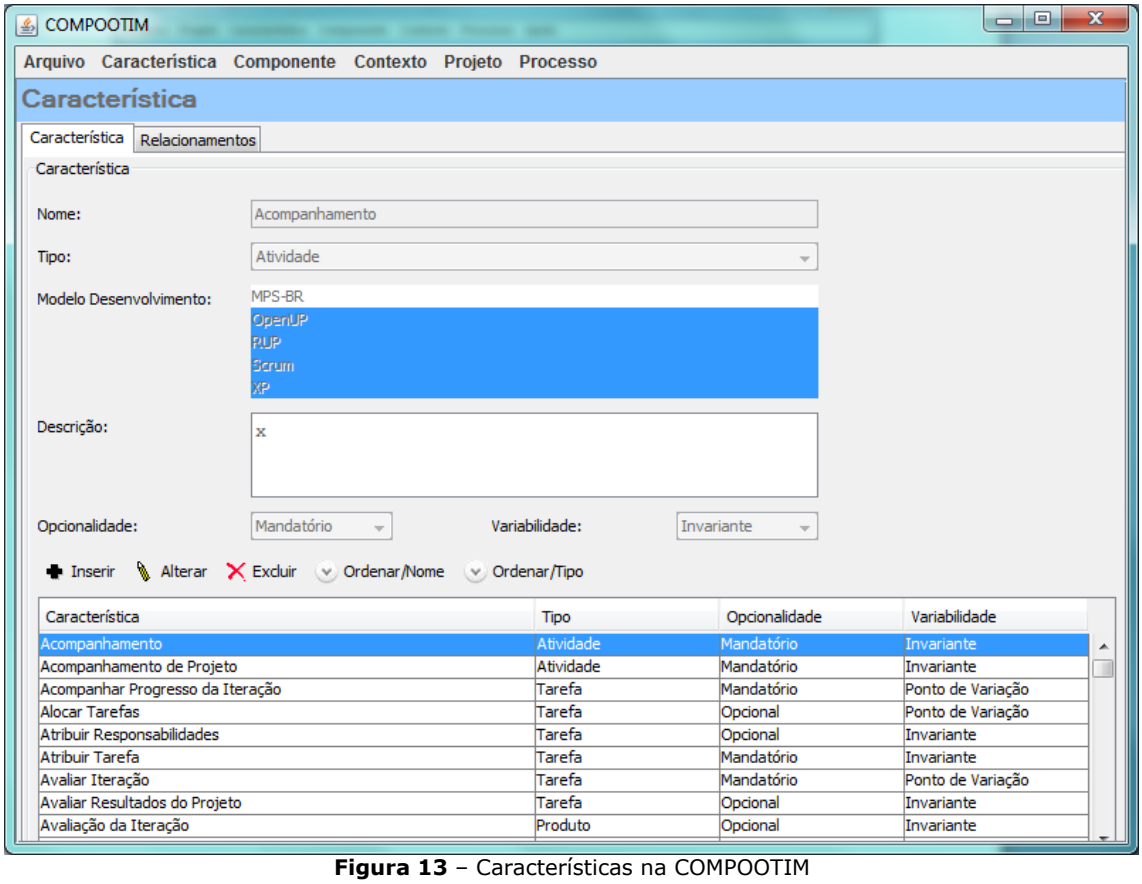


**Figura 13** – Características na COMPOOTIM

## 5.4.Regras de Composição de Características

A especificação de restrições entre as características, que inclui os relacionamentos de dependência e mútua exclusividade, é uma forma de indicar a necessidade ou incompatibilidade da seleção conjunta de características (OLIVEIRA, 2006). Kang *et al*. (1990) propõem a representação de tais conceitos por meio do uso de regras de composição:

- **Inclusivas**: definem relações de dependência entre duas ou mais características, indicando que elas devem ser selecionadas em conjunto.
- **Exclusivas**: definem relações de mútua exclusividade entre duas ou mais características que não devem ser escolhidas em conjunto.

Na notação Odyssey-FEX (OLIVEIRA, 2006), as regras de composição são expressas pela seguinte estrutura: antecedente + palavra-chave + consequente. A palavra-chave representa o tipo de regra: "requer" (*requires*), referente às regras inclusivas; e "exclui" (*excludes*), referente às regras exclusivas. Antecedente e consequente são expressões (literais ou *booleanas*) e representam uma característica ou combinação de características do domínio.

A partir do modelo de características da linha de processos de referência para gerência de projeto, foram criadas as regras de composição de características

apresentadas na Tabela 5. Estas regras já foram modeladas no Odyssey (Figura 14) e cadastradas na COMPOOTIM, conforme Figura 15.

As regras inclusivas indicam que quando a característica indicada no antecedente é selecionada, a característica do consequente também deve ser selecionada. Por exemplo, para Resolver Exceções e Problemas é importante Desenvolver Plano de Resolução de Problemas previamente. Já as regras exclusivas indicam que quando a característica do antecedente é escolhida, a característica do consequente não pode ser selecionada. Por exemplo, ao optar por Realizar Reunião Diária, não é preciso Monitorar Status do Projeto, pois essas duas características se tornam redundantes. Estas regras já aparecem assinaladas no canto inferior das características da Figura 12.

**Tabela 5** – Regras de composição de características

| # regra | Antecedente | Tipo | Operador | Consequente |
|---|---|---|---|---|
| R | Resolver Exceções e Problemas | Inclusiva | *Requires* | Desenvolver Plano de Resolução de Problemas |
| R_1 | Iniciar Projeto | Inclusiva | *Requires* | Encerrar Projeto |
| R_2 | Atribuir Responsabilidades | Inclusiva | *Requires* | Selecionar Equipe |
| R_3 | Monitoramento e Controle de Projeto | Inclusiva | *Requires* | Monitorar Status do Projeto |
| R_4 | Definir Equipe e Organização do Projeto | Inclusiva | *Requires* | Alocar Tarefas |
| R_5 | Escolher Tarefas | Inclusiva | *Requires* | Realizar Reunião Diária |
| X | Realizar Reunião Diária | Exclusiva | *Excludes* | Monitorar Status do Projeto |

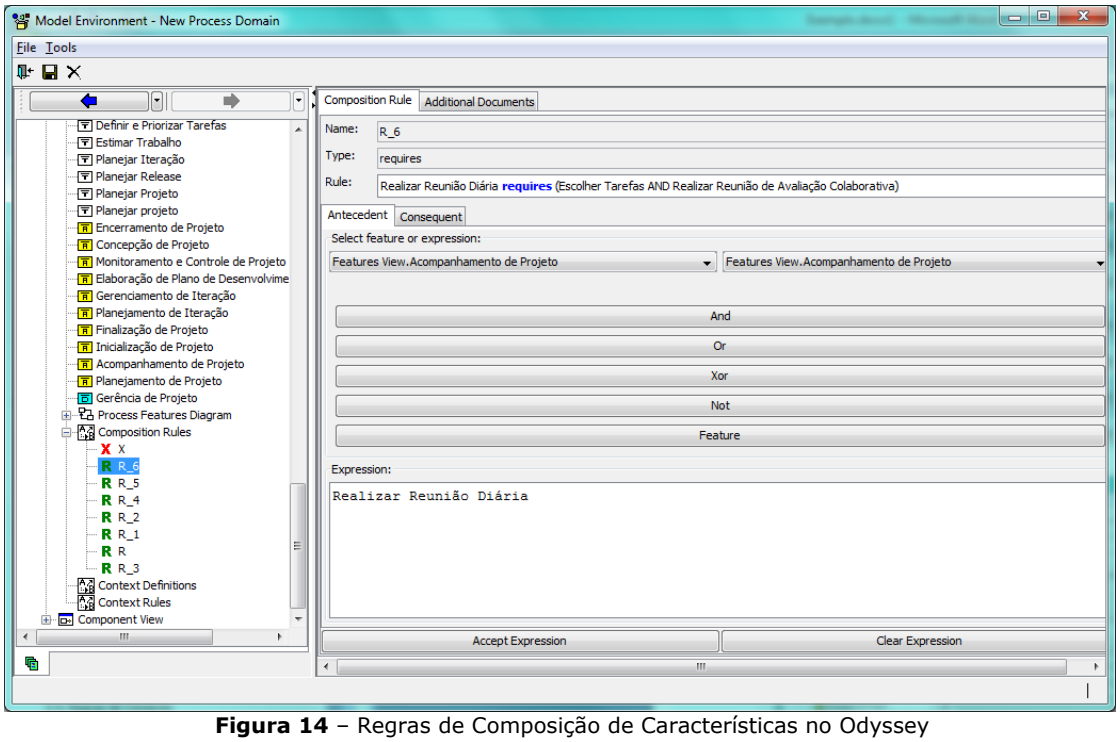


**Figura 14** – Regras de Composição de Características no Odyssey

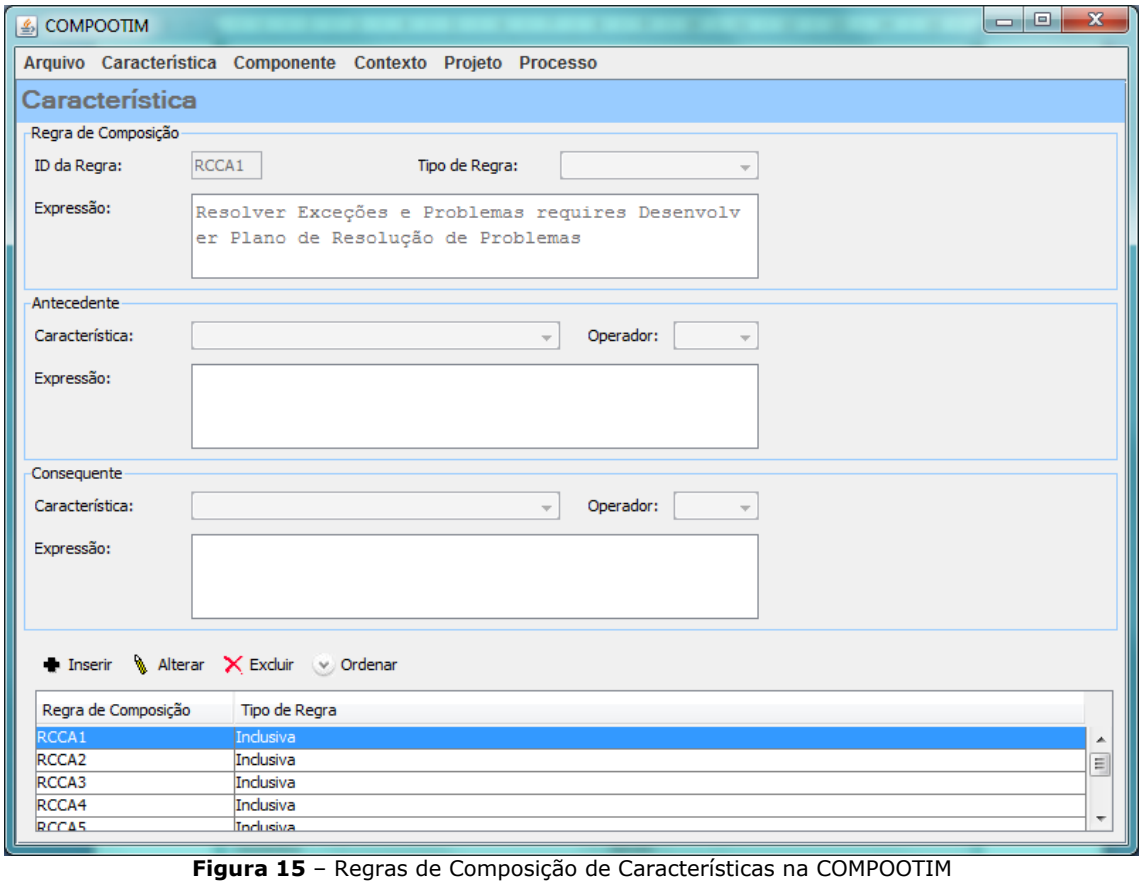


**Figura 15** – Regras de Composição de Características na COMPOOTIM

## 5.5.Modelagem de Contexto

Após a modelagem das características do processo e suas regras de composição, o próximo passo é a modelagem de contexto para a criação dos seus conjuntos de dados. Esta modelagem de contexto se baseou na estrutura de representação de contexto proposta por Fernandes *et al*. (2008) e apresentada na próxima seção.

### *5.5.1.Estrutura de Representação de Contexto*

A abordagem UbiFEX (FERNANDES *et al.*, 2008) tem como objetivo permitir a representação de contexto em modelos de características. Assim, a UbiFEX estende a notação Odyssey-FEX (OLIVEIRA, 2006) através da criação de duas novas características: entidade de contexto e informação de contexto.

A visão geral da abordagem, composta por quatro atividades, é apresentada na Figura 16. A primeira atividade é definir as entidades de contexto. As entidades de contexto (*context entity*) representam as dimensões de contexto relevantes para o domínio.

Em seguida, devem ser definidas as informações de contexto relevantes para o domínio em questão. As informações de contexto (*context information*) representam as características que devem ser coletadas para caracterizar as entidades de contexto do domínio (FERNANDES *et al.*, 2008).

O terceiro passo é criar as definições de contexto (*context definition*), ou seja, caracterizar as situações específicas que podem acontecer. As definições de contexto descrevem situações relevantes para o domínio, tendo como base as entidades e informações de contexto modeladas anteriormente. A UbiFEX permite que uma nova definição de contexto seja definida a partir de outras previamente definidas, promovendo o reuso e facilitando a atividade de manutenção (FERNANDES *et al.*, 2008).

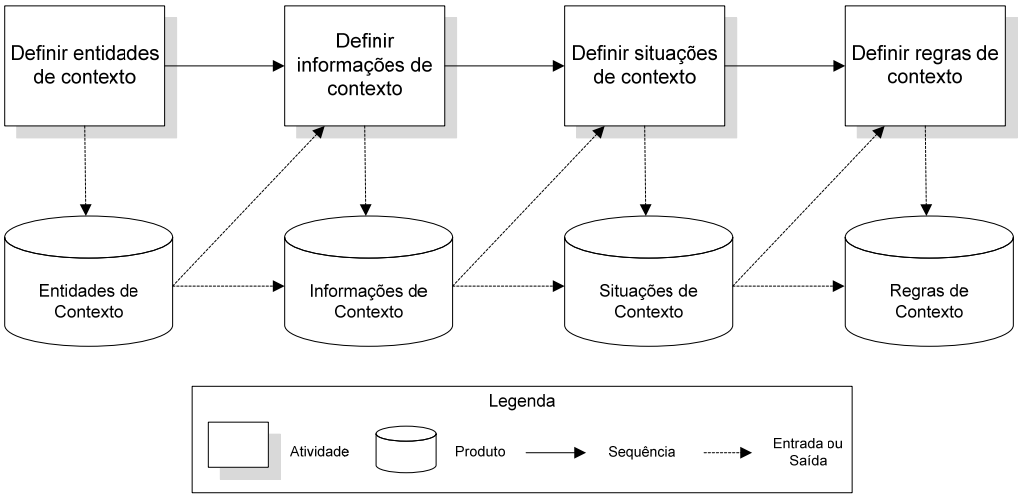


**Figura 16** – Visão geral da abordagem UbiFEX

Por fim, as ações que devem ser tomadas para uma determinada situação podem ser especificadas pelas regras de contexto (*context rules*). Estas regras definem como uma situação de contexto impacta na configuração de um processo, indicando, por exemplo, decisões a respeito da seleção de variantes em um ponto de variação.

As regras de contexto têm como propriedades: um identificador e uma expressão. Essa expressão é formada por um antecedente, o operador "implica" (*implies*) e um consequente. O antecedente pode ser formado por: uma definição de contexto; ou combinação de definições de contexto e operadores lógicos; ou combinação, por meio de operadores lógicos, de definições de contexto e características. Os operadores lógicos utilizados no antecedente podem ser do tipo E (AND), OU (OR), OU-Exclusivo (XOR) ou NÃO (NOT) (FERNANDES *et al.*, 2008).

O consequente, por sua vez, é formado por uma característica ou uma combinação de características. Os operadores lógicos utilizados no consequente podem ser do tipo E (AND) ou NÃO (NOT). A presença de uma característica no consequente indica a sua adição. No entanto, a presença do operador lógico NÃO (NOT) indica a remoção da característica ao qual ele é aplicado (FERNANDES *et al.*, 2008).

Esta abordagem UbiFEX, com todas as suas atividades e produtos, foi utilizada na presente pesquisa e um exemplo de sua aplicação ao domínio de

processos de software é apresentada nas próximas seções. Porém, vale ressaltar que a construção deste modelo de contexto requer o conhecimento de um especialista no domínio para definir de forma apropriada cada um dos seus elementos.

### *5.5.2. Modelo de Contexto para Processos de Software*

O modelo de contexto proposto para processos de software foi apresentado inicialmente em (MAGDALENO, 2010c). Mais recentemente, este modelo foi revisto, formalizado e validado em (LEITE, 2011). Esta seção detalha o modelo de contexto que está efetivamente sendo usado nesse exemplo, pois este modelo vem sendo amadurecido ao longo destes trabalhos. Este modelo segue a estrutura de representação de contexto da UbiFEX (FERNANDES *et al.*, 2008), apresentada na seção anterior.

#### *5.5.2.1. Dimensões de Contexto*

Araujo *et al*. (2004) sugerem as seguintes dimensões de contexto para o desenvolvimento de software: indivíduo, papel, equipe, tarefa, projeto, organização, domínio da engenharia de software, produto de software, domínio do negócio e cliente/usuário (Figura 17).

Ainda que todas as dimensões agreguem informações de contexto importantes, as dimensões organização, projeto e equipe são particularmente relevantes para o presente trabalho de pesquisa e, por isso, estão destacadas na Figura 17. As dimensões organização e projeto devem ser consideradas, visto que a adaptação de processos, geralmente, é realizada nestes dois níveis (MAGDALENO, 2010c, PEDREIRA *et al.*, 2007). A dimensão equipe também é importante, dado o interesse deste trabalho de pesquisa no aspecto da colaboração.

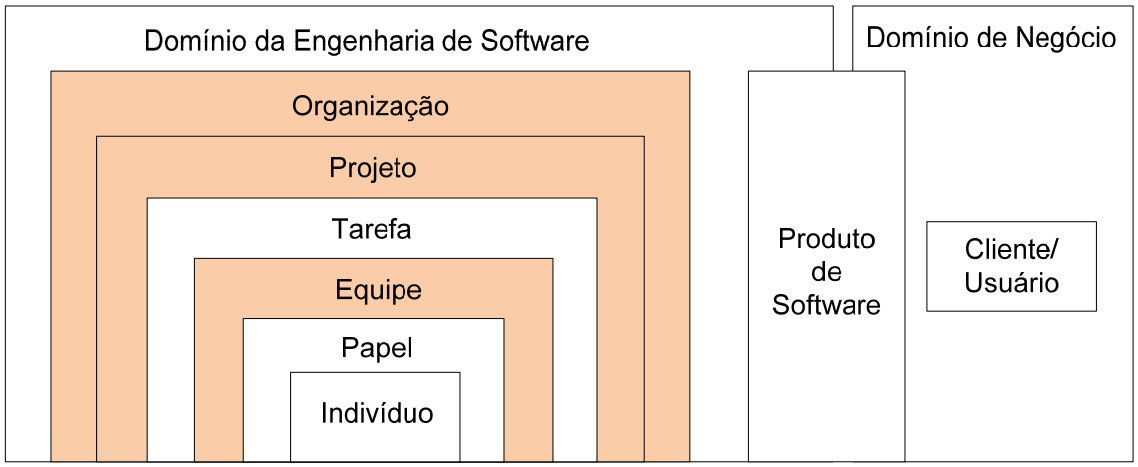


**Figura 17** – Dimensões de contexto no desenvolvimento de software. Adaptado de: Araujo *et al*. (2004)

A dimensão organização descreve a estrutura e a cultura organizacionais e os objetivos de negócio da empresa. A dimensão projeto compreende as características e o escopo do desenvolvimento, incluindo, por exemplo, informações

referentes ao tamanho, duração e complexidade do projeto. A dimensão equipe reflete o contexto resultante da agregação dos participantes da equipe do projeto e compreende informações relativas a experiência, conhecimento e localização dos membros da equipe.

Estas dimensões foram modeladas no Odyssey (Figura 19) e cadastradas na ferramenta COMPOOTIM (Figura 18).

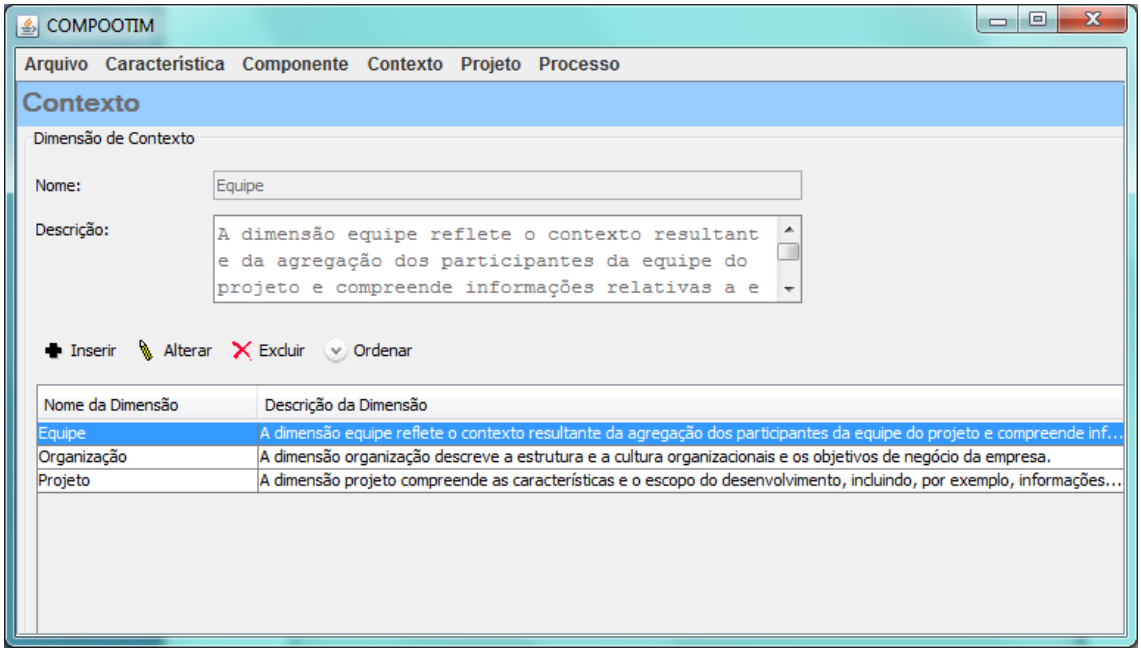


**Figura 18** – Dimensões de contexto na COMPOOTIM

*5.5.2.2.Informações de Contexto*

Para cada uma destas dimensões de contexto, pode-se pensar em um conjunto de informações de contexto, de acordo com os aspectos de colaboração e disciplina dos processos de software. A determinação das informações de contexto não é uma atividade trivial (ROSEMANN e RECKER, 2006, SAIDANI e NURCAN, 2007). Ainda não existe um consenso sobre quais conhecimentos são considerados contextuais. Vários tipos de informações contribuem com o contexto e a relevância de cada informação, depende do foco de atenção ou da tarefa em questão (BREZILLON, 1999).

Para o objetivo deste trabalho de pesquisa, é suficiente identificar as informações de contexto relevantes para caracterizar um projeto e balancear a colaboração e a disciplina no domínio de desenvolvimento de software. Assim, um conjunto inicial de informações de contexto é apresentado nas próximas seções. Estes exemplos de informações de contexto foram obtidos através da revisão e adaptação de trabalhos já publicados na literatura (ARMBRUST *et al.*, 2008, BEKKERS *et al.*, 2008, BERGER, 2003, BOEHM e TURNER, 2003, BURNS e DENNIS, 1985, COCKBURN, 2000, 2001, GINSBERG e QUINN, 1995, HENDERSON-SELLERS, 2002, LAANTI e KETTUNEN, 2005, LITTLE, 2005, MACHADO, 2000, MNKANDLA,

2008, PARK *et al.*, 2006, QUMER e HENDERSON-SELLERS, 2008, SANTOS, 2009, SLOOTEN e BRINKKEMPER, 1993, WEERD, 2009, XU e RAMESH, 2008) ou através da própria experiência da autora deste trabalho em definição de processos de desenvolvimento de software. Estas informações de contexto estão descritas em (LEITE, 2011, MAGDALENO, 2010c).

A Tabela 6 resume as informações de contexto e seus atributos. Em relação à granularidade e complexidade, uma informação de contexto pode ser ou não atômica. Uma informação de contexto atômica já representa a menor unidade. Quando a informação de contexto não é atômica, ela é composta por outras informações de contexto. Vale ressaltar ainda que algumas informações de contexto são mais estáticas (como a estrutura organizacional), enquanto outras têm um caráter mais dinâmico (como a estabilidade dos requisitos).

**Tabela 6** – Informações de contexto

| Dimensão | Informações de Contexto | Composição | Valores |
|---|---|---|---|
| **Organização** | Estrutura organizacional | Complexidade | - Muito baixa;<br>- Baixa;<br>- Média;<br>- Alta;<br>- Muito alta |
| | | Rigidez | - Muito baixa;<br>- Baixa;<br>- Média;<br>- Alta;<br>- Muito alta |
| | Cultura organizacional | Tipo | - Tradicional;<br>- Democrática |
| | | Tomada de decisão | - Descentralizada;<br>- Centralizada |
| | Objetivo de negócio | *Atômica* | - Inovação;<br>- Conquista de um novo segmento de mercado;<br>- Aumento da qualidade;<br>- Diminuição de custos |
| **Projeto** | Relacionamento com o cliente | Nível de envolvimento | - Muito alto;<br>- Alto;<br>- Médio;<br>- Baixo;<br>- Muito baixo |
| | | Disponibilidade | - Muito alta;<br>- Alta;<br>- Média;<br>- Baixa;<br>- Muito baixa |
| | | Nível de confiança | - Muito alta;<br>- Alta;<br>- Média;<br>- Baixa;<br>- Muito baixa |

| Dimensão | Informações de Contexto | Composição | Valores |
|---|---|---|---|
| | | Número de representantes | - Um;<br>- Múltiplos |
| | | Experiência no domínio | - Muito alta - especialista;<br>- Alta - muito experiente;<br>- Média - experiente;<br>- Baixa - pouco experiente;<br>- Muito baixa - sem experiência |
| | | Habilidade de expressar requisitos | - Muito alta - expressivo;<br>- Alta - muito comunicativo;<br>- Média - comunicativo;<br>- Baixa - pouco comunicativo;<br>- Muito baixa - sem habilidade |
| | Tamanho do problema | *Atômica* | - Muito pequeno;<br>- Pequeno;<br>- Médio;<br>- Grande;<br>- Muito grande |
| | Complexidade do projeto | *Atômica* | - Muito Baixa - projetos que não envolvem funcionalidades simples;<br>- Baixa - projetos que envolvem funcionalidades pouco complexas;<br>- Média - projetos que envolvem funcionalidades complexas;<br>- Alta - projetos que envolvem funcionalidades razoavelmente complexas;<br>- Muito alta - projetos que envolvem funcionalidades muito complexas |
| | Duração do Projeto | *Atômica* | - Muito curta – < 2 meses;<br>- Curta – De 2 a 6 meses;<br>- Média – De 6 a 12 meses;<br>- Longa – De 12 a 24 meses;<br>- Muito longa – > 24 meses |
| | Criticidade do Projeto | *Atômica* | - Muito baixa – perda de conforto;<br>- Baixa - perda de valores financeiros discretos e facilmente recuperáveis;<br>- Média - perdas moderadas, mas |

| Dimensão | Informações de Contexto | Composição | Valores |
|---|---|---|---|
| | | | recuperáveis;<br>- Alta - perda de quantias significativas ou de funções críticas do negócio;<br>- Muito alta - perda de vidas humanas ou descontinuidade do negócio |
| | Novidade do Projeto | *Atômica* | - Muito alta;<br>- Alta;<br>- Média;<br>- Baixa;<br>- Muito baixa |
| | Tecnologia de desenvolvimento do produto | Maturidade | - Muito alta;<br>- Alta;<br>- Média;<br>- Baixa;<br>- Muito baixa |
| | Ambiente tecnológico de desenvolvimento | Maturidade | - Muito alta;<br>- Alta;<br>- Média;<br>- Baixa;<br>- Muito baixa |
| | Estabilidade dos requisitos | *Atômica* | - Muito Baixa - requisitos muito instáveis;<br>- Baixa - requisitos instáveis;<br>- Média - requisitos com grau médio de estabilidade;<br>- Alta - requisitos praticamente estáveis;<br>- Muito alta - requisitos estáveis |
| **Equipe** | Tamanho da equipe | *Atômica* | - Muito grande – > 80 pessoas;<br>- Grande – 40 a 80 pessoas;<br>- Média – 11 a 39 pessoas;<br>- Pequena – 7 a 10 pessoas;<br>- Muito pequena – < 6 pessoas |
| | Experiência técnica | Experiência no domínio | - Muito alta – > 6 anos;<br>- Alta – 4 a 6 anos;<br>- Média – 2 a 4 anos;<br>- Baixa – 1 a 2 anos;<br>- Muito baixa – < 1 ano |
| | | Experiência no desenvolvimento de software | - Muito alta – > 6 anos;<br>- Alta – 4 a 6 anos;<br>- Média – 2 a 4 anos;<br>- Baixa – 1 a 2 anos; |

| Dimensão | Informações de Contexto | Composição | Valores |
|---|---|---|---|
| | | | - Muito baixa – < 1 ano |
| | | Experiência na tecnologia | - Muito alta – > 6 anos;<br>- Alta – 4 a 6 anos;<br>- Média – 2 a 4 anos;<br>- Baixa – 1 a 2 anos;<br>- Muito baixa – < 1 ano |
| | Experiência gerencial | *Atômica* | - Muito alta - especialista;<br>- Alta - muito experiente;<br>- Média - experiente;<br>- Baixa - pouco experiente;<br>- Muito baixa - sem experiência |
| | Experiência no trabalho em equipe | *Atômica* | - Muito alta;<br>- Alta;<br>- Média;<br>- Baixa;<br>- Muito baixa |
| | Proximidade da equipe | *Atômica* | - Localizada – equipe reunida no mesmo local de trabalho;<br>- Distribuída – equipe dispersa em diferentes localizações |
| | Estabilidade da equipe | *Atômica* | - Muito alta;<br>- Alta;<br>- Média;<br>- Baixa;<br>- Muito baixa |

Estas informações de contexto foram modeladas no Odyssey (Figura 19) e cadastradas na ferramenta COMPOOTIM (Figura 20).

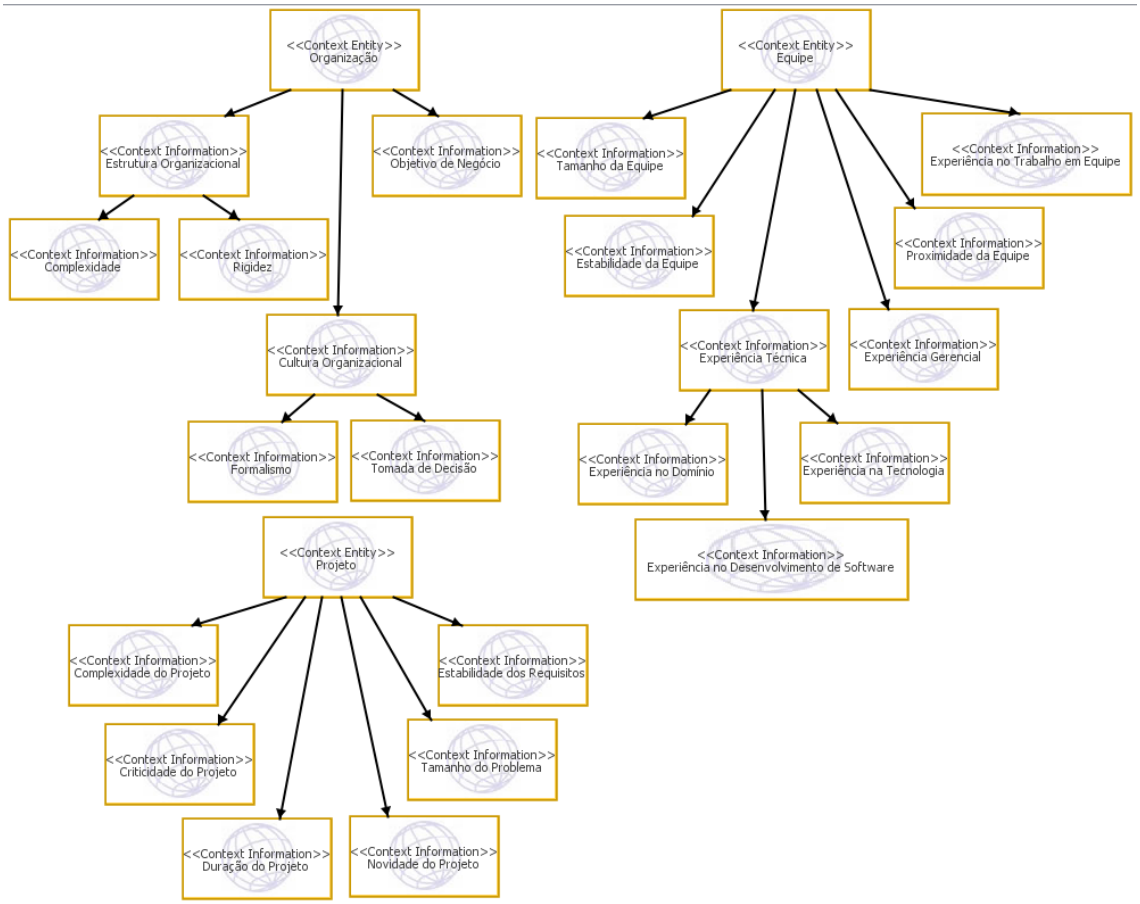


**Figura 19** – Diagrama parcial das dimensões e informações de contexto no Odyssey

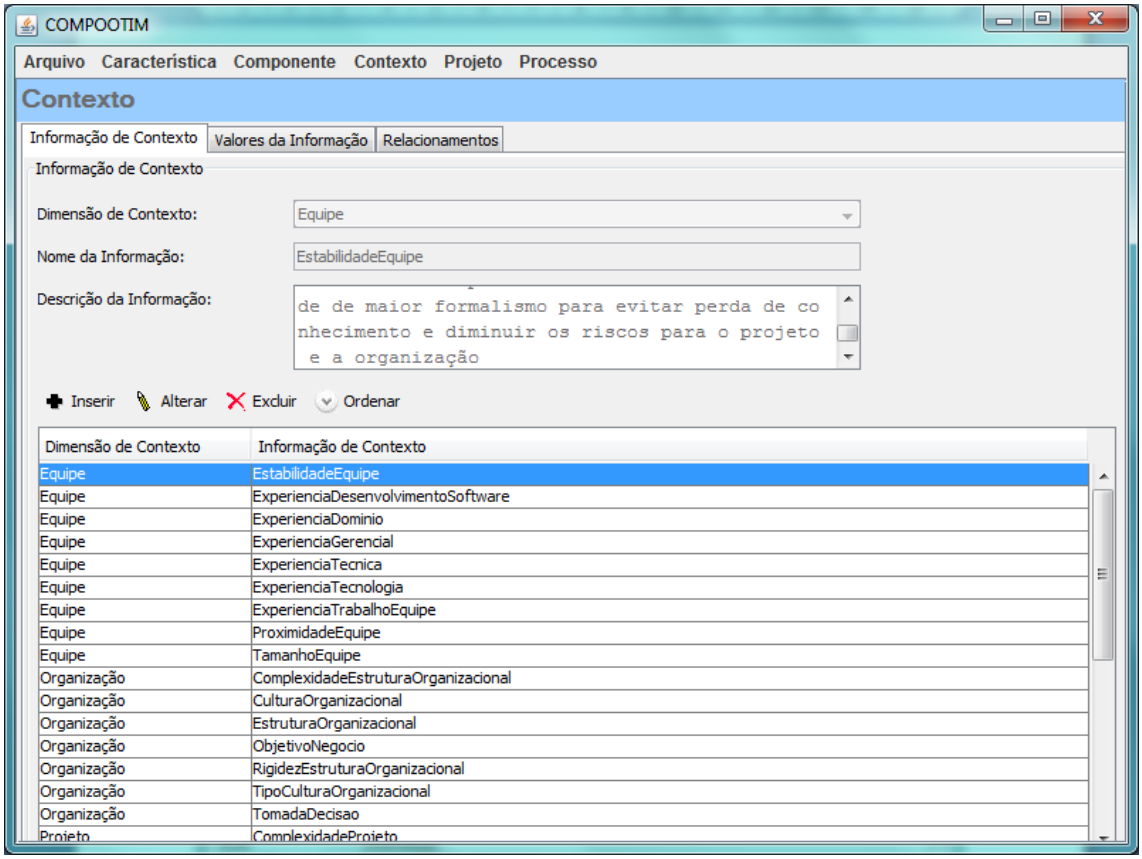


**Figura 20** – Informações de contexto na COMPOOTIM

*5.5.2.3.Situações de Contexto*

As situações de contexto representam circunstâncias que podem acontecer baseadas na combinação de valores de determinadas informações de contexto. As situações de contexto definidas dentro do domínio de processos de software, foram agrupadas pelas dimensões de contexto. As situações de contexto usadas como exemplo são apresentadas abaixo e foram adaptadas e evoluídas a partir do trabalho de Leite (2011).

**a) Situações de Contexto de Equipe**

As situações de contexto da dimensão equipe são apresentadas na Tabela 7 e brevemente descritas a seguir.

- Equipe Inexperiente – equipe com pouca ou nenhuma experiência.
- Equipe em Aprendizado – equipe que possui algum nível de experiência.
- Equipe Experiente – equipe com alta experiência.
- Equipe Próxima – equipe fisicamente concentrada em um único local.
- Equipe Dispersa – equipe geograficamente distribuída.

**Tabela 7** – Situações de contexto de equipe

| Informação de Contexto | Equipe Inexperiente | Equipe em Aprendizado | Equipe Experiente | Equipe Próxima | Equipe Dispersa |
|---|---|---|---|---|---|
| **Experiência no Domínio** | Muito Baixa / Baixa | Média | Muito Alta / Alta | - | - |
| **Experiência no Desenvolvimento de Software** | Muito Baixa / Baixa | Média | Muito Alta / Alta | - | - |
| **Experiência na Tecnologia** | Muito Baixa / Baixa | Média | Muito Alta / Alta | - | - |
| **Experiência no Trabalho em Equipe** | Muito Baixa / Baixa | Média | Muito Alta / Alta | - | - |
| **Proximidade da Equipe** | - | - | - | Localizada | Distribuída |

**b) Situações de Contexto de Projeto**

As situações de contexto da dimensão projeto são apresentadas na Tabela 8 e brevemente descritas a seguir.

- Projeto Desafiador – projeto cuja natureza pode ser desconhecida pela equipe, onde a complexidade é alta, o tamanho do projeto é grande e a duração é longa.
- Projeto Típico – projeto mediano em complexidade, tamanho e desafios.
- Projeto Simples – projetos cuja natureza é conhecida pela equipe.
- Projeto Inovador – projeto com um caráter de novidade, onde os requisitos mudam constantemente.

**Tabela 8** – Situações de contexto de projeto

| Informação de Contexto | Projeto Desafiador | Projeto Típico | Projeto Simples | Projeto Inovador |
|---|---|---|---|---|
| **Complexidade do Projeto** | Muito Alta / Alta | Média | Muito Baixa / Baixa | - |
| **Criticidade do Projeto** | Muito Alta / Alta | Média | Muito Alta / Alta | - |
| **Tamanho do Problema** | Muito Grande / Grande | Médio | Muito Pequeno / Pequeno | - |
| **Duração do Projeto** | Muito Longa / Longa | Média | Muito Curta / Curta | - |
| **Novidade do Projeto** | - | Média | Muito Baixa / Baixa | Muito Alta / Alta |
| **Estabilidade dos Requisitos** | | Média | Muito Alta / Alta | Muito Baixa / Baixa |

Também foram criadas algumas situações de contexto para caracterizar o relacionamento com o cliente no projeto (Tabela 9):

- Cliente Ausente – cliente não presente para a interação com a equipe do projeto.
- Cliente Participativo – cliente altamente participativo e disponível para a equipe do projeto.
- Cliente Especialista – cliente com grande conhecimento no domínio do negócio.
- Cliente Novato – cliente com pouco conhecimento no domínio do negócio.
- Cliente Articulado – cliente com grande habilidade para expressar os requisitos desejados.

**Tabela 9** – Situações de contexto do relacionamento com o cliente no projeto

| Informação de Contexto | Cliente Ausente | Cliente Participativo | Cliente Especialista | Cliente Novato | Cliente Articulado |
|---|---|---|---|---|---|
| **Nível de Envolvimento do Cliente** | Muito Baixo / Baixo | Muito Alto / Alto | Muito Baixa / Baixa | - | - |
| **Disponibilidade do Cliente** | Muito Baixa / Baixa | Muito Alta / Alta | Muito Alta / Alta | - | - |
| **Experiência do Cliente no Domínio** | - | - | Muito Alta / Alta | Muito Baixa / Baixa | - |
| **Habilidade em Expressar Requisitos** | - | - | - | - | Muito Alta / Alta |

### c) Situações de Contexto de Organização

As situações de contexto da dimensão organização são apresentadas na Tabela 10 e brevemente descritas a seguir.

- Organização Conservadora – organização com uma estrutura complexa e rígida, uma cultura tradicional e a tomada de decisão centralizada.

- Organização Moderada – organização cujos objetivos aceitam algumas inovações, mas com pouco risco.
- Organização Arrojada – organização aberta à inovação.

**Tabela 10** – Situações de contexto da organização

| Informação de Contexto | Organização Conservadora | Organização Moderada | Organização Arrojada |
|---|---|---|---|
| **Complexidade da Estrutura Organizacional** | Muito Alta / Alta | Média | Muito Baixa / Baixa |
| **Rigidez da Estrutura Organizacional** | Muito Alta / Alta | Média | Muito Baixa / Baixa |
| **Tipo da Cultura Organizacional** | Tradicional | Tradicional | Democrática |
| **Tomada de Decisão** | Centralizada | Centralizada | Descentralizada |

Estas situações de contexto foram modeladas no Odyssey (Figura 21) e cadastradas na ferramenta COMPOOTIM (Figura 22).

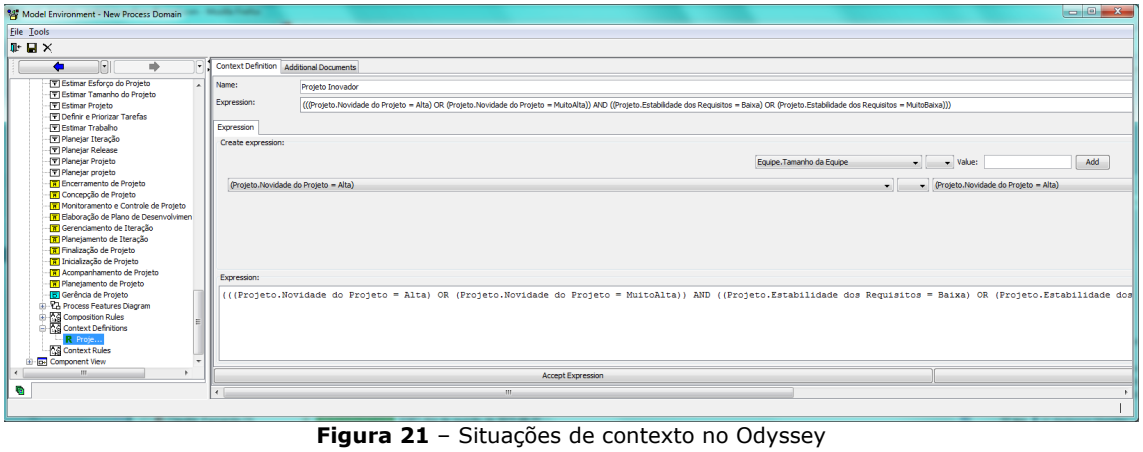


**Figura 21** – Situações de contexto no Odyssey

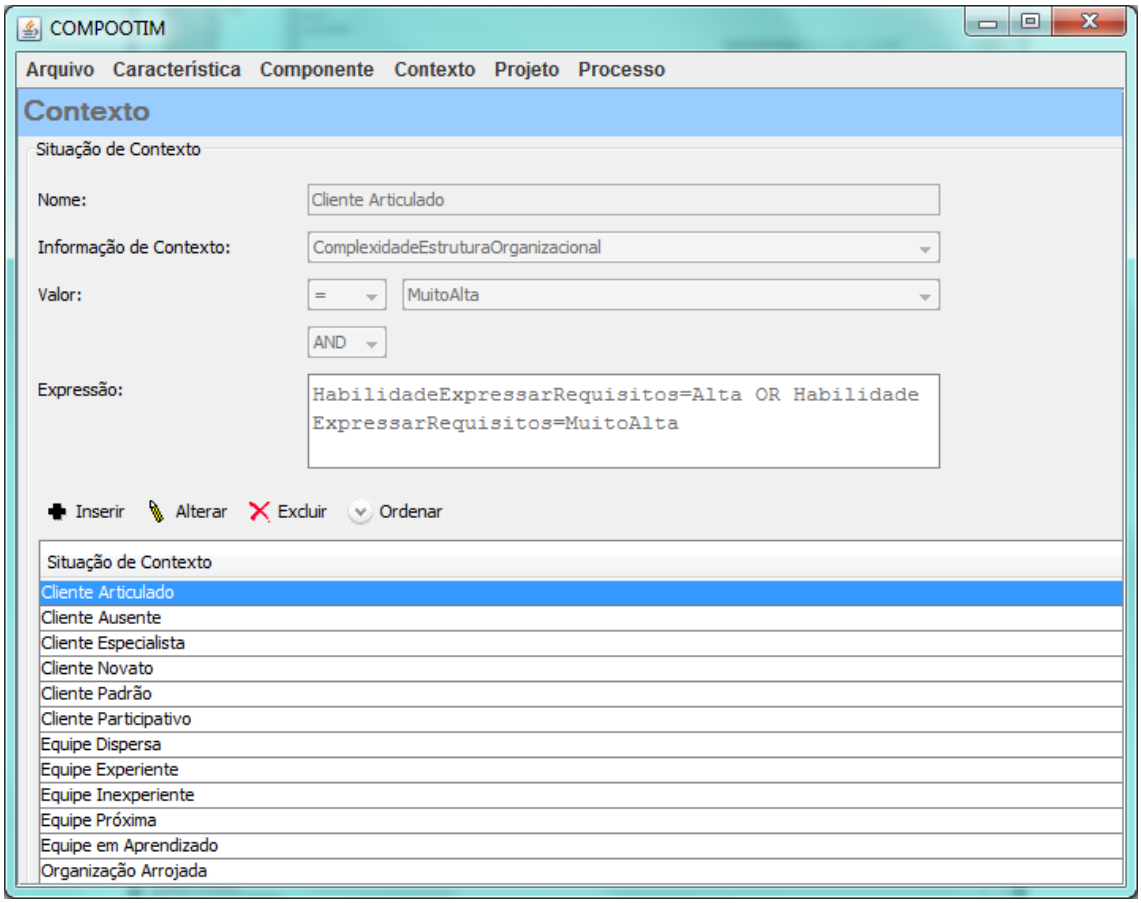


**Figura 22** – Situações de contexto na COMPOOTIM

### *5.5.2.4. Regras de Contexto*

Após a definição dos contextos relevantes para o domínio, as ações que devem ser tomadas caso uma determinada situação ocorra podem ser especificadas por meio das regras de contexto. Para ajudar a composição do processo, é necessário estabelecer regras de contexto que relacionem as situações de contexto à seleção de práticas do modelo de características da linha de processos.

Alguns exemplos de regras de contexto foram construídos utilizando parte das situações de contexto listadas na seção anterior e o modelo de características da linha de processos, apresentado na Figura 12. A Tabela 11 resume estes exemplos de regras de contexto. Estas regras foram modeladas no Odyssey (Figura 23) e na COMPOOTIM (Figura 24).

**Tabela 11** – Regras de contexto

| # regra | Situação de Contexto | Operador | Característica |
|---|---|---|---|
| **CR1** | Equipe Experiente | *Implies* | Escolher Tarefas |
| **CR2** | Equipe Próxima | *Implies* | Realizar Reunião Diária |
| **CR3** | Equipe Dispersa | *Implies* | Definir Equipe e Organização do Projeto |

| # regra | Situação de Contexto | Operador | Característica |
|---|---|---|---|
| CR4 | Organização Conservadora | *Implies* | Elaborar Relatório Mensal AND Realizar Avaliação Gerencial Individual |
| CR5 | Cliente Ausente | *Implies* | Desenvolver Plano de Gerenciamento de Riscos AND Desenvolver Plano de Aceitação de Produto |

Vale ressaltar que estas regras de contexto foram derivadas empiricamente com base no conhecimento e experiência da autora, mas ainda é necessário mais trabalho para estabelecer correlações válidas entre as situações de contexto e as práticas de desenvolvimento de software.

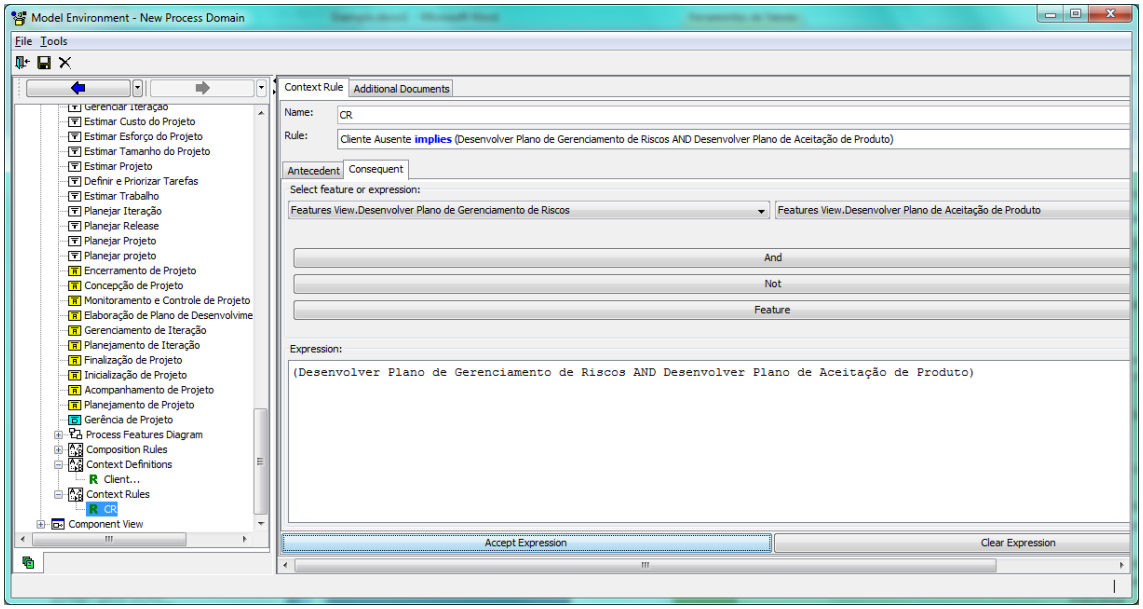


**Figura 23** – Regras de contexto no Odyssey

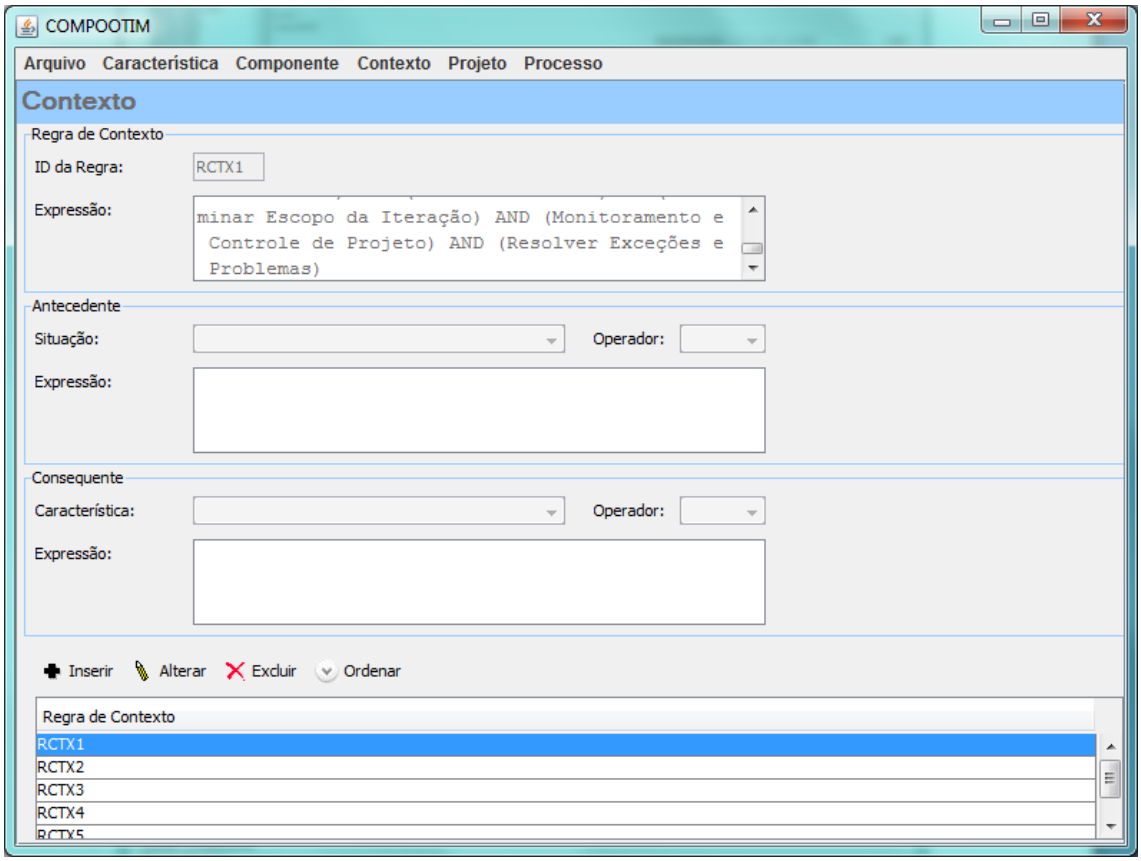


**Figura 24** – Regras de contexto na COMPOOTIM

# 6.Uso da linha de processos baseada em contexto

Um novo projeto de desenvolvimento de software, chamado CDSoft, pretende criar um produto inovador. Devido à novidade do produto, ele também tem requisitos voláteis que mudam com frequência. Por causa desta instabilidade de requisitos, a documentação torna-se rapidamente obsoleta e a necessidade de compartilhamento de conhecimento entre os membros da equipe é intensificada. O gerente de projeto precisa definir um processo para esse projeto utilizando a linha de processos apresentada na seção anterior.

## 6.1.Caracterização de Projeto

Para utilizar todos esses conjuntos de dados definidos nas seções anteriores, o primeiro passo é caracterizar o contexto de um projeto. O gerente informa o contexto do projeto, atribuindo valores para informações de contexto. A Tabela 12 resume o contexto do projeto CDSoft, de acordo com as informações de contexto sugeridas na Tabela 6.

Este projeto também foi cadastrado na COMPOOTIM. Os dados básicos do projeto são apresentados na Figura 25. Os valores das informações de contexto deste projeto são apresentados na Figura 26.

**Tabela 12** – Caracterização do contexto do projeto

| Dimensão | Informações de Contexto | Valores |
|---|---|---|
| **Organização** | Estrutura organizacional - Complexidade | Muito alta |
| | Estrutura organizacional - Rigidez | Muito alta |
| | Cultura organizacional – Tipo | Tradicional |
| | Cultura organizacional - Tomada de decisão | Centralizada |
| | Objetivo de negócio | Aumento da qualidade |
| **Projeto** | Relacionamento com cliente - Nível de envolvimento | Muito alto |
| | Relacionamento com cliente - Disponibilidade | Alta |
| | Relacionamento com cliente - Nível de confiança | Alta |
| | Relacionamento com cliente - Número de representantes | Um |
| | Relacionamento com cliente - Experiência no domínio | Muito alta |
| | Relacionamento com cliente - Habilidade de expressar requisitos | Média |
| | Tamanho do problema | Médio |
| | Complexidade do projeto | Alta |
| | Duração do Projeto | Longa |
| | Criticidade do Projeto | Média |
| | Novidade do Projeto | Alta |
| | Tecnologia de desenvolvimento do produto - Maturidade | Alta |
| | Ambiente tecnológico de desenvolvimento - Maturidade | Alta |
| | Estabilidade dos requisitos | Baixa |
| **Equipe** | Tamanho da equipe | Médio |
| | Experiência técnica - Experiência no domínio | Média |
| | Experiência técnica - Experiência no desenvolvimento de software | Média |
| | Experiência técnica - Experiência na tecnologia | Média |
| | Experiência gerencial | Muito alta |
| | Experiência no trabalho em equipe | Média |
| | Proximidade da equipe | Localizada |
| | Estabilidade da equipe | Baixa |

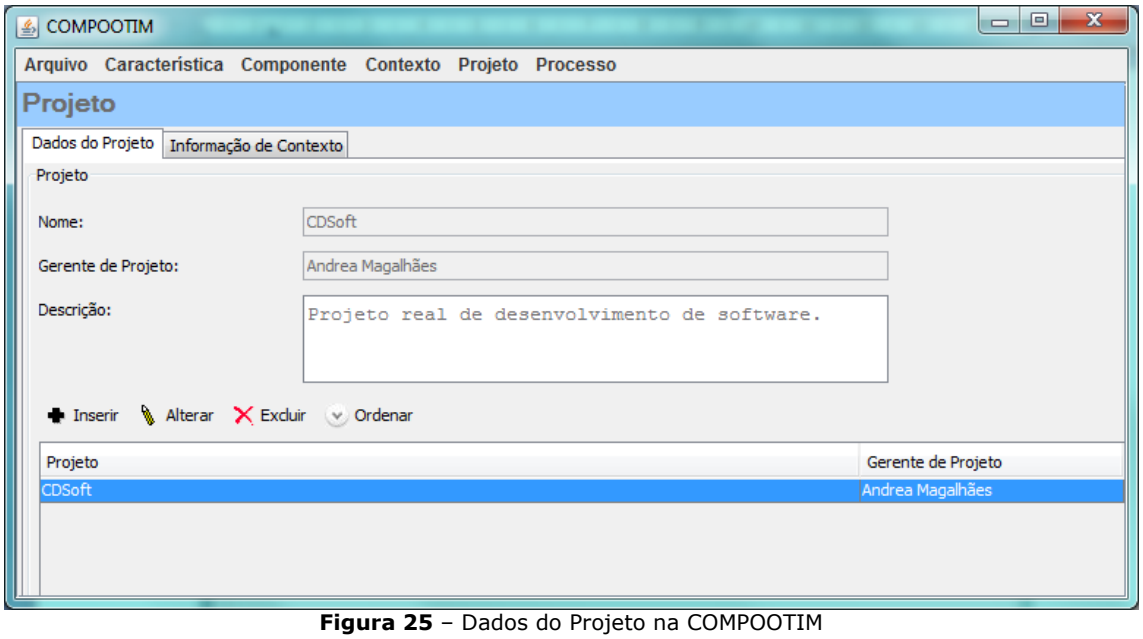


**Figura 25** – Dados do Projeto na COMPOOTIM

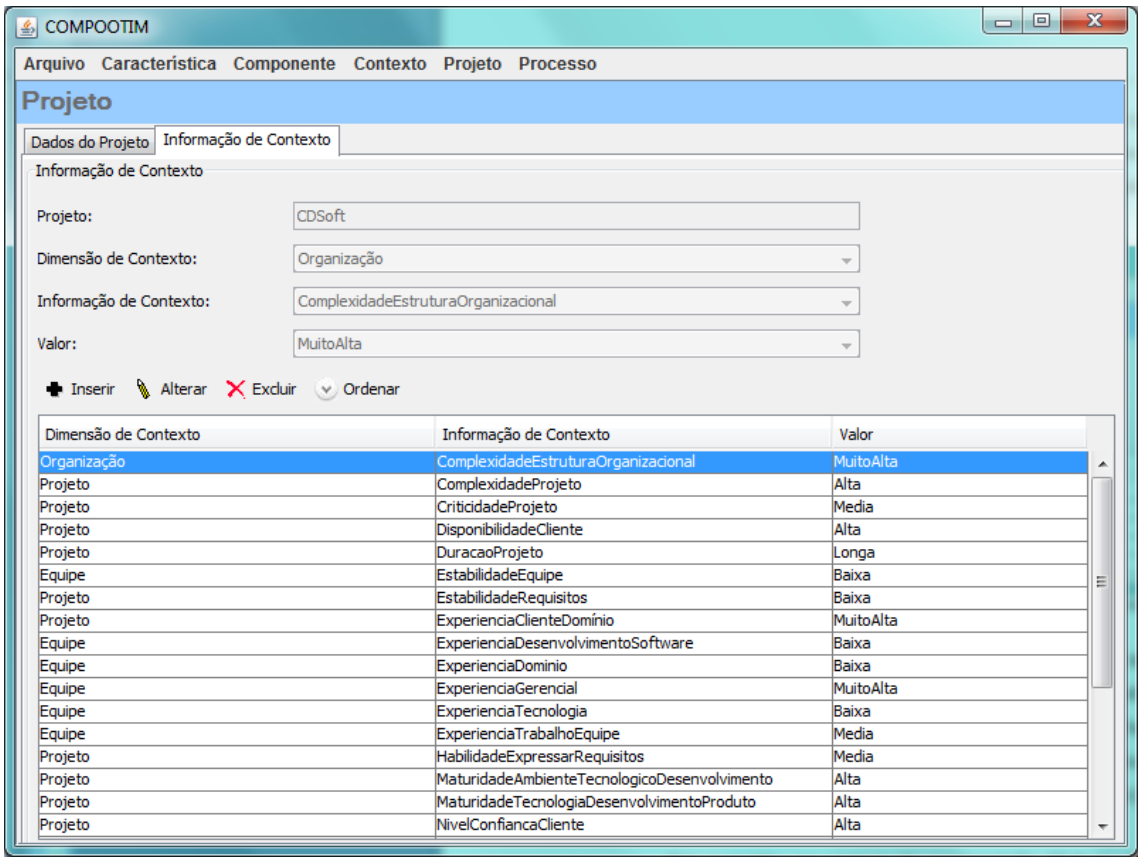


**Figura 26** – Informações do Contexto do Projeto na COMPOOTIM

Com base no contexto do projeto e na linha de processos criada, o mecanismo de otimização da COMPOOTIM ajudará o gerente de projeto a compor um processo específico para o projeto, aderente ao contexto atual. Ele começa descobrindo a situação atual do contexto do projeto a partir da combinação das informações de contexto do projeto (Tabela 12) e as situações de contexto pré-

definidas (Seção 5.5.2.3). Assim, o projeto CDSoft é caracterizado pelas seguintes situações: (i) cliente especialista; (ii) cliente participativo; (iii) equipe próxima; (iv) equipe em aprendizado; (v) organização conservadora; (vi) projeto inovador (Figura 27).

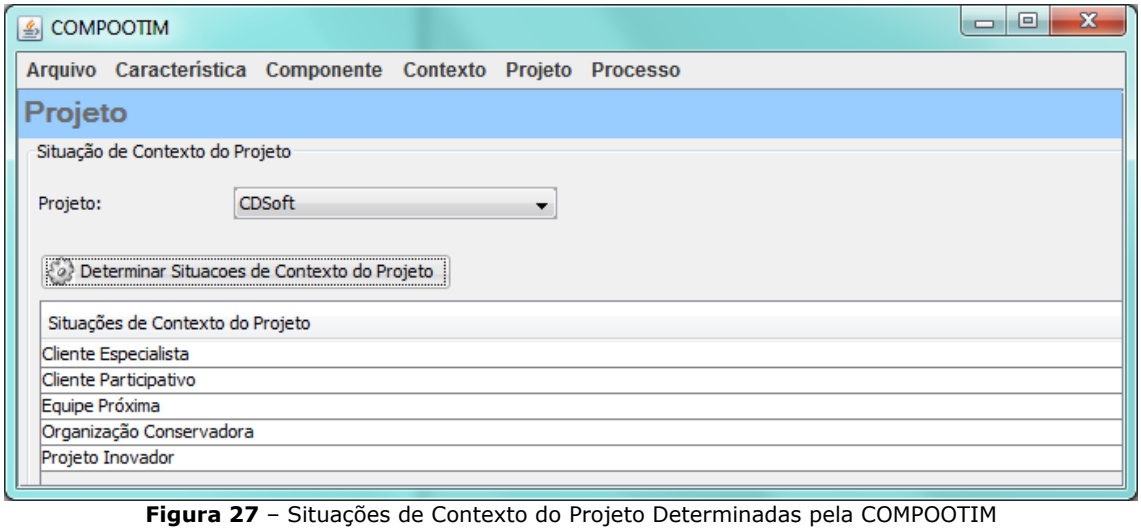


**Figura 27** – Situações de Contexto do Projeto Determinadas pela COMPOOTIM

A partir daí, são inferidas com base nas regras de contexto (Tabela 11), algumas características do processo que devem estar presentes no processo resultante: (i) Elaborar Relatório Mensal; (ii) Realizar Avaliação Gerencial Individual; (iii) Realizar Reunião Diária.

Em seguida, as regras de composição (Tabela 5) são aplicadas e a última exclui a tarefa Monitorar Status do Projeto. Há também algumas decisões sobre configuração de variabilidades e opcionalidades que o gerente do projeto deverá tomar com base na sua percepção. Por exemplo, o gerente de projeto pode: escolher que tipo de estimativa prefere usar (como o esforço); decidir não ter um início ou encerramento formal do projeto; e optar por trabalhar com um mínimo de atividades, ou seja, sem planos desnecessários e revisões.

Tudo isso significa que o processo composto se encaixará no contexto de execução do projeto em alto grau. O modelo de características do processo recortado é apresentado na Figura 28. A seleção dessas características também vai influenciar a seleção ou exclusão dos componentes de processo nas etapas futuras.

A partir daí, o processo pode ser instanciado. Durante a execução do processo, o gerente de projeto, ajudado pelo mecanismo de monitoramento automatizado, pode acompanhar se o contexto do projeto mudou. Assim, o gerente de projeto pode decidir adaptar dinamicamente o processo (durante a execução do projeto), substituindo características ou componentes de processo.

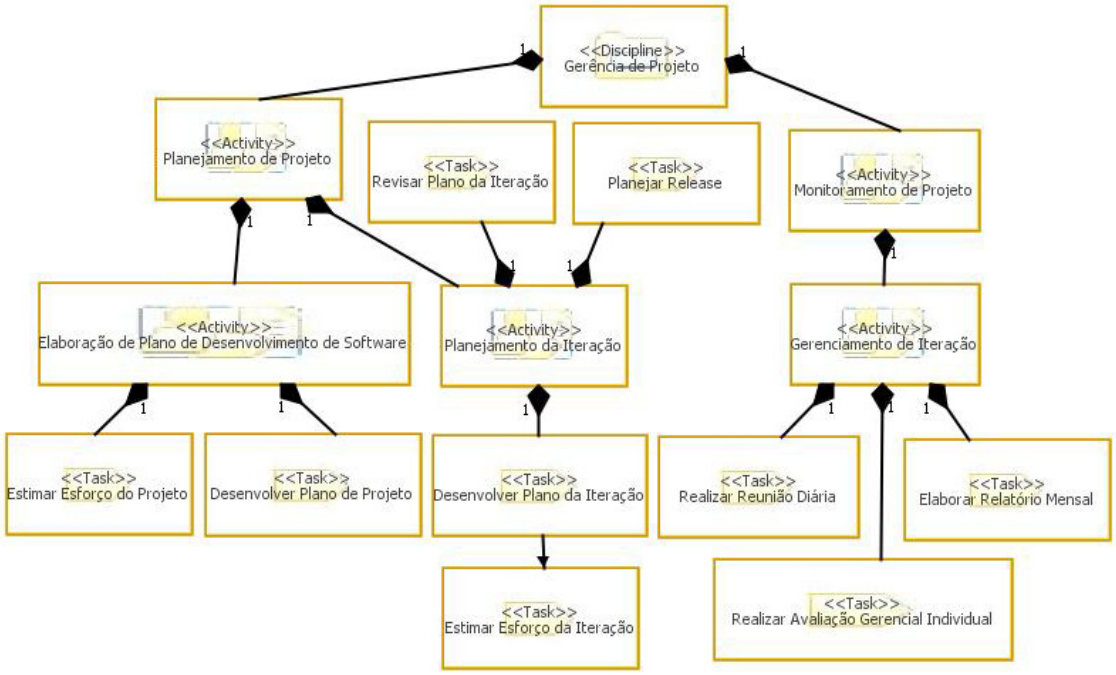


**Figura 28** – Características de processo específicas do projeto CDSoft

# 7.Conclusão

Este trabalho apresentou a abordagem COMPOOTIM, que é baseada em uma estrutura de linha de processo baseado em contexto. A ideia principal é apoiar as decisões gerentes de projeto, automatizando e otimizando a composição do processo específico para o projeto.

Este apoio deve fornecer ao gerente um embasamento para a tomada de decisão sobre a melhor forma de compor o processo específico para o projeto, uma vez que possibilita que seja considerado um universo maior de fatores e alternativas. O apoio sugerido diz respeito a propor ao gerente de projeto um processo que satisfaça as restrições e que otimize alguns dos fatores envolvidos no problema. Porém, a decisão final sobre o processo que será efetivamente adotado continua sendo dele.

Como resultado, espera-se que a complexidade e o esforço necessários para composição de processos seja reduzida. Hollenbach e Frakes (1996) mostram que é possível reduzir em pelo menos dez vezes o tempo e o esforço necessários para definir um processo específico para um projeto quando o processo é reutilizado em vez de criá-lo a partir do zero.

Como trabalho futuro, está previsto validar, através de estudos experimentais, a aplicabilidade da solução em desenvolvimento de software reais através da realização de alguns estudos de caso. Essas validações irão fornecer *feedback* para melhorar a abordagem.

# Agradecimentos

Este trabalho é parcialmente financiado pelo CNPq (sob o processo no. 142006/2008-4).

# Referências Bibliográficas

AALST, W. M. P.; GIINTHER, C. W., 2007, "Finding Structure in Unstructured Processes: The Case for Process Mining". *Application of Concurrency to System Design (ACSD)*, p. 3-12, Bratislava, Slovakia.

ALEIXO, F. A.; FREIRE, M. A.; SANTOS, W. C. *et al.*, 2010, "Uma Abordagem para Gerência e Customização de Variabilidades em Processos de Software". *Simpósio Brasileiro de Engenharia de Software (SBES) - Congresso Brasileiro de Software: Teoria e Prática (CBSOFT)*, p. 119-128, Salvador, BA, Brasil.

ARAUJO, R. M. DE; SANTORO, F. M.; BRÉZILLON, P. *et al.*, 2004, "Context Models for Managing Collaborative Software Development Knowledge". In: *International Workshop on Modeling and Retrieval of Context (MRC)*, p. 61-72, Ulm.

ARMBRUST, O.; KATAHIRA, M.; MIYAMOTO, Y. *et al.*, 2008, "Scoping Software Process Models - Initial Concepts and Experience from Defining Space Standards", *Making Globally Distributed Software Development a Success Story*, Berlin / Heidelberg: Springer, p. 160-172.

BARNETT, L., 2004, "Applying Open Source Processes In Corporate Development Organizations", *Forrester Research*, p. 1-15.

BARRETO, A.; ROCHA, A. R.; MURTA, L., 2010, "Supporting the Definition of Software Processes at Consulting Organizations via Software Process Lines". In: *International Conference on the Quality of Information and Communications Technology (QUATIC)*, p. 15-24, Porto, Portugal.

BECK, K., 2004, *Extreme Programming Explained: Embrace Change*. 2 ed. Boston, MA, USA, Addison-Wesley.

BECK, K.; BEEDLE, M.; BENNEKUM, A. VAN; *et al.*, 2001. Manifesto for Agile Software Development. Disponivel em: http://agilemanifesto.org/. Acesso em: 15 dez 2008.

BEKKERS, W.; VAN DE WEERD, I.; BRINKKEMPER, S. *et al.*, 2008, "The Influence of Situational Factors in Software Product Management: An Empirical Study". *International Workshop on Software Product Management (IWSPM)*, p. 41-48, Barcelona, Catalonia, Spain.

BERGER, P. M., 2003, *Instanciação de Processos de Software em Ambientes Configurados na Estação TABA*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BOEHM, B.; TURNER, R., 2003, *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA, USA, Addison-Wesley.

BOSCH, J., 2004, "Software Variability Management". In: *International Conference on Software Engineering (ICSE)*, p. 720-721, Scotland, UK.

BREZILLON, P., 1999, "Context in problem solving: a survey", *Knowledge Engineering Review*, v. 14, n. 1, p. 47-80.

BRINKKEMPER, S., 1996, "Method engineering: engineering of information systems development methods and tools", *Information and Software Technology*, v. 38, n. 4, p. 275-280.

BURNS, R. N.; DENNIS, A. R., 1985, "Selecting the appropriate application development methodology", *SIGMIS Database*, v. 17, n. 1, p. 19-23.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S., 2006, *CMMI: Guidelines for Process Integration and Product Improvement*. 2 ed. Boston, MA, USA, Addison-Wesley.

CLARKE, J.; DOLADO, J. J.; HARMAN, M. *et al.*, 2003, "Reformulating software engineering as a search problem", *IEEE Proceedings-Software*, v. 150, n. 3, p. 161–175.

COCKBURN, A., 2000, "Selecting a Project's Methodology", *IEEE Softw.*, v. 17, n. 4, p. 64-71.

COCKBURN, A., 2001, *Agile Software Development*. Boston, MA, USA, Addison-Wesley.

COSTA, A. J. S., 2010, *Um Mecanismo de Adaptação de Processos de Software*. Dissertação de Mestrado, Universidade Federal do Pará (UFPA), Belém, PA, Brasil.

CUGOLA, G.; GHEZZI, C., 1998, "Software processes: A retrospective and a path to the future", *Software Process Improvement and Practice (SPIP) Journal*, v. 4, n. 3, p. 101-123.

DEELSTRA, S.; SINNEMA, M.; BOSCH, J., 2005, "Product derivation in software product families: a case study", *Journal of Systems and Software (JSS)*, v. 74, n. 2, p. 173-194.

EBERT, C., 2007, "Open Source Drives Innovation", *IEEE Software*, v. 24, n. 3, p. 105-109.

FEI DAI; TONG LI, 2007, "Tailoring Software Evolution Process". *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, p. 782-787, Nagoya, Japan.

FEILER, P.; HUMPHREY, W. S., 1992, *Software Process Development and Enactment: Concepts and Definitions*, Technical Report CMU/SEI-92-TR-004, SEI-CMU.

FERNANDES, P.; WERNER, C.; MURTA, L. G. P., 2008, "Feature modeling for context-aware software product lines". In: *International Conference on Software Engineering and Knowledge Engineering (SEKE)*, p. 758–763

FITZGERALD, B.; RUSSO, N. L.; O'KANE, T., 2003, "Software development method tailoring at Motorola", *Communications of ACM*, v. 46, n. 4, p. 64-70.

FSF, 2008. The Free Software Definition. Disponivel em: http://www.gnu.org/philosophy/free-sw.html. Acesso em: 31 maio 2008.

FUGGETTA, A., 2000, "Software process: a roadmap". In: *Proceedings of the Conference on The Future of Software Engineering*, p. 25-34, Limerick, Ireland.

GINSBERG, M.; QUINN, L., 1995, *Process Tailoring and the Software Capability Maturity Model* CMU/SEI-94-TR-024, SEI-CMU. Disponivel em: http://www.sei.cmu.edu/publications/documents/94.reports/94.tr.024.html.

GLASS, R. L., 2001, "Agile Versus Traditional: Make Love, Not War!", *Cutter IT Journal*, v. 14, n. 12, p. 12-18.

GLAZER, H.; DALTON, J.; ANDERSON, D. *et al.*, 2008, *CMMI or Agile: Why Not Embrace Both!*, SEI-CMU. Disponivel em: http://www.sei.cmu.edu/publications/documents/08.reports/08tn003.html.

HANSSON, C.; DITTRICH, Y.; GUSTAFSSON, B. *et al.*, 2006, "How agile are industrial software development practices?", *Journal of Systems and Software (JSS)*, v. 79, n. 9, p. 1295-1311.

HARMAN, M.; JONES, B. F., 2001, "Search-Based Software Engineering", *Information and Software Technology*, v. 43, p. 833-839.

HENDERSON-SELLERS, B., 2002, "Agile or Rigorous OO Methodologies: Getting the Best of Both Worlds", *Cutter IT Journal*, v. 15, n. 1, p. 25-33.

HERBSLEB, J. D.; PAULISH, D. J.; BASS, M., 2005, "Global software development at siemens: experience from nine projects". In: *Proceedings of the 27th international conference on Software engineering*, p. 524-533, St. Louis, MO, USA.

HOLLENBACH, C.; FRAKES, W., 1996, "Software Process Reuse in an Industrial Setting". In: *International Conference on Software Reuse (ICSR)*, p. 22-30

HUMPHREY, W. S., 1989, *Managing the Software Process*. Boston, MA, USA, Addison-Wesley.

IBM, 2009. Rational Unified Process (RUP). Disponivel em: http://www-01.ibm.com/software/awdtools/rup/. Acesso em: 12 abr 2010.

ISO/IEC, 1995, *ISO/IEC 12207: Information technology - Software life cycle processes*. Geneva, ISO/IEC.

JAUFMAN, O.; MUNCH, J., 2005, "Acquisition of a Project-Specific Process", *Product Focused Software Process Improvement*, Berlin / Heidelberg: Springer Verlag, p. 328-342.

KANG, K.; COHEN, S.; HESS, J. *et al.*, 1990, *Feature-Oriented Domain Analysis* CMU/SEI-90-TR-21, CMU-SEI. Disponivel em: http://www.sei.cmu.edu/domain-engineering/FODA.html.

KÜMMEL, G. Z., 2007, *Uma Abordagem para a Criação de Arquiteturas de Referência de Domínio a partir da Comparação de Modelos Arquiteturais de Aplicações*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro.

LAANTI, M.; KETTUNEN, P., 2005, "How to Steer an Embedded Software Project: Tactics for Selecting Agile Software Process Models", *Information and Software Technology*, v. 47, n. 9, p. 587-608.

LEITE, A. M. DOS S., 2011, *Modelo de Contexto para Adaptação de Processos de Software*. Dissertação de Mestrado, Programa de Pós-Graduação em Informática (PPGI) – UNIRIO, Rio de Janeiro, RJ, Brasil.

LINDVALL, M.; RUS, I., 2000, "Process diversity in software development", *Software, IEEE*, v. 17, n. 4 (ago.), p. 14-18.

LITTLE, T., 2005, "Context-adaptive agility: managing complexity and uncertainty", *IEEE Software*, v. 22, n. 3, p. 28-35.

MACHADO, L. F. D. C., 2000, *Modelo para Definição de Processos de Software na Estação TABA*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MAGDALENO, A. M., 2010a, "An optimization-based approach to software development process tailoring". In: *PhD Track - International Symposium on Search Based Software Engineering (SSBSE)*, p. 40-43, Benevento, Italy.

MAGDALENO, A. M., 2010b, "Balancing Collaboration and Discipline in Software Development Processes". In: *Doctoral Symposium of International Conference on Software Engineering (ICSE)*, p. 331-332, Cape Town, South Africa.

MAGDALENO, A. M., 2010c, *Apoio à Decisão para o Balanceamento de Colaboração e Disciplina nos Processos deDesenvolvimento de Software*. Exame de Qualificação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MAGDALENO, A. M.; BARROS, M. DE O.; WERNER, C. M. L. *et al.*, 2010, "Formulando a Adaptação de Processos de Desenvolvimento de Software como um Problema de Otimização". *I Workshop Brasileiro de Otimização em Engenharia de Software (WOES)*, p. 56-64, Salvador, BA, Brasil.

MAGDALENO, A. M.; WERNER, C. M. L.; ARAUJO, R. M., 2011, "Reconciling Software Development Models: A Quasi-Systematic Review", *Journal of Systems and Software (JSS)*, p. (to appear).

MARTÍNEZ-RUIZ, T.; GARCÍA, F.; PIATTINI, M., 2008, "Towards a SPEM v2. 0 Extension to Define Process Lines Variability Mechanisms", *Software Engineering Research, Management and Applications*, p. 115-130.

MNKANDLA, E., 2008, *A framework for agile methodology practices: a family of methodologies approach*. Doctoral thesis, Faculty of Engineering and the Built Environment, University of The Witwatersrand, Johannesburg. Disponivel em: http://hdl.handle.net/10539/5666.

MONTERO, I.; PENA, J.; RUIZ-CORTÉS, A., 2007, "Business Family Engineering: Does it make sense?". In: *I JISBD Taller sobre Procesos de Negocio e Ingeniería del Software (PNIS)*, p. 34-40, Zaragoza, España.

NORTHROP, L. M., 2002, "SEI's software product line tenets", *IEEE Software*, v. 19, n. 4, p. 32-40.

NUNES, V. T.; WERNER, C.; SANTORO, F. M., 2010, "Context-Based Process Line". In: *International Conference on Enterprise Information Systems (ICEIS)*, p. 277-282, Funchal, Madeira, Portugal.

ODYSSEY, 2011. Odyssey SDE Homepage. Disponivel em: http://reuse.cos.ufrj.br. Acesso em: 26 ago 2010.

OLIVEIRA, R. F. DE, 2006, *Formalização e Verificação de Consistência na Representação de Variabilidades*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

OSTERWEIL, L., 1987, "Software processes are software too". In: *International Conference on Software Engineering (ICSE)*, p. 2-13, Monterey, CA, USA.

PAPADIMITRIOU, C. H.; STEIGLITZ, K., 1998, *Combinatorial Optimization: Algorithms and Complexity*. Unabridged ed. New York, USA, Dover Publications.

PARK, S.; NA, H.; PARK, S. *et al.*, 2006, "A semi-automated filtering technique for software process tailoring using neural network", *Expert Systems with Applications*, v. 30, n. 2 (fev.), p. 179-189.

PATEL, C.; LYCETT, M.; MACREDIE, R. *et al.*, 2006, "Perceptions of Agility and Collaboration in Software Development Practice". In: *Hawaii International Conference on System Sciences (HICSS)*, p. 1-7, Kauai, Hawaii, USA.

PAULK, M. C., 2001, "Extreme programming from a CMM perspective", *Software, IEEE*, v. 18, n. 6, p. 19-26.

PAULK, M. C., 2009, *A History of the Capability Maturity Model for Software*, Technical Report, American Society for Quality (ASQ). Disponivel em: www.asq.org.

PEDREIRA, O.; PIATTINI, M.; LUACES, M. R. *et al.*, 2007, "A systematic review of software process tailoring", *SIGSOFT Software Engineering Notes*, v. 32, n. 3, p. 1-6.

PRESSMAN, R. S., 2001, *Software Engineering: A Practitioner's Approach*. 5 ed. McGraw-Hill.

QUMER, A.; HENDERSON-SELLERS, B., 2008, "An evaluation of the degree of agility in six agile methods and its applicability for method engineering", *Information and Software Technology*, v. 50, n. 4 (mar.), p. 280-295.

RAMAN, S., 2000, "It is software process, stupid: next millennium software quality key", *Aerospace and Electronic Systems Magazine, IEEE*, v. 15, n. 6, p. 33-37.

RAYMOND, E. S., 2001, *The Cathedral & the Bazaar*. Revised & Expanded ed. O'Reilly Media.

ROCHA, A. R. C. DA; MALDONADO, J. C.; WEBER, K. C., 2001, *Qualidade de Software: Teoria e Prática*. São Paulo, SP, Brasil, Prentice Hall.

ROMBACH, D., 2006, "Integrated Software Process and Product Lines", *Unifying the Software Process Spectrum*, Berlin/Heidelberg: Springer-Verlag, p. 83-90.

ROSEMANN, M.; RECKER, J., 2006, "Context-aware Process Design: Exploring the Extrinsic Drivers for Process Flexibility". In: *Workshop on Business Process Modeling, Development, and Support (BPMDS)*, p. 149-158, Luxembourg.

SAIDANI, O.; NURCAN, S., 2007, "Towards context aware business process modelling". In: *8th Workshop on Business Process Modeling, Development, and Support (BPMDS)*, p. 265-273, Trondheim, Norway.

SANTOS, V. A., 2009, *Aprendizado Organizacional e Melhoria Contínua de Processos de Software através de Reuso de Processos de Software*. Dissertação de Mestrado, Universidade Estadual do Ceará (UEC), Fortaleza, CE, Brasil.

SCHNIEDERS, A.; PUHLMANN, F., 2006, "Variability mechanisms in e-business process families". *International Conference on Business Information Systems (BIS)*, p. 583-601, Klagenfurt, Austria.

SCHWABER, K., 2004, *Agile Project Management with Scrum*. Washington, DC, USA, Microsoft Press.

SIEBEL, N. T.; COOK, S.; SATPATHY, M. *et al.*, 2003, "Latitudinal and longitudinal process diversity", *Journal of Software Maintenance: Research and Practice*, v. 15, n. 1 (jan.), p. 9–25.

SLOOTEN, K. VAN; BRINKKEMPER, S., 1993, "A Method Engineering Approach to Information Systems Development". In: *IFIP WG8.1 Working Conference on Information System Development Process*, p. 167-186, Como, Italy.

SOFTEX, 2011, *Melhoria de Processo do Software Brasileiro – Guia Geral*, Modelo de Qualidade Disponivel em: http://www.softex.br.

SOMMERVILLE, I., 2004, *Software Engineering*. 7 ed. Addison Wesley.

SWENSON, K. D.; PALMER, N.; KEMSLEY, S. *et al.*, 2011, *Social BPM*. Future Strategies Inc.

TAURION, C., 2004, *Software Livre: Potencialidades e Modelos de Negócio*. 2 ed. Rio de Janeiro, RJ, Brasil, Brasport.

TEIXEIRA, E. N., 2011, *OdysseyProcess-FEX: Uma Abordagem para Modelagem de Variabilidades de Linha de Processos de Software*. M.Sc. Dissertation, COPPE/UFRJ (In Portuguese), Rio de Janeiro, RJ, Brasil.

TERNITE, T., 2009, "Process Lines: A Product Line Approach Designed for Process Model Development". *35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, p. 173-180

THEUNISSEN, M.; KOURIE, D.; BOAKE, A., 2008, "Corporate-, Agile- and Open Source Software Development: A Witch's Brew or An Elixir of Life?", *Balancing Agility and Formalism in Software Engineering: Second IFIP TC 2 Central and East European Conference on Software Engineering Techniques (CEE-SET)*, Springer-Verlag, p. 84-95.

TURK, D.; FRANCE, R.; RUMPE, B., 2002, "Limitations of agile software processes". In: *International Conference on Extreme Programming and Flexible Processes in Software Engineering*, p. 43-46, Alghero, Italy.

TURNER, R.; JAIN, A., 2002, "Agile Meets CMMI: Culture Clash or Common Cause?", *Extreme Programming and Agile Methods - XP/Agile Universe 2002*, 1 ed, chapter 2418, Heidelberg: Springer-Verlag, p. 153-165.

WARSTA, J.; ABRAHAMSSON, P., 2003, "Is Open Source Software Development Essentially an Agile Method?". In: *Proceedings of the Workshop on Open Source Software Development*, p. 143-147, Portland, OR, USA.

WASHIZAKI, H., 2006, "Building Software Process Line Architectures from Bottom Up", *Product-Focused Software Process Improvement (PROFES)*, , chapter 4034, Amsterdam, The Netherlands: LNCS, p. 415-421.

WEERD, G. C. VAN DE, 2009, *Advancing in software product management: An incremental method engineering approach*. Tese de Doutorado, Utrecht University, Netherlands. Disponivel em: http://igitur-archive.library.uu.nl/dissertations/2009-0826-200144/UUindex.html.

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