UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO

GUILHERME VINÍCIUS MENDES VIEIRA
WALLACE PRÉVOT DA SILVA

PROCESS BROKER: UMA INFRA-ESTRUTURA DE MEDIAÇÃO DE COMPONENTES DE PROCESSO.

NITERÓI
2010

GUILHERME VINÍCIUS MENDES VIEIRA
WALLACE PRÉVOT DA SILVA

PROCESS BROKER: UMA INFRA-ESTRUTURA DE MEDIAÇÃO DE COMPONENTES DE PROCESSO.

Dissertação apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Área de Concentração: Engenharia de Software.

Orientador: LEONARDO GRESTA PAULINO MURTA
Co-orientador: AHILTON SILVA BARRETO

NITERÓI
2010

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

V658 Vieira, Guilherme Vinícius Mendes.
Process broker : uma infra-estrutura de mediação de componentes de processo / Guilherme Vinícius Mendes Vieira, Wallace Prévot da Silva. – Niterói, RJ : [s.n.], 2010.
91 f.

Trabalho (Conclusão de Curso) – Departamento de Computação, Universidade Federal Fluminense, 2010.
Orientadores: Leonardo Gresta Paulino Murta, Ahilton Silva Barreto.

1. Desenvolvimento de software. 2. Engenharia de software. 3. XML – Linguagem de marcação estendida. I. Silva, Wallace Prévot da. II. Título.

CDD 005.12

GUILHERME VINÍCIUS MENDES VIEIRA
WALLACE PRÉVOT DA SILVA

PROCESS BROKER: UMA INFRA-ESTRUTURA DE MEDIAÇÃO DE COMPONENTES DE PROCESSO.

Dissertação apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Área de Concentração: Engenharia de Software.

Aprovado em dezembro de 2010.

BANCA EXAMINADORA

_____

Prof. D.Sc. LEONARDO GRESTA PAULINO MURTA – Orientador
UFF

_____

M.Sc. AHILTON SILVA BARRETO – Co-orientador
COPPE/UFRJ

_____

Profa. D.Sc. TERESA CRISTINA DE AGUIAR
UFF

_____

Profa. D.Sc. VANESSA BRAGANHOLO MURTA
UFF

NITERÓI
2010

Aos nossos pais, irmãos, familiares e amigos.

AGRADECIMENTOS

A Deus, pela oportunidade de mais uma conquista.

Aos nossos pais, pelo incentivo, carinho, dedicação e apoio.

Aos nossos irmãos, por estarem sempre presentes, nos encorajando e acreditando em nosso potencial.

Aos amigos, pela companhia, convívio e incentivo, fazendo dessa jornada uma aventura agradável.

Ao professor Leonardo Gresta Paulino Murta e Ahilton Silva Barreto, pelo apoio, dedicação, confiança e orientação.

Às professoras Teresa Cristina de Aguiar e Vanessa Braganholo Murta por fazerem parte da banca examinadora e pelas contribuições para este trabalho.

A todos os demais professores, pelos ensinamentos e contribuições para o nosso aprendizado.

A todos que, de alguma forma, contribuíram para a realização desse trabalho e para a conclusão dessa etapa em nossas vidas.

“Há homens que lutam um dia e são bons.
Há outros que lutam um ano e são melhores. (...)
Porém, há os que lutam toda a vida.
Esses são os imprescindíveis.”

Bertolt Brecht

RESUMO

Processos de software são indispensáveis para a construção de softwares de qualidade. Acredita-se que a qualidade do software desenvolvido está intrinsecamente relacionada ao processo de software adotado. No entanto, a definição de um processo de software pode se tornar uma Atividade complexa e custosa para uma organização e exige um profissional bastante experiente e especializado. Diante dessa dificuldade, um processo de software poderia ser reaproveitado em outras organizações, uma vez que esse conhecimento esteja explicitado, formalizado e disponibilizado. Uma das formas de explicitar esse conhecimento é através da definição e disponibilização de Componentes de Processo de Software e de outras estruturas reutilizáveis de processo de software. Técnicas para possibilitar o reuso do conhecimento relacionado a Processos de Software e a disponibilização de uma orientação adequada para a escolha e aplicação dessas estruturas vem sendo desenvolvidas. Sendo assim, Processos de Software podem ser definidos através de Componentes de Processo de Software reutilizáveis. Neste contexto, o objetivo deste trabalho é o desenvolvimento de uma infra-estrutura que permita que instituições implementadoras de Processos de Software, organizações desenvolvedoras de software ou projetos específicos nessas organizações possam negociar Componentes de Processo de Software.

PALAVRAS-CHAVE: processo, software, reutilização, componentes, xml.

ABSTRACT

Software processes are necessary for developing quality softwares. It is believed that the quality of software developed is intrinsically related to software process adopted. However, the definition of a software process can become a complex and costly to an organization and requires a very experienced and specialized professional. Given this difficulty, a software process could be reused in other organizations, since this knowledge is explicit, formalized and available. One way to explicit this knowledge is by definition and availability of software process components and other reusable software process structures. Techniques to enable the reuse of knowledge related to software processes and the provision of adequate guidance for choosing and implementing these structures has been developed.Thus, software processes can be defined by reusable software process components. In this context, the objective of this work is to develop an infrastructure that allows software process implementing institutions, software development organizations or projects in these organizations can negotiate software process components.

KEYWORDS: process, software, reusability, components, xml.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

1.1 MOTIVAÇÃO

Atualmente, verifica-se que as organizações desenvolvedoras de software têm se preocupado com a qualidade de seus produtos, devido à grande concorrência, em um mercado consumidor cada vez mais exigente e dependente de um aparato tecnológico (CARD, 1995 *apud* FLORAC *et al.*, 1997).

Entende-se que a qualidade de um software esteja intrinsecamente ligada ao processo de software. Um processo de software fornece um guia para o desenvolvimento de um projeto. Nele estão contidas as Atividades, a estrutura a ser adotada, a organização, os artefatos requeridos e produzidos, os procedimentos adotados, os recursos utilizados, entre outros (FUGGETA, 2000).

Acredita-se ainda que uma organização bem gerenciada, com processos bem definidos, tenha maior probabilidade de desenvolver produtos que sigam às exigências do cliente dentro do cronograma e do orçamento, quando comparada a uma organização mal gerenciada e sem processos definidos (SOLINGEN, 2004).

Todavia, definir um processo de software não é uma tarefa simples. É uma Atividade complexa e custosa, onde se exigem profissionais experientes e altamente capacitados. Devido a essa complexidade, as organizações têm enfrentado problemas em definir seus processos. A quantidade de conhecimento envolvida e, muita das vezes, o fato das organizações não possuírem um profissional capacitado para tal Atividade, faz com quem elas busquem consultoria externa para a definição dos seus processos (BARRETO *et al.*, 2009).

Como a definição de um processo é uma tarefa um tanto complexa, e o conhecimento necessário é restrito aos profissionais da área, observa-se a necessidade da criação de meios de apoio que permitam que esse conhecimento, provido pelos profissionais experientes, seja explicitado, disponibilizado e disseminado, garantindo o reuso desse conhecimento.

Esse apoio, para ser efetivo, deve considerar muitos dos aspectos normalmente considerados por um engenheiro de processos no momento de definir processos, tais como: (i) rastreabilidade entre partes do processo e normas, modelos de maturidade, objetivos organizacionais e requisitos do processo; (ii) outros processos anteriormente definidos e sua adequação ou não a situações específicas; (iii) consistência entre as diversas partes do processo, garantindo que o que é requerido por uma parte do processo seja produzido por outra, e que todos os pré-requisitos sejam satisfeitos; (iv) conformidade do processo definido a padrões, normas e modelos de maturidade; e (v) adequação do processo à organização que o utilizará (BARRETO, 2007).

Frequentemente, instituições implementadoras de processo, aquelas que fornecem consultoria, são acionadas para definir processos em diferentes organizações. Após uma instituição implementadora definir vários processos para diferentes organizações, o que provavelmente resultaria em um processo único para cada organização, poderia ser percebido que há pontos de semelhanças e variações entre esses processos. Isso indica que o conceito de reutilização pode ser aplicado a Processos de Software. Uma abordagem de definição de processos, através da reutilização desses pontos de semelhança, possivelmente tornaria a Atividade menos complexa. Para isso é necessário que a representação de semelhanças e variabilidades seja realizada de maneira formal, visando explicitar o que é comum e o que varia a cada definição de processos (BARRETO, 2007).

Modelos de maturidade como o MPS.BR (SOFTEX, 2007), por exemplo, institui que organizações com maior nível de maturidade definam seus processos a partir de sub-processos ou elementos de processos, isto é, unidades menores de processo. Esse cenário caracteriza a definição de processos a partir de componentes reutilizáveis, ou seja, é factível uma componentização de processos. Portanto uma forma de explicitar e disponibilizar o conhecimento necessário para definição de processos é através de Componentes de Processo e de outras estruturas reutilizáveis, além de guias para auxiliar na escolha desses itens em cada situação (BARRETO, 2007).

Contudo, para que Componentes de Processo possam transitar entre instituições implementadoras e organizações desenvolvedoras de software, é necessária uma infra-

estrutura de apoio. Essa infra-estrutura de apoio deve ser capaz de tratar questões relacionadas à importação e exportação de processos nesses ambientes, assim como viabilizar o transporte dos Componentes de Processo, permitindo que as bibliotecas de Componentes de Processo das organizações sejam alimentadas por componentes provenientes das instituições implementadoras. Por outro lado, as instituições implementadoras poderiam coletar métricas de desempenho dos seus Componentes de Processo em diversos projetos de diferentes organizações, possibilitando fazer análises de estabilidade e capacidade dos componentes de maneira mais abrangente para *benchmarking*, o que as ajudaria na manutenção e melhoria de um determinado componente.

1.2 OBJETIVO

O objetivo deste trabalho é construir uma infra-estrutura mediadora de Componentes de Processos de Software, denominada de Process Broker, que possibilite o intercâmbio de Componentes de Processos de Software entre as instituições implementadoras e as organizações, enfatizando o conceito de reutilização de componentes. Essa abordagem tem por objetivo prover um agente facilitador à definição de Processos de Software em diferentes níveis através da disponibilização e disseminação de Componentes de Processo de Software reutilizáveis entre as organizações e instituições implementadores.

O Process Broker surgiu devido à demanda de outra aplicação, um ambiente de definição de Processos de Software baseado em Componentes de Processo. Essa aplicação é parte da tese de doutorado de BARRETO (2007). Portanto, a infra-estrutura do Process Broker é acoplada a essa aplicação. A função do Process Broker é disponibilizar os Componentes de Processo definidos nesse ambiente, de uma forma que possibilite disseminá-los em diversas instâncias desse ambiente.

Devido a essa dependência, a revisão bibliográfica deste trabalho está fortemente baseada em BARRETO (2007), incluindo algumas citações a outros autores. Por uma questão de organização, foi preferido explicitar essa decisão do que a todo o momento fazer o uso de *apud*.

Um possível cenário para o uso do Process Broker seria na interação entre as instituições implementadores de processo de software e organizações desenvolvedoras de

software. As instituições implementadoras seriam encarregadas de desenvolver os Componentes de Processo de Software através do ambiente de definição e disponibilizá-los, fazendo então o uso do Process Broker, em uma biblioteca de componentes acessível tanto pelas próprias instituições implementadoras como pelas organizações. Outra opção de disponibilização seria via um canal estreito e direto de comunicação entre a Instituição Implementadora e a Organização, como um simples e-mail. A Figura 1, ilustra o possível cenário onde o Process Broker é inserido.

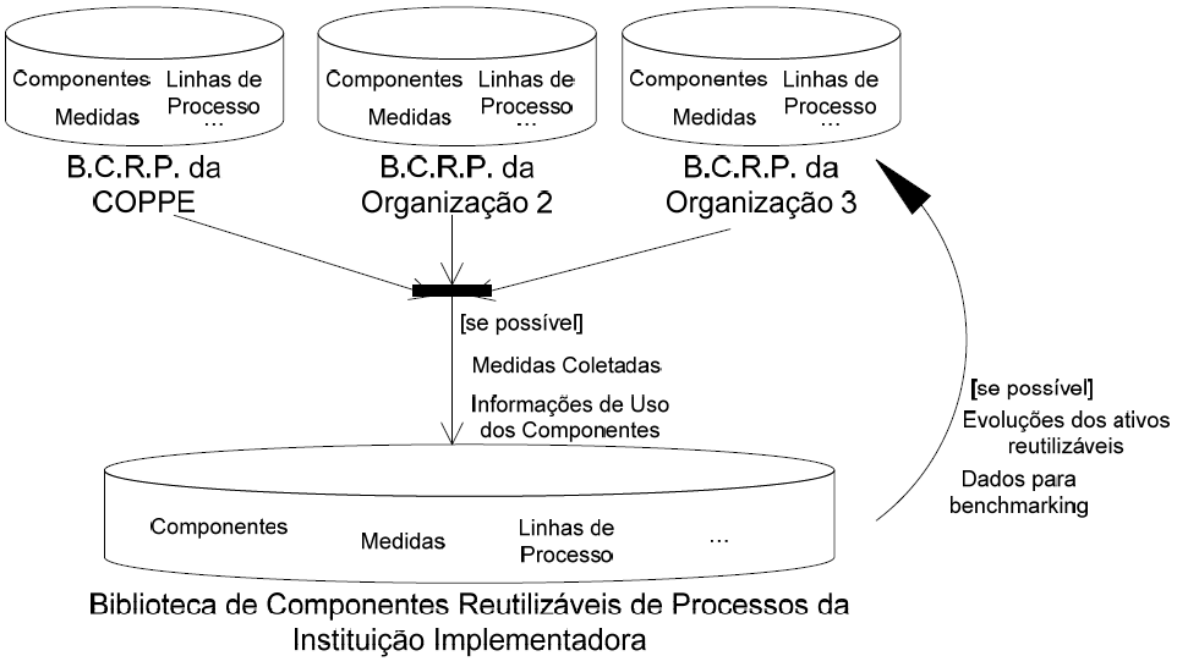

Figura 1: Interação entre as bases das organizações e da instituição implementadora (BARRETO, 2007)

Concluindo, essa infra-estrutura deve permitir que um componente criado em um determinado ambiente não se limite apenas àquela instância, mas que esse componente possa ser difundido e utilizado em outros locais que utilizem o mesmo ambiente. Portanto, esse trabalho engloba a definição de uma representação de Componentes de Processo de Software que permita o efetivo intercâmbio de informações e a construção dessa infra-estrutura mediadora.

1.3 ORGANIZAÇÃO DO TRABALHO

Esse capítulo introdutório tem como objetivo contextualizar o trabalho, destacando os pontos principais que motivam o seu desenvolvimento. Esses enfoques serão detalhados ao longo dos próximos capítulos. O capítulo 2, Processos de Software, apresenta uma revisão bibliográfica sobre Processos de Software de maneira geral, aprofundando-se no conceito de definição de processo de software. O capítulo 3, Reutilização de Processos de Software através de Componentes de Processo de Software, contém uma revisão bibliográfica sobre o reuso de Componentes de Processo, que é uma das motivações do Process Broker, isto é, facilitar o reuso de Componentes de Processo entre diversas organizações. O capítulo 4, Process Broker, apresenta a abordagem proposta por esse trabalho de conclusão de curso, assim como sua implementação e exemplo de sua utilização. A Conclusão deste trabalho como um todo de maneira sucinta e objetiva está no capítulo 5. Por fim as Referências Bibliográficas consultadas são listadas e os Anexos que são referenciados ao longo do texto são apresentados.

CAPÍTULO 2 - PROCESSOS DE SOFTWARE

2.1 INTRODUÇÃO

Quando se elabora um produto, normalmente é necessário realizar uma série de passos, como um roteiro que auxilie na criação de um produto de qualidade. Esse roteiro é chamado de processo. Um processo de software é a aplicação do conceito de processo no contexto de desenvolvimento de software. FUGGETA (2000) define processo de software como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos, e artefatos que são necessários para conceber, desenvolver, implantar e manter um produto de software. A norma internacional ISO/IEC 15504 (2003) estabelece que um processo é um conjunto de Atividades inter-relacionadas que transforma entradas em saídas.

Atualmente, é indispensável a utilização de processos no desenvolvimento de software, pois é fato que desenvolver software vai muito além de programação e ferramentas. Essa complexa tarefa compreende também os recursos humanos envolvidos e principalmente os procedimentos criados e utilizados para alcançar o produto final: um software de qualidade. Sendo assim, um processo de software bem definido permite que profissionais de engenharia de software possam trabalhar de forma ordenada, possibilitando um melhor entendimento do seu trabalho, bem como de outras Atividades executadas por outros membros da mesma equipe (HUMPHREY, 1989).

Um processo bem definido melhora a maneira como o trabalho é conduzido, estabelecendo uma organização estruturada das Atividades. Ele fornece um planejamento a ser seguido caso ocorram problemas e elabora maneiras de preveni-los ou resolve-los. O processo de software define como o desenvolvimento é organizado, gerenciado, medido, apoiado e melhorado (ACUÑA et al., 2000).

Organizações que controlam seus processos são capazes de prever as características dos seus produtos e serviços, seus custos e cronogramas, e melhorar sua eficácia, eficiência, e rentabilidade de suas operações técnicas e de negócio (FLORAC e CARLETON, 1999).

É imprescindível que haja uma forma de gerenciar processos, de modo que a partir de um roteiro definido, seja possível alcançar os objetivos esperados. A gerência de processos deve verificar se os produtos e serviços provenientes do processo estão condizentes com os requisitos estabelecidos pelo cliente e com os próprios objetivos de negócios da organização, se os processos definidos estão sendo seguidos e realizar alterações nos processos quando necessário. FLORAC e CARLETON (1999) atribuem à gerência de processo de software quatro responsabilidades:

- Definir o processo: Projetar processos que estejam alinhados aos objetivos de negócios e técnicos da organização, assegurando que a organização tenha a habilidade de executar e manter os processos.
- Medir o processo: Coletar métricas dos processos, analisando o desempenho de cada processo. As métricas devem ser guardadas para avaliar a estabilidade e capacidade do processo, interpretar os resultados, prever custos futuros e desempenho e identificar possíveis oportunidades para melhoria no processo. A questão sobre estabilidade e capacidade do processo é abordada na seção 2.3.
- Controlar o processo: Assegurar que o processo comporta-se de maneira consistente, identificando variações de desempenho causadas eventualmente de modo que seja realizada uma correção nessas falhas.
- Melhorar o processo: Realizar melhorias em um processo através de alterações que venham a melhorar a capacidade do processo ou simplesmente substituindo sub-processos existentes por outros que sejam mais eficazes e eficientes.

Este capítulo tem como objetivo introduzir o conceito de Processos de Software, focando na parte de definição de processos, pois este é o ponto de partida do enfoque teórico para chegar à aplicação do Process Broker. Uma vez que o processo é definido, ele pode ser difundido entre as organizações. São citadas também as normas e padrões internacionais,

fazendo uma relação como cada uma determina como deve ser realizada a definição de processos. Para finalizar, é discutido um fator relevante em processos, que é a questão da estabilidade e capacidade de Processos de Software.

2.2 DEFINIÇÃO DE PROCESSOS DE SOFTWARE

No desenvolvimento de software, existem várias maneiras distintas de executar as diversas tarefas que compõe o roteiro de elaboração do software. Através da definição de processos, todas essas tarefas são explicitamente formalizadas, de modo que guiem os profissionais de software ao longo do desenvolvimento de maneira organizada. A partir de um processo definido, fica perceptível o que se espera que deva ser feito, qual a abordagem adotada para a realização da tarefa, em que momento a tarefa deve ser executada e quais são as pessoas envolvidas em cada uma das Atividades pertencentes ao processo. Do ponto de vista prático, definir processo é uma tarefa altamente complexa, pois a Engenharia de Software não é uma Atividade que pode ser encarada como um procedimento repetitivo. A Engenharia de Software trabalha com um processo intelectual, de modo que seja ajustado conforme as necessidades da organização. Um balanceamento é, portanto, necessário entre as necessidades individuais de flexibilidade e a necessidade organizacional de padronização e consistência (HUMPHREY, 1989). De acordo com HUMPHREY (1989), alguns dos fatores a serem considerados são:

- Uma vez que projetos de software possuem diferenças, seus Processos de Softwares também as possuem.
- Na falta de um processo de software universal, organizações e projetos devem definir processos que atendam suas próprias necessidades específicas.
- O processo utilizado para um dado projeto deve considerar o nível de experiência dos membros da equipe, a situação corrente dos produtos e as ferramentas e infra-estrutura disponível.

Um processo de software bem definido deve conter todas as informações relevantes para sua execução, como as Atividades a serem desempenhadas, os métodos adotados, as ferramentas de apoio, os diferentes tipos de perfil de recursos humanos envolvidos, os insumos produzidos e utilizados etc.

De acordo com MADHAVJI (1991), as vantagens de se ter processos definidos basicamente são: a capacidade de medi-los, reutiliza-los, gerencia-los, promover um melhor entendimento entre as equipes e possibilitar a criação de uma base para próximos níveis de melhoria de processos.

A definição de Processos de Software não é uma Atividade trivial, pois envolve experiência e muitos conceitos da Engenharia de Software. É necessário um profissional altamente qualificado e experiente para definir processos de maneira adequada, pois além de todo o conhecimento técnico, é imprescindível considerar outros fatores como as necessidades e características do projeto ou da organização, a experiência das pessoas envolvidas no desenvolvimento do software, as regras de negócios e políticas que a organização em questão adere, a infra-estrutura e recursos disponíveis, as conformidades com os modelos de referência, etc.

Segundo WANG e KING (2000), as principais abordagens para a definição de processos são:

- Definição de processos com base em modelos de referência – abordagem *Top-down*: (1) Selecionar, a partir de modelos de processo de referência, elementos para a reutilização, estabelecendo um processo padrão para a organização; (2) Derivar, a partir do processo padrão organizacional, processos instanciados para os projetos da organização, que devem ser adequados à complexidade, tamanho e demais características do projeto; e (3) Aplicar o processo instanciado nos projetos;
- Definição de processos com base na cultura da organização – abordagem *Bottom-up:* Neste caso, aproveita-se o máximo os processos já existentes na organização, que serão sucessivamente melhorados em função dos resultados de avaliações e objetivos de melhoria;
- Definição de processos com base nos objetivos organizacionais: Em função dos objetivos estabelecidos para a organização, são definidos quais são os processos que atendem estes objetivos, sendo então priorizados, definidos e implantados conforme planos e recursos.
- Definição de processos com base nas necessidades dos clientes: A definição dos processos é orientada àqueles que têm maior potencial de gerar satisfação aos clientes da organização.

Apesar das quatro abordagens definidas acima, também é possível encontrar abordagens híbridas, que apresentam uma interseção de características.

Além da necessidade de se ter processos especializados, ou seja, específicos para projetos, deve ser considerada também a necessidade de se ter processos padronizados. Com processos padronizados, a experiência em cada projeto pode oferecer uma possibilidade de melhoria no processo geral, além de fornecer uma base para medições do processo e da qualidade. Outro ponto que viabiliza a adoção de processos padrão é a questão crítica do tempo. Definir processos é uma Atividade complexa e custosa e requer tempo e um esforço consideravelmente qualificado. Em outras palavras, há a necessidade do conhecimento técnico para ser realizada. Portanto, é significativamente custoso que a cada novo projeto, seja necessário uma demanda para uma nova definição de processo (HUMPHREY, 1989).

Embora cada projeto tenha sua peculiaridade, é possível identificar conjuntos de ativos de processos comuns para toda a organização, ou seja, informações que são utilizadas para prover o sucesso de um projeto. Esses conjuntos constituem o processo padrão de uma organização. A partir do processo padrão, outros processos podem ser especializados, levando em consideração características como questões tecnológicas, paradigmas ou domínios de aplicação etc. Mais adiante, será abordada detalhadamente a questão do processo padrão.

2.2.1 Definição de Processos de Software em Níveis

A definição de Processos de Software pode ocorrer em diferentes níveis. Em uma abordagem *top-down*, por exemplo, ela pode ter seu ponto de partida na instituição implementadora, posteriormente na organização desenvolvedora de software e culminando em um projeto específico a ser desenvolvido na organização.

As instituições implementadoras são organizações tecnicamente capacitadas, cuja finalidade é fornecer consultoria na definição, implementação e melhoria de Processos de Software às organizações desenvolvedoras de software. Esse apoio às organizações tem se tornado cada vez mais essencial, uma vez que a competição por mercado é cada vez mais acirrada, o que justifica que seus processos tendam a ser constantemente melhorados.

As organizações desenvolvedoras de software são aquelas que desenvolvem soluções para diversas empresas de diferentes segmentos, através da construção de programas de computador. Atualmente há uma variedade de organizações desenvolvedoras de software presentes no mercado, possibilitando às diversas empresas o poder de escolha no momento de

adquirir um software. O fator que diferencia uma organização desenvolvedora de software das outras é a capacidade dela produzir softwares de qualidade. Existem teorias que acreditam que isso implique diretamente no processo de software que elas executam na elaboração desses softwares. Portanto, baseando-se nesse fundamento, encoraja-se que organizações desenvolvedoras de software definam seus processos padrão e os especializem adequadamente (FUGGETA, 2000).

Projetos surgem nas organizações desenvolvedoras de software a partir da demanda para a construção de software. Cada projeto tem por objetivo prover meios para a concretização do produto final, que no caso é o software, no prazo estipulado. Não se deve confundir projeto com processo. Projeto é o esforço exigido em um certo momento, ou seja, tem data de início e término definida para alcançar um objetivo específico. Processo, por sua vez, é um conjunto de tarefas relacionadas que são executadas repetidamente, de forma a atingir um objetivo. São nos projetos onde os processos de desenvolvimento de software são finalmente aplicados. Ter projetos de software com características distintas implica diretamente em Processos de Software com características distintas. Por exemplo, um software científico possui características diferentes de uma loja virtual *web*. Isso resulta em projetos com características diferentes, que por sua vez estão associados a Processos de Software distintos, caracterizando um efeito dominó.

2.2.2 Definição de Processos de Software em normas e modelos de referência

A definição de Processos de Software tem sido considerada um fator fundamental para que uma organização atinja níveis mais altos de maturidade. Devido a essa importância, atualmente existem vários padrões internacionais e nacionais que se preocupam com isso e descrevem modelos de referência de processo. Dentre as normas, podem ser citadas ISO/IEC 24774, ISO/IEC 12207 e ISO/IEC 15504. No que tange aos modelos de maturidade, destacam-se o CMMI-DEV, em nível internacional, e o MPS.BR, em nível nacional. Nesta seção estão descritas, resumidamente, essas normas e modelos de maturidade citados, no que diz respeito à área de definição de processos (BARRETO, 2007).

De maneira geral, a norma internacional ISO/IEC 15504 (2003) define processo padrão como sendo o conjunto de definições dos processos básicos que guiam todos os

processos de uma organização. Esse conjunto de definições compreende os elementos fundamentais e suas inter-relações, de modo que estejam presentes nos processos definidos que são implementados em projetos nas organizações. Um processo padrão estabelece Atividades consistentes que são utilizadas pela organização em diversos projetos, e, além disso, contribui para a estabilidade e melhorias de longo prazo, devido à análise de experiências em cada projeto. Já um processo definido, trata-se de um processo que é gerenciado e adaptado através do conjunto de processos padrão da organização, utilizando-se de guias organizacionais de adaptação. Nesses guias estão contidas orientações que auxiliam uma organização no momento de adaptar a descrição de processo de um processo padrão para atender às necessidades específicas de um projeto, pois descrevem o que pode e o que não pode ser alterado na instância de um processo padrão. Por sua vez, um processo especializado é uma instância do processo padrão, onde são levadas em consideração questões específicas como tecnologias, paradigmas, domínios de aplicação, etc. Com isso, para cada projeto, pode-se instanciar um processo definido baseado no processo padrão, sendo ele especializado ou não. Exemplificando, suponha que uma organização que desenvolva aplicações financeiras para plataformas desktop e web, tenha o seu conjunto de processos padrão, de modo a garantir que todos os projetos tenham processos condizentes com as políticas e a garantia da qualidade da organização. Para cada projeto que essa organização venha a desenvolver, ela deverá criar um processo definido a partir do processo padrão, fazendo o uso dos guias organizacionais de adaptação. Agora, suponha que surja a demanda para construir uma aplicação para dispositivos de telefones móveis. Isso requer que o processo padrão seja especializado, de forma que atenda à nova demanda tecnológica.

A ISO/IEC 15504 (2003) também se preocupa com a melhoria de processos e com a capacidade de processo. A capacidade de processo é definida através de uma escala de seis níveis: nível 0 (Incompleto), nível 1 (Executado), nível 2 (Gerenciado), nível 3 (Estabelecido), nível 4 (Previsível) e nível 5 (Em otimização). No que diz respeito à definição de processos, vale ressaltar o nível de capacidade 3. Nesse nível um processo é definido através da adaptação do processo padrão. Dados referentes à execução do processo são coletados de modo que os processos sejam analisados, identificando possíveis oportunidades de melhoria tanto no processo padrão quanto no definido.

A ISO/IEC 24774 (2006) estabelece a descrição de processos através de atributos e regras para a sua formulação, como título (escopo do processo), propósito (objetivo do processo), resultados esperados (resultado obtido após a execução bem sucedida do processo),

Atividades (ações para se alcançar os resultados esperados) e tarefas (ações para se obter uma Atividade). A norma orienta como os atributos de um processo devem ser descritos, fornecendo exemplos de utilização.

No contexto de definição de processos, a ISO/IEC 12207 (2004), institui o processo de estabelecimento de processos. Esse processo estabelece um conjunto de processos padrão de modo que seja aplicável às suas Atividades de negócio, através de uma adaptação.

O CMMI (*Capability Maturity Model Integration*) é um modelo de maturidade de melhoria de processos para o desenvolvimento de produtos e serviços. O objetivo do CMMI-DEV (para desenvolvimento) é ajudar organizações a melhorarem seus processos tanto de desenvolvimento como de manutenção em produtos e serviços. Existem dois tipos de representação no CMMI: contínua e em estágios. A representação contínua visa à melhoria a uma área de processo específica. Ela possui seis níveis de capacidade: nível 0 (Incompleto), nível 1 (Desempenhado), nível 2 (Gerenciado), nível 3 (Definido), nível 4 (Gerenciado Quantitativamente) e nível 5 (Otimizado). Já a representação em estágios visa à melhoria da unidade organizacional. Ela é representada em cinco níveis de maturidade: nível 1 (Inicial), nível 2 (Gerenciado), nível 3 (Definido), nível 4 (Gerenciado Quantitativamente) e nível 5 (Otimizado) (CHRISSIS et al., 2006).

No que tange à questão de definição de processos, os níveis que se preocupam com esse assunto são o nível 3 (Definido) e o nível 4 (Gerenciado Quantitativamente). Destacam-se quatro áreas de processos no nível 3: Definição do Processo Organizacional (*Organizational Process Definition* – OPD), Gerência Integrada do Projeto (*Integrated Project Management* – IPM) e Foco no Processo Organizacional (*Organization Process Focus* – OPF). Já no nível adiante, o nível 4, há ainda uma área de processo relevante à definição de processo: Gerência Quantitativa de Projeto (*Quantitative Project Management* – QPM) (CHRISSIS et al., 2006).

Intuitivamente pode-se imaginar que a área de processo Definição do Processo Organizacional seja a que esteja mais ligada à definição de processos. E de fato, está. Ela estabelece a definição dos processos padrão de uma organização e guias para sua adaptação para a definição de processos específicos (CHRISSIS et al., 2006).

A área de processo Gerência Integrada do Projeto estabelece e mantêm o projeto de acordo com um processo definido e integrado, que é proveniente de uma adaptação de um conjunto de processos padrão da organização. Essa área de processo também tem uma relação com a definição de processos, uma vez que um processo definido no início de um projeto dá-

se através da adaptação de um processo padrão da organização. A área engloba ainda o gerenciamento de projetos, onde eles compartilham ativos de processo, dados e lições aprendidas (CHRISSIS et al., 2006).

A área de processo Foco no Processo Organizacional preocupa-se com o planejamento, implementação e melhorias nos processos organizacionais através da identificação dos pontos fracos e fortes dos processos e ativos de processos da organização. Esta área de processo também apresenta uma relação com a definição de processos, pois envolve o estabelecimento das necessidades e objetivos dos processos da organização, e ainda há a possibilidade de melhorias nos processos e em seus ativos, e com isso propaga-se a melhoria para o conjunto de processos padrão da organização (CHRISSIS et al., 2006).

E por último, a área de processo Gerência Quantitativa de Projetos estabelece que a composição do processo definido do projeto deve ser realizada através da análise dos dados sobre a estabilidade e capacidade dos sub-processos que vão compor o processo (CHRISSIS et al., 2006).

O MPS.BR é um programa de melhoria de processos brasileiro. O MPS.BR tem seus alicerces nos conceitos de maturidade e capacidade do processo para a avaliação e melhoria de qualidade de produtos de software e serviços. Uma das metas do MPS.BR é definir e aprimorar um modelo de melhoria e avaliação de processo de software, visando preferencialmente às micro, pequenas e médias empresas, de forma a atender as suas necessidades de negócio e ser reconhecido nacionalmente e internacionalmente como um modelo aplicável à indústria de software. O MPS.BR estabelece um modelo de processos de software e um método de avaliação de processos. Esta estrutura fornece sustentação e garante que o MPS.BR esteja sendo empregado de forma coerente com as suas definições. O MPS estabelece também um modelo de negócio para apoiar a sua adoção pelas empresas brasileiras desenvolvedoras de software (SOFTEX, 2007).

O Modelo de Referência do MPS.BR, o MR-MPS, estabelece sete níveis de maturidade (SOFTEX, 2007): nível A (Em otimização), nível B (Gerenciado Quantitativamente), nível C (Definido), nível D (Largamente Definido), nível E (Parcialmente Definido), nível F (Gerenciado) e nível G (Parcialmente Gerenciado).

Os processos no MR-MPS descrevem os objetivos a serem alcançados e os resultados esperados. A capacidade do processo é representada por um conjunto de atributos de processo descrito em termos de resultados esperados. Atributos de processo são características mensuráveis da capacidade do processo aplicável a qualquer processo (ISO/IEC, 2004). Em

relação à definição de processos, destaca-se o atributo de processo AP 3.1, onde o processo é definido. O resultado esperado desse atributo de processo é a definição de um processo padrão, incluindo as diretrizes para a sua adaptação para o processo definido, a sequência e interação do processo padrão com outros processos (SOFTEX, 2007).

Existem também três processos que estão ligados com a definição de Processos de Software no MPS.BR: Definição do Processo Organizacional (DFP), Avaliação e Melhoria do Processo Organizacional (AMP) e Gerência de Projetos (GPR) (SOFTEX, 2007).

O processo de Definição do Processo Organizacional (DFP) tem como objetivo estabelecer e manter ativos de processo organizacional e padrões do ambiente de trabalho (SOFTEX, 2007).

O processo de Avaliação e Melhoria do Processo Organizacional (AMP) tem como objetivo avaliar o quanto os processos padrão da organização contribuem para atingir os objetivos de negócio da organização, auxiliando-a com o planejamento e na melhoria contínua dos processos através da análise dos ativos de processo organizacional (SOFTEX, 2007).

Finalmente, o processo de Gerência de Projetos (GPR) tem como objetivo estabelecer que o processo definido seja instituído através de uma estratégia para adaptação do processo da organização e que a seleção dos sub-processos para compor o processo definido seja efetuado com base em seu histórico de execução, levando em conta fatores como a estabilidade e capacidade (SOFTEX, 2007).

2.3 ESTABILIDADE E CAPACIDADE DE PROCESSOS DE SOFTWARE

A definição de processos em organizações de mais alta maturidade é realizada através da escolha de sub-processos que irão compor o processo. Além de compor o processo com base em sub-processos menores, a escolha dos sub-processos deve ser realizada baseando-se nos dados sobre estabilidade e capacidade de sub-processos. Dessa forma, é fundamental que organizações de mais alta maturidade saibam identificar e controlar a estabilidade e capacidade dos sub-processos (CHRISSIS et al., 2006).

Todos os processos são projetados para produzir resultados. Os produtos e serviços gerados a partir de processos possuem atributos quantitativos. Segundo PRESSMAN (2006), esses atributos permitem aos engenheiros de software ter ideia da eficácia do processo de

software e dos projetos que são conduzidos utilizando o processo como arcabouço. Esses dados são analisados, comparados com médias anteriores e avaliados para verificar se houve melhorias de qualidade e produtividade. Além disso, também é possível detectar áreas de problema, de modo que soluções possam ser desenvolvidas, e então o processo de software possa ser melhorado.

Na maioria das vezes, quando as medidas coletadas do desempenho dos processos variam, define-se que o processo não está sob controle. Processos controlados são aqueles que possuem uma variabilidade estável. Sem estabilidade, não é possível fazer predições para os resultados dos processos e ainda não há como considerar o processo em questão como processo repetível a ser usado como base para melhorias (FLORAC e CARLETON, 1999).

O processo estável pode ser definido como um processo cuja característica principal é a previsibilidade. Uma vez que o desempenho e a variabilidade são conhecidos, é possível planejar estimativas utilizando como base seu desempenho no passado (WHEELER e CHAMBERS, 1992).

O fato de um processo ser estável não significa que ele tem um bom desempenho. Para que isso ocorra, ele também deve ser um processo capaz. A análise de capacidade de um processo abrange os requisitos definidos pela organização. Mesmo que um processo tenha seu desempenho conhecido e estável, ele pode não atender às especificações definidas. Neste caso, diz-se que o processo é estável, mas não é capaz de atender às especificações definidas para um determinado cenário de utilização. Quando um processo é estável e possui uma conformidade com os requisitos, diz-se que o processo é capaz (CAMPOS et al. 2007). Exemplificando, imagine que seja segunda-feira, e é preciso adquirir um livro, que deverá ser lido até a próxima segunda-feira. Existem duas maneiras de obter o livro: ir à livraria mais próxima ou realizar a compra pela internet. Adquirindo o exemplar pela internet, há grandes chances dele não ser entregue até sexta-feira, ainda mais se o pagamento for efetuado através de boleto bancário, visto que o processo demora até dois dias úteis para a compensação bancária. Já se dirigindo à livraria, é possível obter o livro no mesmo dia e então iniciar a leitura. Supondo que a livraria tenha o livro e a leitura seja conduzida com disciplina, pode-se afirmar que o processo de ir à livraria é estável e capaz. Concluindo, o conceito de capacidade depende da estabilidade do processo e da habilidade do processo estar aderente aos requisitos do cliente.

Na definição de processos, todas essas informações referentes à capacidade e estabilidade dos sub-processos são muito importantes para a composição dos processos

definidos. Assim, um dos critérios a ser utilizado no momento de escolher os sub-processos pode ser a existência de dados sobre seu desempenho. Além disso, os objetivos de qualidade e desempenho do processo a ser definido também devem ser levados em consideração. Pode-se avaliar se os sub-processos apresentaram comportamento estável em utilizações anteriores em contextos comparáveis. É importante também considerar se os dados de desempenho dos sub-processos satisfazem aos objetivos de qualidade e desempenho do projeto que irá fazer uso do sub-processo (CHRISSIS et al., 2006).

2.4 CONSIDERAÇÕES FINAIS

Este capítulo abordou sobre Processos de Software através de uma revisão da literatura, de forma a prover uma contextualização e base para o entendimento dos próximos capítulos que o sucedem. Os conceitos principais sobre o tema foram abordados de forma sucinta, enfatizando o cenário da definição de processos. Esse contexto é onde o Process Broker está inserido, tendo um papel como agente facilitador no momento das organizações desenvolvedoras de software definirem seus processos, através da disponibilização de Componentes de Processo oriundos das instituições implementadoras. Esse processo será mais bem detalhado nos próximos capítulos.

Além disso, foram descritas algumas normas internacionais e modelos de maturidade relacionando-as com a questão da definição de processos.

Por fim, conceitos como a estabilidade e capacidade de processos foram apresentados, denotando a sua extrema importância no momento da definição de processos. Esses conceitos podem ser relevantes em um futuro aprimoramento do Process Broker, onde as organizações desenvolvedoras de software disponibilizariam informações referente à estabilidade e capacidade dos Componentes de Processo em seus projetos. Por uma limitação de tempo, o escopo atual do Process Broker não abrange esse ponto.

CAPÍTULO 3 – REUTILIZAÇÃO DE PROCESSOS DE SOFTWARE ATRAVÉS DE COMPONENTES DE PROCESSO DE SOFTWARE

3.1 INTRODUÇÃO

A reutilização de Processos de Software é uma abordagem importante da Engenharia de Software, onde as informações adquiridas em projetos de desenvolvimento anteriores são reutilizadas com a intenção de produzir uma melhoria contínua de processos. Essa abordagem tem como consequência a diminuição do esforço e custo necessário para estabelecer um processo para um novo projeto (COSTA *et al.*, 2007).

A abordagem de reutilização de Processos de Software já recebeu bastante atenção dos pesquisadores e da indústria em meados da década de 90, e atualmente é possível identificar pesquisas para o armazenamento e a definição de elementos reutilizáveis de processo (ELLMER *et al.*, 1996).

Segundo HOLLEBANCH e FRAKES (1996), pode-se diminuir em pelo menos dez vezes o tempo e o esforço necessário para definir um processo de software para um projeto quando se utiliza um processo reutilizável ao invés de criá-lo desde o princípio. De acordo com esses autores, para possibilitar a reutilização é necessário uma abordagem que identifique os elementos comuns e variantes de processos específicos de projeto e crie definições de processos que possam ser reutilizadas.

A reutilização de processo de software não se limita apenas ao reuso de definições de processo de software, mas trata também da reutilização de informações relacionadas às

execuções do processo, ou seja, conhecimentos e experiências adquiridas em determinados contextos (RU-ZHI *et al.*, 2005).

Este capítulo dá continuidade à revisão bibliográfica desta monografia. Além de se discutir sobre a reutilização de Processos de Software, ele também aborda os conceitos-chave sobre Componente de Processo de Software, que são referenciados mais adiante, no Capítulo 4.

3.2 COMPONENTES DE PROCESSO DE SOFTWARE

Para permitir a reutilização de Processos de Software, é necessário definir uma forma de como o conhecimento relacionado a Processos de Software deve ser estruturado e explicitado. Uma delas é através de Componentes de Processo de Software.

Várias obras da literatura apontam semelhanças entre software e Processos de Software, por exemplo OSTERWEIL (1987). Também é possível observar analogias entre a reutilização de software e a reutilização de Processos de Software. Técnicas de reutilização de software como a utilização de componentes têm sido desenvolvidas também para Componentes de Processo.

No contexto de software, componentes são elementos fundamentais que servem como unidade de encapsulamento com interfaces bem definidas e podem ser reutilizados ou substituídos, beneficiando a produtividade (MURTA, 2006).

De forma análoga, um Componente de Processo pode ser encarado como um encapsulamento de informações e comportamentos de processo em um dado nível de granularidade (GARY e LINDQUIST, 1999).

Na literatura não há um consenso sobre quais informações devem estar contidas em Componentes de Processo de Software, havendo uma variação para cada abordagem de como são descritos e estruturados.

Segundo o SPEM (2006) – *Software Process Engineering Metamodel*, um Componente de Processo é considerado um agrupamento de descrições de processo que é internamente consistente e pode ser reutilizado com outros Componentes de Processo para compor um processo completo. Deve ser autocontido, ou seja, não ter referências de dentro de um componente para elementos que estão fora do componente.

Componentes de processo são decomposições de um processo de software. Podem ser utilizados para compor um processo de software completo, através da aglutinação a outros componentes. Partindo da premissa que um processo de software completo pode ser decomposto em sub-processos, chega-se à conclusão que Componentes de Processos podem ser vistos como sub-processos. Dessa forma, eles podem representar sub-processos como previstos em modelos como o CMMI-DEV (CHRISSIS *et al.*, 2006) e o MPS.BR (SOFTEX, 2007).

A abordagem de BARRETO (2007), a qual é relevante para esse trabalho, estabelece que um Componente de Processo poderá existir em qualquer nível de detalhamento, ou seja, abrangendo desde uma Atividade a um processo completo. Entretanto, um componente deve ser formado por no mínimo uma Atividade, pois uma unidade inferior a isso só traria mais complexidade desnecessária e dificultaria a composição de processos. Na ótica dessa abordagem, um Componente de Processo e Atividade são elementos de processo, que são decomposições de um processo de software, em um dado nível de granularidade.

Entende-se como Atividade um conjunto de passos fixos necessários para se alcançar um objetivo. Além desse conceito usual, BARRETO (2007) define, no contexto de sua abordagem, que uma Atividade deve ser completamente definida, ou seja, não apresenta variabilidade. O que diferencia um Componente de Processo de uma Atividade é que um componente é definido para a reutilização. Um Componente de Processo é normalmente algo em mais alto nível que uma Atividade, de modo a possibilitar a um engenheiro de processo compor seus processos a partir de blocos maiores, o que facilitaria o seu reuso. Além disso, Componentes de Processo apresentam características que Atividades não possuem, como possuir rastreabilidade para modelos de maturidade. Essa diferenciação entre Componente de Processo e Atividade fica mais evidente na seção 3.4, onde é discutida a estrutura de um componente e uma Atividade.

3.2.1 Arquiteturas de Processo de Software

Compor um processo por inteiro utilizando apenas componentes parece não ser suficiente. É necessária uma estrutura reutilizável maior, como uma linha de processo (WASHIZAKI, 2006). Para explicar esse conceito, é necessário sair do contexto de processo

de software para o de software em si, definindo primeiro a arquitetura de processo de software.

Uma arquitetura de software “normal” define a estrutura de um único sistema de software. Já uma arquitetura de processo representa um fluxo de trabalho, sendo composta por Componentes de Processos, Atividades ou qualquer combinação entre eles. Ela define uma estrutura que o processo deve possuir, e nela estão os principais elementos e como eles se relacionam, sem que o conteúdo dos elementos seja detalhado (BARRETO, 2007).

Uma arquitetura de processo que apresenta pontos de variação, por conseqüência, permite a variabilidade, pois há diversas maneiras de atender à arquitetura em questão, dependendo da seleção das variantes escolhidas. Uma arquitetura de processos pode conter mais de um ponto de variação, componentes totalmente definidos, ou seja, que não permitem variação e devem ser implementados conforme descritos, e, finalmente, pode conter também componentes opcionais, que podem ou não estar presentes na arquitetura. Esse tipo de arquitetura que apresenta pontos de variação, elementos obrigatórios e opcionais, é definido como linha de processos, na abordagem de BARRETO (2007).

Novamente no contexto de software, uma arquitetura de linha de produtos define a estrutura para um conjunto de produtos relacionados (GARG *et al.*, 2003). Ela distingue os elementos obrigatórios, que devem estar presentes em todas as arquiteturas de produtos, e pontos de variação, que apresentam diferenças entre as arquiteturas de produtos. Três tipos de pontos de variação são usados para diferenciar uma arquitetura da outra (GARG *et al.*, 2003): (i) elementos opcionais, que podem ou não estar presentes em uma arquitetura de produtos particular; (ii) elementos variantes, que devem estar presentes e podem ser instanciados via uma ou várias alternativas; (iii) elementos variantes opcionais, que são elementos variantes, porém podem ou não estar presentes. A partir daí, conclui-se que possa haver quatro classificações para um elemento na arquitetura de linha de produtos: opcional variante, opcional invariante, obrigatório variante e obrigatório invariante.

Uma linha de produtos pode ser vista como uma fábrica que produz produtos semelhantes, cada um com um conjunto de características, através da composição de componentes. Seguindo esse raciocínio, acredita-se que Processos de Software podem ser elaborados a partir de Componentes de Processo pré-existentes, sendo que cada instância seria composta por um conjunto de características específicas (BARRETO, 2007).

Analogamente, linhas de Processos de Software são linhas de produtos cujos produtos são Processos de Software. Conceitos como variabilidade e opcionalidade também estão

presentes. A variabilidade é representada através de pontos de variação e variantes de processo. Pontos de variação são componentes que podem ser alterados de acordo com uma necessidade específica, como por exemplo, um projeto. Em outras palavras, são Componentes de Processo que admitem diversas estruturas internas, que podem ser realizadas por diversas variantes. Já variantes de processo são possíveis implementações de um ponto de variação (WASHIZAKI, 2006).

Ainda relacionado à arquitetura de processo, BARRETO (2007) define que um Componente de Processo pode possuir uma arquitetura interna. Essa arquitetura permite que um componente seja detalhado e decomposto em outros componentes. Entretanto, pode haver componentes sem arquitetura interna. Neste caso, o componente terá o seu propósito definido, sendo executado sem seguir uma estrutura definida. A arquitetura interna será melhor detalhada mais adiante.

3.3 CLASSIFICAÇÃO DOS COMPONENTES DE PROCESSO DE SOFTWARE

Considerando a abordagem adotada, ao analisar um Componente de Processo em relação a uma arquitetura de processo, pode-se classificá-lo em três tipos: componente obrigatório, componente opcional e componente variante. Componente obrigatório é o componente que não admite variabilidades e deve estar contido no processo da forma que é descrito. Em contrapartida, um componente opcional pode ou não estar no processo, ficando a critério do engenheiro de processo. Por fim, componente variante é aquele que possui uma possível implementação de um ponto de variação, ou seja, é um candidato no momento de selecionar um dos componentes em um ponto de variação.

Além das três classificações já apresentadas, pode-se ainda classificar um componente, quando analisado isoladamente, em concreto ou abstrato. Componente concreto é aquele que não admite variabilidade, ou seja, deve ser executada da forma que é descrito, sem qualquer variação. Todo componente concreto possui uma arquitetura interna definida, composta por outros componentes concretos ou Atividades. Em um componente concreto não há decisões para serem tomadas, tudo já é definido por completo, podendo até já ser utilizado em um processo. Por outro lado, um componente abstrato admite variabilidades, isto é, ele não está atrelado a apenas uma forma de realização. Um componente abstrato pode ou não ter uma

arquitetura interna. Um componente abstrato não está pronto de imediato para ser executado em um processo de software, pois sua definição não é completa e por isso é necessário que escolhas ainda sejam feitas pelo engenheiro de processo (BARRETO, 2007).

Para melhor entendimento dessas diversas classificações, considere a linha de processo representada pela Figura 2. O primeiro componente da linha de processo, da esquerda para a direita, é o componente concreto “Planejar o Processo”. Por ser concreto, ele não admite variabilidades e deve ser instanciado no processo a ser definido, através da linha de processo, exatamente como foi definido. Em seguida tem-se o componente “Planejar Projeto”. Observe que ele tem a sua arquitetura interna detalhada. Com isso tem-se um nível mais fino de detalhamento. Ele é composto por outros três componentes: “Definir Estimativas”, “Criar Planos” e “Obter Comprometimento”. Todos são componentes concretos, exceto o componente “Definir Estimativas”. Este, por sua vez, trata-se de um componente abstrato, ou seja, admite variabilidade, desempenhando na linha de processo o papel de ponto variação. Ele possui três formas distintas de implementação, ou seja, três variantes: “Experiência”, “Pontos Caso de Uso” e “Pontos de Função”. Portanto, o componente abstrato “Definir Estimativas” não deve ser instanciado de imediato no processo a ser definido, pois é preciso que o engenheiro de processo selecione uma das suas três variantes para que possa de fato ser utilizado. Prosseguindo na linha de processo, tem-se o componente “Avaliação de Qualidade”. Ao analisá-lo em relação à linha de processo, pode-se classificá-lo como componente opcional, ou seja, ele pode ou não estar instanciado no processo a ser definido. Ainda nessa perspectiva de classificação, todos os outros componentes da linha de processo são componentes obrigatórios, ou seja, diferentemente dos componentes opcionais, devem estar presentes no processo a ser definido. Além disso, o componente “Definir Estimativas” pode ser classificado como ponto de variação. Já os componentes presentes em sua arquitetura interna, são classificados como componentes variantes, pois são possíveis implementações do ponto de variação “Definir Estimativas”.

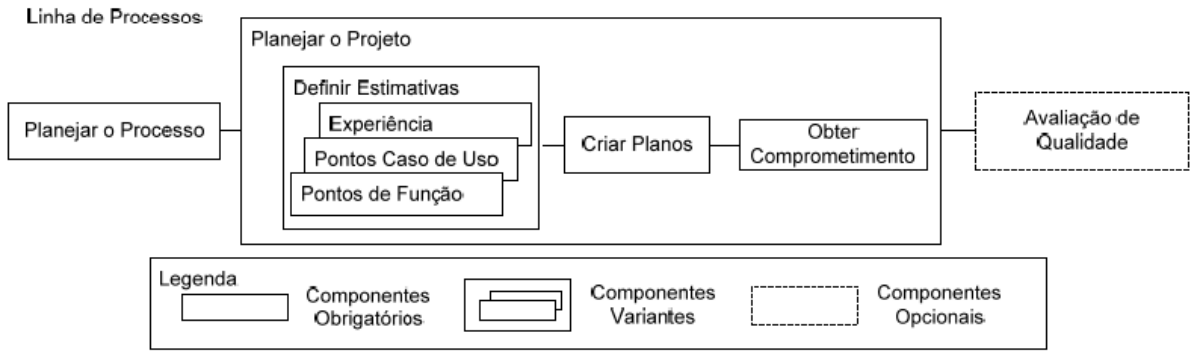

Figura 2: Exemplo de Linha de Processos (BARRETO *et al.*, 2009)

A Figura 3 exemplifica uma possível derivação da linha de processos apresentada na Figura 2. O processo derivado optou por ter escolhido "Experiência" como variante a implementar o ponto de variação "Definir Estimativas", e ainda preferiu não incluir o componente opcional "Avaliação de Qualidade".

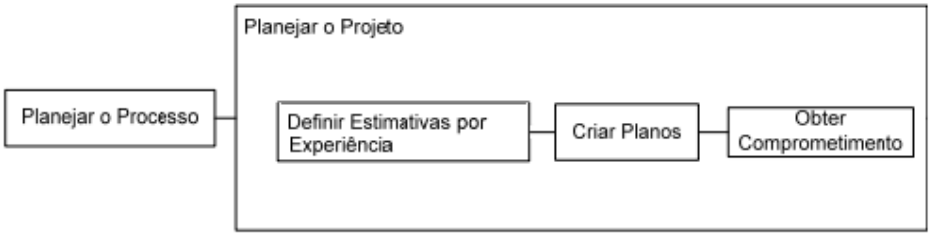

Figura 3: Exemplo de Processo derivado de uma Linha de Processos

A Figura 4 também representa um processo derivado da linha de processos da Figura 2. Dessa vez, foi escolhido "Pontos de Função" como variante a implementar o ponto de variação "Definir Estimativas", e além disso, há a inclusão do componente opcional "Avaliação de Qualidade". O processo da Figura 3 poderia ser uma boa solução para um projeto menos crítico, enquanto que o da Figura 4 para um projeto mais crítico, devido ao seu grau de formalidade, pois as estimativas serão realizadas através da técnica de pontos de função e ainda há uma preocupação com a qualidade do que venha a ser desenvolvido.

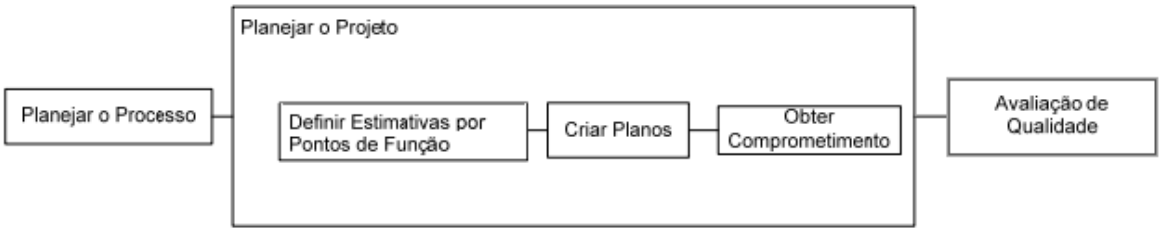

Figura 4: Exemplo de Processo derivado de uma Linha de Processos

3.4 ESTRUTURA DE COMPONENTES DE PROCESSO DE SOFTWARE

Ainda não há um consenso sobre quais informações um Componente de Processo deve conter, existindo variações em cada abordagem. Entretanto, essa seção tem como objetivo descrever a estrutura de um Componente de Processo de Software, segundo a visão de BARRETO (2007). Essa estrutura é utilizada no Capítulo 4 pelo Process Broker.

O Componente de Processo de Software deve conter um conjunto de informações necessárias, de modo que possibilite ao profissional o conhecimento necessário na escolha de um componente. Em conformidade com a abordagem adotada, um Componente de Processo de Software deve possuir os seguintes atributos:

- Identificador: código que identifica um componente.
- Tipo: classifica o componente em concreto e abstrato, ou seja, se admite variabilidades ou não.
- Nome: apresenta o nome do Componente de Processo. Este atributo tem a finalidade de identificar o componente com um nome único.
- Descrição: apresenta uma breve descrição do componente. Essa descrição tem por objetivo fornecer ao engenheiro uma contextualização em alto nível acerca do componente. A partir dessa descrição, pode-se julgar se o componente é candidato à escolha.
- Critérios de Entrada: estabelece uma pré-condição para a execução do componente, ou seja, o que deve ser verdade para que o componente possa ser executado.
- Critérios de Saída: estabelece uma pós-condição ao término da execução do componente, ou seja, o que deve ser verdade após a execução do componente.
- Organização de Definição: indica qual a procedência do Componente de Processo, em

outras palavras, cita a organização que o definiu. Geralmente este campo será preenchido com o nome de uma Instituição Implementadora, mas nada impede que as organizações desenvolvedoras de software possam definir seus próprios componentes, caso sejam capazes. Em uma biblioteca de componentes, que naturalmente tende a crescer com um tempo, identificar quem definiu cada componente é importante para saber a quem delegar uma modificação, caso seja necessário uma alteração de caráter corretivo ou evolutivo.

- Responsável: identifica qual é o perfil responsável pela execução do componente.
- Participantes: descrevem quais são os outros perfis que participam da execução do Componente de Processo, que não necessariamente são responsáveis pela execução, mas que de certa forma estão envolvidos com aquele componente;
- Artefatos Requeridos (Parâmetros de Entrada): estabelecem os produtos de trabalho que são insumo para a execução do componente;
- Artefatos Produzidos (Parâmetros de Saída): indicam os produtos de trabalho que são gerados a partir da execução do componente;
- Ferramentas de Apoio: listam quais ferramentas podem ser utilizadas para apoiar a execução do processo;
- Características Atendidas: determinam quais características que o componente atende. Uma característica é como uma espécie de funcionalidade ou classificação que o processo deve possuir, por exemplo: apoio ao nível G do MPS.BR;
- Características Conflitantes: determinam quais características que conflitam com o componente, ou seja, se a característica for escolhida, o componente não deve ser selecionado;
- Variações: indica de qual outro componente um dado componente é variante;
- Arquitetura Interna: detalha da estrutura interna do componente, caso possua outros componentes e/ou Atividades em seu interior.

Para exemplificar a estrutura de um Componente de Processo, suponha o detalhamento seguinte para o primeiro componente da linha de processo representado pela Figura 2, o componente “Planejar o Processo”. Esse componente tem “CON0001” como identificador, “Planejar o Processo” como seu nome e possui a seguinte descrição: “O objetivo do componente é definir o plano do processo para o projeto e instanciar o ambiente de apoio à execução do processo. Envolve: (i) Identificar/Entender Requisitos do Cliente e Requisitos do Software; (ii) Planejar Processo; (iii) Avaliar o Planejamento do Processo pelo GQPP; (iv)

Avaliar e Obter Comprometimento com o Plano do Processo; e (v) Instanciar Ambiente". A partir dessa descrição sabe-se qual o propósito geral do Componente de Processo, e então o engenheiro de processo decidirá se o componente é um candidato a ser selecionado conforme a sua demanda. Uma vez que o componente seja identificado como possível candidato, deve-se então analisar em mais baixo nível o componente, ou seja, informações mais detalhadas, para só então efetuar a seleção. Prosseguindo na análise do componente, seu tipo é descrito como "Concreto", ou seja, não admite variabilidades e deve ser utilizado exatamente conforme fora definido. O "Gerente de Projeto" é identificado como responsável, ou seja, é atribuído ao gerente de projeto a responsabilidade de executar o planejamento do processo. Ainda no que tange aos perfis de recursos humanos, o componente possui como participantes: "Cliente; Membro do GQPP (Grupo da Garantia e Qualidade do Processo e do Produto); e Revisor Técnico." Esses três perfis listados não têm a responsabilidade de planejar o processo, porém possuem participações fundamentais nessa etapa. O componente possui como artefatos requeridos os seguintes itens: "Levantamento de requisitos; Proposta de desenvolvimento; Proposta técnica e comercial; e Solicitação de serviço". Para que o planejamento do processo seja efetivamente planejado é fundamental que esses quatro artefatos estejam disponíveis, pois eles servirão de insumo. Não será possível planejar o processo, segundo o componente, caso falte um desses artefatos. O componente descreve como artefatos produzidos os seguintes itens: "Ambiente do projeto; Estrutura analítica do projeto; Modelo de análise e projeto; Plano do Processo". Esses quatro itens serão necessariamente produzidos durante o planejamento do processo. "*AdaptoPro* (Estação TABA[1]); *Enterprise Architect* (UML); Ferramenta de *e-mail*; MS Word (editor de texto)" são definidas como ferramentas pelo componente. Essas são todas as ferramentas necessárias que suportam o planejamento de processo. O componente "Planejamento do Processo" foi definido pelo "LENS (Área de Qualidade do Laboratório de Engenharia de Software da COPPE UFRJ)", portanto esta será sua organização de definição. Por fim, o componente ainda descreve suas características. São descritas como características atendidas: "CMMI – Nível 2; MPS.BR – Nível F; e Implantação usando TABA". Em outras palavras, o componente é aderente com os modelos de maturidade CMMI e MPS.BR no nível 2 e F respectivamente, e além disso sua implantação é orientada pela estação TABA. E por fim, possui "Implantação não usando

[1] A estação TABA é um ambiente de desenvolvimento de software centrado em processos, e foi desenvolvida a partir de diversos trabalhos acadêmicos no contexto da COPPE/UFRJ, desde 1990.

TABA” como característica conflitante, ou seja, esse componente não deve ser selecionado caso o planejamento do processo não seja através da estação TABA.

É possível observar no exemplo anterior que não foram citados os atributos: critérios de entrada, critérios de saída, variação e arquitetura interna. Não é obrigatório que um Componente de Processo contenha todos os atributos.

Uma Atividade de processo é menos complexa que um Componente de Processo. Ela possui a mesma estrutura de um Componente de Processo, excetuando-se os atributos: identificador, tipo, organização de definição, características atendidas, características conflitantes, variações e arquitetura interna.

3.4.1 Estrutura da Arquitetura Interna de um Componente de Processo de Software

A arquitetura interna, por ser o atributo de um Componente de Processo que possui uma estrutura mais complexa, merece um melhor destaque. Uma arquitetura interna é formada por um conjunto de conexões. Essas conexões, por sua vez, contêm itens de uma arquitetura de processo.

Itens de uma arquitetura interna podem ser compreendidos em item de início da arquitetura, item de fim da arquitetura e item de elemento de processo. Itens de início e fim de arquitetura apenas representam o estado inicial e final de uma arquitetura interna de um Componente de Processo respectivamente, ou seja, marcam o início e o fim do fluxo de uma arquitetura interna. Já os itens de elemento de processo são os elementos de processo que fazem parte da arquitetura, sendo que nesta abordagem podem ser um Componente de Processo, seja ele concreto ou abstrato, ou uma Atividade.

Existem cinco tipos diferentes de conexão em uma arquitetura interna de um Componente de Processo. Essa distinção entre as conexões serve para identificar os possíveis fluxos em uma conexão. Seguem listados abaixo os tipos de conexões e suas definições:

- Fim-Fim: este tipo de dependência considera que uma tarefa deve continuar a sua execução apenas se a outra também estiver executando. Se uma termina, a outra também finaliza.
- Fim-Início: indica que uma tarefa sucessora começa a ser executada apenas quando a predecessora termina.

- Início-Fim: determina que se uma tarefa começa a outra termina. Exemplo: ao iniciar uma tarefa, a predecessora pode ser finalizada, pois não é mais necessária, ou seja, ela perde o sentido para que seja executada devido à execução da sua sucessora.
- Início-Início: este tipo dependência indica que o início de uma tarefa desencadeia a inicialização da outra.
- Simples: é utilizado apenas para conectar itens início e fim a itens de elementos de processos.

Considere o exemplo da Figura 5. Ela representa a arquitetura interna de um componente. Esta imagem foi gerada a partir da aplicação desenvolvida por BARRETO (2007), que será melhor abordada no capítulo seguinte. A arquitetura interna citada contém quatro itens de arquitetura e três conexões. O item de início de arquitetura é representado graficamente pelo círculo preto. Por sua vez, o item de fim de arquitetura é representado através do círculo preto envolvido por uma circunferência. Eles indicam o início e o fim de uma arquitetura interna. Por outro lado os itens de elementos de processo são de extrema importância, pois são eles que efetivamente definem a estrutura interna do componente. No exemplo, eles são representados pelo componente concreto “Planejar Riscos” e pela Atividade “Planejar Processo”. A primeira conexão é do tipo Simples, pois envolve o item de início de arquitetura como item origem da conexão, e tem o Componente de Processo concreto “Planejar Riscos” como item destino. A segunda conexão é do tipo Fim-Início, e engloba o componente concreto “Planejar Riscos” como item origem e a Atividade “Planejar Processo” como item destino da conexão. Em outras palavras, a conexão desse trecho da arquitetura interna informa que o planejamento do processo só será iniciado quando for concluído o planejamento de risco. E por fim, a terceira conexão é do tipo Simples, pois conecta a Atividade “Planejar Processo” ao item fim de arquitetura.

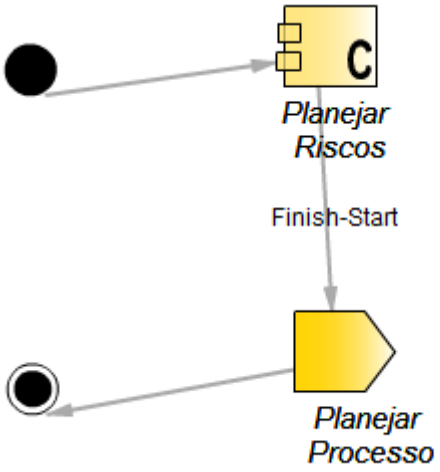

Figura 5: Arquitetura interna de um Componente de Processo

3.5 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado resumidamente todo o enfoque teórico que é efetivamente utilizado na implementação do Process Broker, que em suma é a abordagem proposta por BARRETO (2007).

O conceito de Componente de Processos foi abordado gradativamente, desde sua idealização, em alto nível, até a sua estrutura interna, em mais baixo nível. Além disso, esse conceito é a peça chave de todo esse trabalho de conclusão de curso, pois Componentes de Processo de Software são o insumo e o produto do Process Broker, que será detalhado no capítulo seguinte.

CAPÍTULO 4 - PROCESS BROKER

4.1 INTRODUÇÃO

Os Componentes de Processos podem ser armazenados em diferentes organizações e em diferentes representações e, por esse motivo, para tornar viável a reutilização e troca de Componentes de Processos entre as diferentes organizações. Observa-se a necessidade de uma padronização na forma da representação desses componentes bem como a utilização de um ambiente que possibilite essa troca. Através da tecnologia XML, é possível gerar uma linguagem para a representação de Componentes de Processo de Software, assim como a validação desses componentes através de um XML Schema, o qual pode possuir um conjunto de regras de formação para um Componente de Processo. O Process Broker é um ambiente que possibilita a importação e exportação de Componentes de Processo de Software em XML.

Cada organização pode ter uma versão do Process Broker, com seus próprios componentes, podendo ou não existir uma organização que funcione como base central com todos os Componentes de Processo existentes. Para importar componentes de uma organização para outra, bastaria acessar o ambiente Process Broker da organização alvo e, utilizando o Exportador, exportar os componentes desejados. Assim, poderíamos importar estes componentes para a base de dados local através do Importador do Process Broker. Esse cenário é representado na Figura 6.

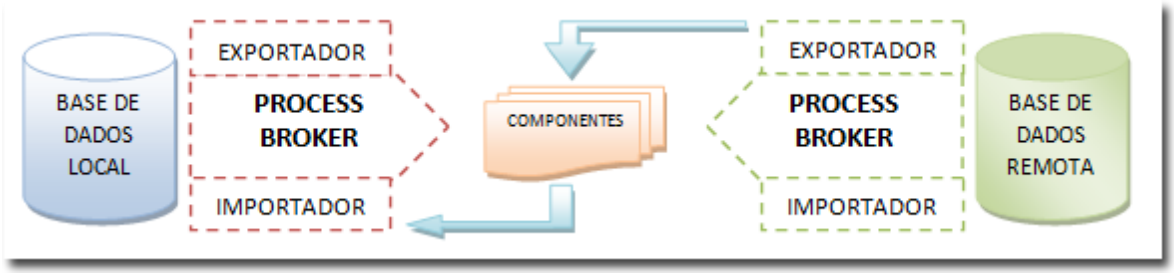

Figura 6: Esquema de funcionamento do Process Broker

O ambiente Process Broker está integrado com o ambiente A2M que tem por objetivo apoiar a definição de Processos de Software e Componentes de Processo de Software, oferecendo ao usuário, entre outras funções, um conjunto de ferramentas que auxiliam nessa definição.

O ambiente A2M possui um banco de dados relacional e uma API para acessar esses dados. O banco de dados contém, entre outros dados, Componentes de Processo de Software que seguem a definição feita por BARRETO (2007) como descrita no Capítulo 3. A API do ambiente A2M disponibiliza, entre outras funções, funções para acesso, consulta e inserção de dados no banco de dados.

O Process Broker acessa as informações dos Componentes de Processo através da API do ambiente A2M. O Importador e Exportador do Process Broker faz uso internamente de algumas dessas funções da API para exportar e importar Componentes de Processo da base de dados local do A2M. Porém, o Process Broker não tem o objetivo de definir Componentes de Processo, mas utiliza as mesmas definições utilizadas no ambiente A2M para representá-los. O objetivo da integração do ambiente do Process Broker com o ambiente A2M é possibilitar a utilização do Process Broker, dando mais visibilidade ao mesmo.

A estrutura do Process Broker está definida conforme apresentado na Figura 7. Duas operações principais fazem parte do Process Broker, a importação de componentes, executada pelo Importador e a exportação de componentes, executada pelo Exportador. Ambos utilizam o Validador e o XML Schema para validar os Componentes de Processo em XML que estejam sendo importados ou exportados.

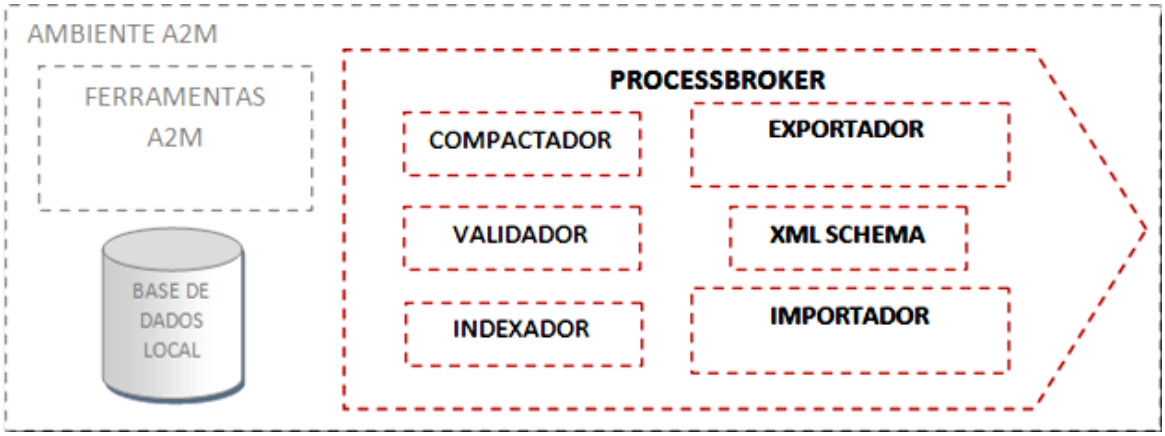

Figura 7: Estrutura do Process Broker

O ciclo de exportação de Componentes de Processo é apresentado na Figura 8. Ao acessar o Process Broker pode-se selecionar os Componentes de Processo da base de dados que se deseja exportar através da interface do Exportador (Figura 23). Após essa seleção, o Exportador exporta cada um dos Componentes de Processo selecionados da base de dados para o formato XML. Esses componentes são validados com o XML Schema, são indexados pelo Indexador gerando um arquivo de índice e por fim todos esses arquivos exportados são compactados em um arquivo ZIP pelo Compactador. Esse arquivo ZIP é a saída do Exportador.

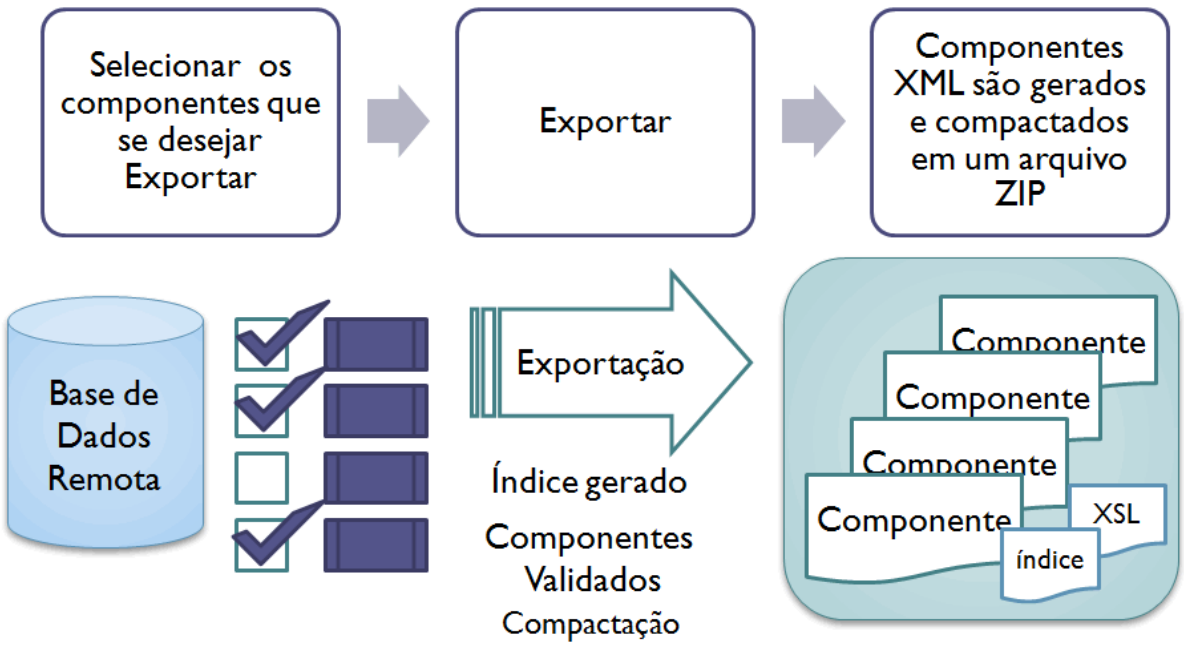

Figura 8: Ciclo de exportação de Componentes de Processo

O ciclo de importação de Componentes de Processo é apresentado na Figura 9. Através da interface do Importador (Figura 25), deve-se selecionar um arquivo ZIP contendo componentes de processo XML previamente exportado por um Exportador do Process Broker. Internamente esse arquivo ZIP é descompactado pelo Compactador e o arquivo de índice é consultado, assim pode-se selecionar quais dos componentes XML importar para a base de dados (Figura 26). Após essa seleção, O Importador importa cada um dos componentes XML verificando sua validade de acordo com o XML Schema e os inserido na base de dados local. Este capítulo apresenta cada uma das estruturas que compõem o ambiente Process Broker, bem como a abordagem adotada em sua implementação.

Figura 9: Ciclo de importação de Componentes de Processo

4.2 XML, XML SCHEMA e XSLT

Através da utilização da tecnologia XML, é possível gerar uma linguagem para a representação de Componentes de Processo de Software, assim como a validação desses componentes, através de um XML Schema, que possui um conjunto de regras de formação para um Componente de Processo.

A tecnologia XML possibilita a representação de um Componente de Processo através de elementos previamente definidos no XML Schema. Para que estes componentes

representados em XML sejam válidos, devem obedecer às regras de formação estabelecidas no XML Schema dos Componentes de Processo. A abordagem do XML Schema atual define os seguintes elementos que formam um Componente de Processo em XML: Identificador, Tipo, Nome, Descrição, Critérios de Entrada, Critérios de Saída, Organização de Definição, Responsável, Parâmetros de Entrada, Parâmetros de Saída, Recursos Humanos, Ferramentas de Apóio, Características Atendidas, Características Conflitantes, Variação e Arquitetura Interna. A Figura 10 mostra o diagrama do XML Schema para Componentes de Processo e o XML Schema completo pode ser obtido no anexo A.

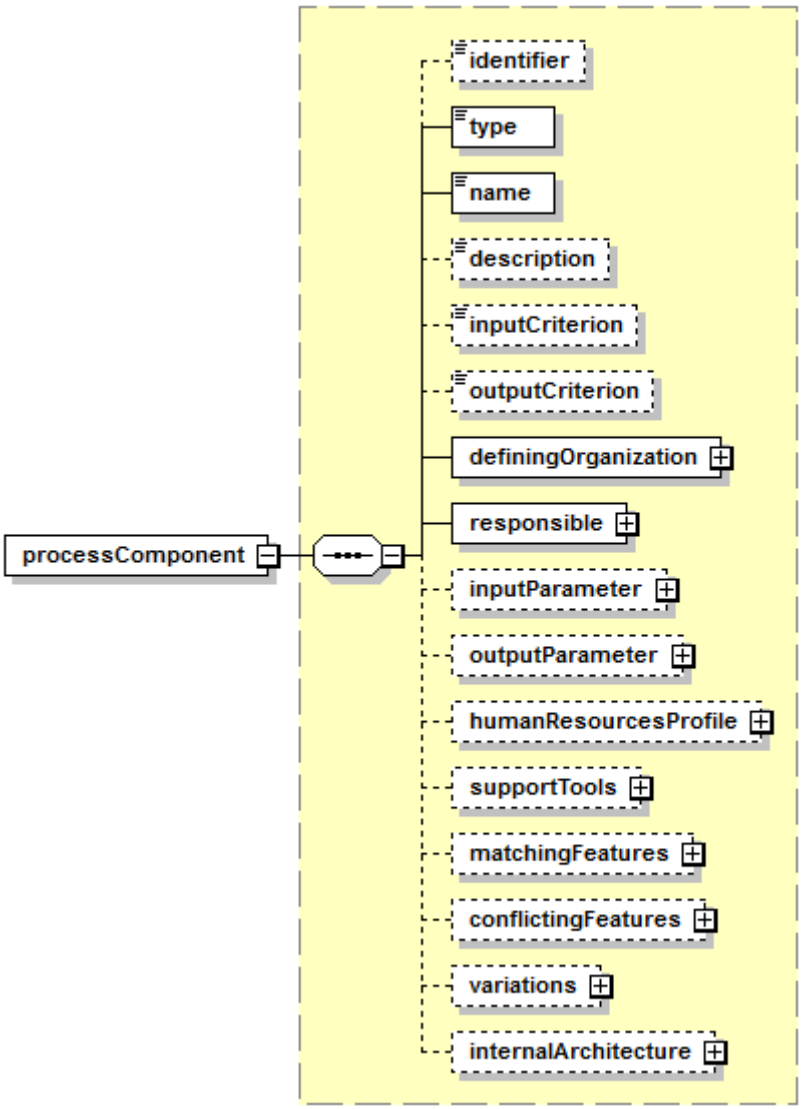

Figura 10: Diagrama do XML Schema para Componentes de Processo

O XML Schema criado para representar Componentes de Processo define o elemento raiz obrigatório <*processComponent*>. O elemento <*processComponent*> por sua vez é definido pela sequência de elementos: <*identifier*>, <*type*>, <*name*>, <*description*>, <*inputCriterion*>, <*outputCriterion*>, <*definingOrganization*>, <*responsible*>, <*inputParameter*>, <*outputParameter*>, <*humanResourcesProfile*>, <*supportTools*>, <*matchingFeatures*> , <*conflictingFeatures*>, <*variations*> e <*internalArchitecture*>. Alguns desses elementos possuem conteúdo, outros são formados por outros elementos, alguns deles são obrigatórios e outros são opcionais. Esses elementos são descritos a seguir.

O elemento opcional <*identifier*> é uma cadeia de caracteres que representa o identificador do componente. Atualmente o identificador não é obrigatório devido à sua definição dentro do banco de dados no ambiente A2M.

O elemento obrigatório <*type*> representa o tipo do componente e possui somente dois possíveis valores: “Abstrato” e “Concreto”.

O elemento obrigatório <*name*> é uma cadeia de caracteres que representa o nome do componente. Dentro do ambiente A2M, cada nome de um Componente de Processo é único.

O elemento opcional <*description*> é uma cadeia de caracteres que possui a descrição do componente.

O elemento opcional <*inputCriterion*> é uma cadeia de caracteres que possui critérios de entrada do componente.

O elemento opcional <*outputCriterion*> é uma cadeia de caracteres que possui critérios de saída do componente.

O elemento obrigatório <*definingOrganization*>, mostrado na Figura 11, é composto pela sequência de elementos: <*name*>, <*description*>, <*mission*> e <*isImplementingInstitution*>. O elemento obrigatório <*name*> é uma cadeia de caracteres que representa o nome da organização. O elemento opcional <*description*> é uma cadeia de caracteres que possui a descrição da organização. O elemento opcional <*mission*> é uma cadeia de caracteres que possui a missão da organização. O elemento obrigatório <*isImplementingInstitution*> mostra se a organização é uma instituição implementadora ou não e possui somente dois possíveis valores: “*true*” e “*false*”.

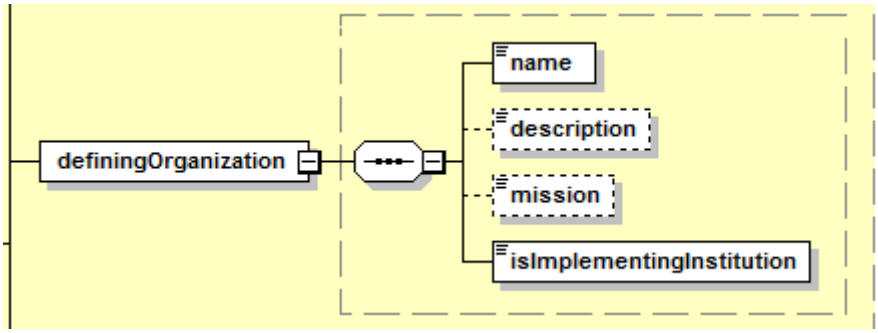

Figura 11: Estrutura do elemento <*definingOrganization*>

O elemento obrigatório <*responsible*>, mostrado na Figura 12, é composto pela sequência de elementos: <*name*> e <*description*>. O elemento obrigatório <*name*> é uma cadeia de caracteres que representa o nome do responsável. O elemento opcional <*description*> é uma cadeia de caracteres que possui a descrição do responsável.

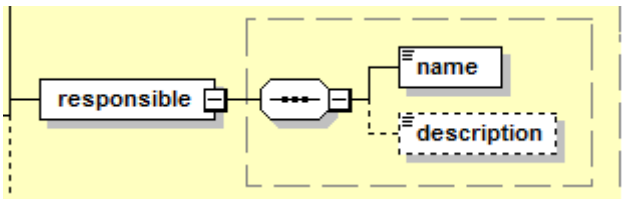

Figura 12: Estrutura do elemento <*responsible*>

O elemento opcional <*inputParameter*>, mostrado na Figura 13, é composto por um ou mais elementos <*parameter*>. Cada elemento <*parameter*> é composto pela sequência de elementos: <*name*> e <*description*>. O elemento obrigatório <*name*> é uma cadeia de caracteres que representa o nome do parâmetro de entrada. O elemento opcional <*description*> é uma cadeia de caracteres que possui a descrição do parâmetro de entrada. A estrutura do elemento <*parameter*> é igualmente utilizada nos elementos <*inputParameter*>, <*outputParameter*>, <*humanResourcesProfile*> e <*supportTools*>.

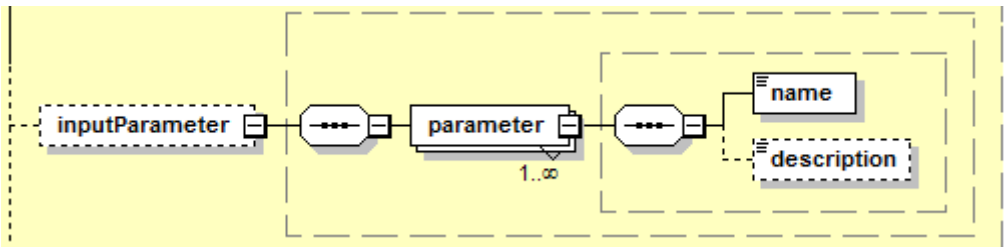

Figura 13: Estrutura do elemento <*inputParameter*>

O elemento opcional <*outputParameter*>, mostrado na Figura 14, é composto por um ou mais elementos <*parameter*>.

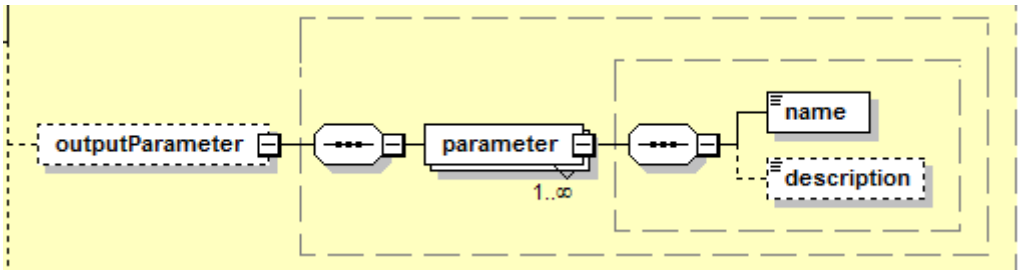

Figura 14: Estrutura do elemento <*outputParameter*>

O elemento opcional <*humanResourcesProfile*>, mostrado na Figura 15, é composto por um ou mais elementos <*parameter*>.

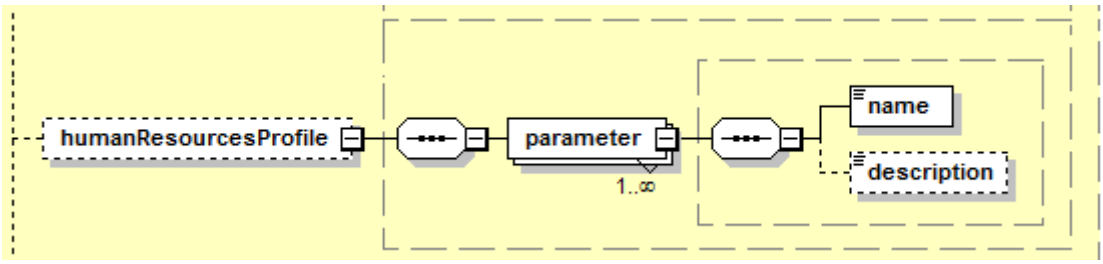

Figura 15: Estrutura do elemento <*humanResourcesProfile*>

O elemento opcional <*supportTools*>, mostrado na Figura 16, é composto por um ou mais elementos <*parameter*>.

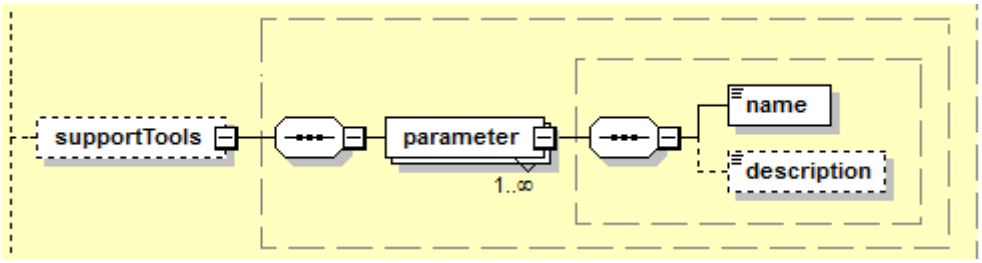

Figura 16: Estrutura do elemento <*supportTools*>

O elemento opcional <*matchingFeatures*>, mostrado na Figura 17 é composto por um ou mais elementos <*parameter*>. A estrutura do elemento <*parameter*> é semelhante aos

anteriores, porém após a seqüência dos elementos <*name*> e <*description*> há um elemento opcional <*featureType*>. O elemento <*featureType*> é composto pela sequência dos elementos <*name*> e <*description*>. O elemento obrigatório <*name*> é uma cadeia de caracteres que representa o nome do tipo da característica e o elemento opcional <*description*> é uma cadeia de caracteres que possui a descrição do tipo da característica. Essa estrutura do elemento <*parameter*> é igualmente utilizada em <*conflictingFeatures*>.

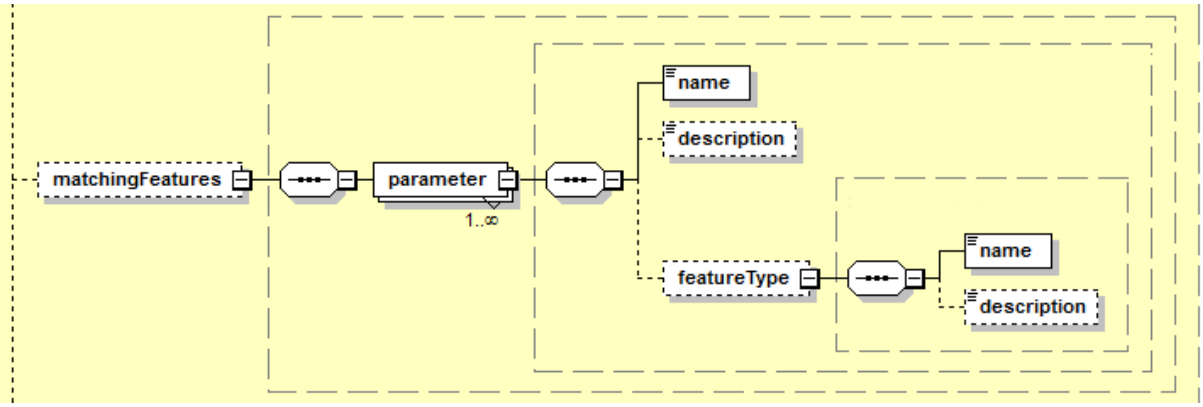

Figura 17: Estrutura do elemento <*matchingFeatures*>

O elemento opcional <*conflictingFeatures*>, mostrado na Figura 18 é composto por um ou mais elementos <*parameter*>.

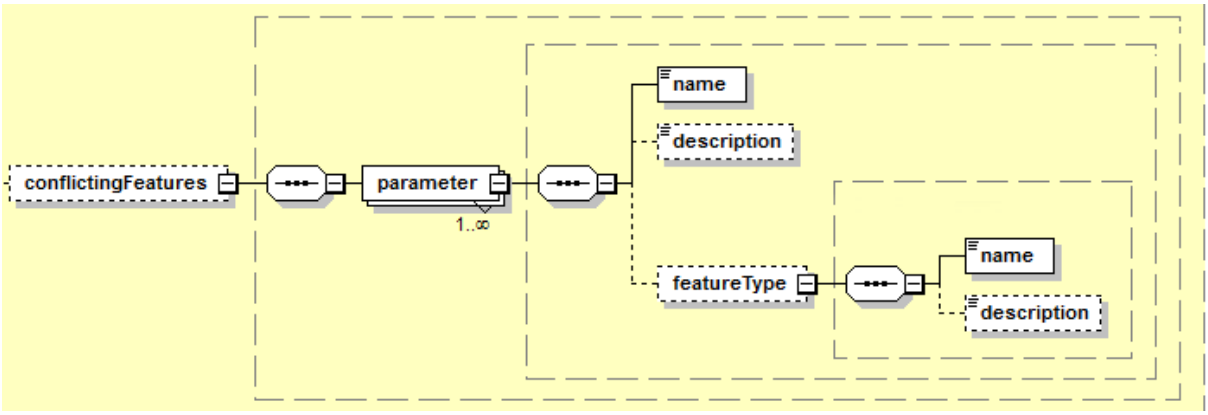

Figura 18: Estrutura do elemento <*conflictingFeatures*>

O elemento opcional <*variations*>, mostrado na Figura 19, é composto por um ou mais elementos <*parameter*>. Cada elemento <*parameter*> é uma cadeia de caracteres que representa o nome do arquivo XML do componente variante.

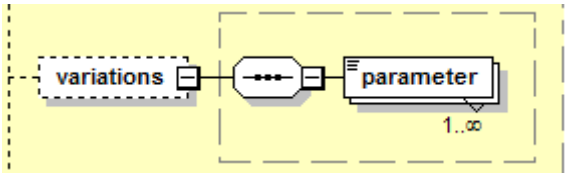

Figura 19: Estrutura do elemento <*variations*>

O elemento opcional <*internalArchitecture*>, mostrado na Figura 20, é composto pela sequência de elementos <*name*>, <*description*> e por um ou mais elementos <*parameter*>. O elemento obrigatório <*name*> é uma cadeia de caracteres que representa o nome da arquitetura interna. O elemento opcional <*description*> é uma cadeia de caracteres que possui a descrição da arquitetura interna. Cada elemento <*parameter*> é composto pela sequência de elementos: <*sourceElement*>, <*connectionKind*> e <*targetElement*>. O elemento obrigatório <*sourceElement*> possui a sequencia de elementos: <*type*>, <*posX*>, <*posY*>, <*condition*>, <*optional*>. O elemento obrigatório <*type*> pode conter um dos seguintes valores: “Componente”, “Atividade”, “ItemInicioArquitetura” ou “ItemFimArquitetura”, que indica qual o tipo do elemento da arquitetura interna representado. Os elementos obrigatórios <*posX*> e <*posY*> possuem a posição do eixo x e eixo y respectivamente da representação gráfica do elemento da arquitetura interna. O elemento opcional <*name*> é uma cadeia de caracteres que representa o nome do elemento da arquitetura interna. O elemento opcional <*optional*> possui o valor “*true*” ou “*false*” indicando se o elemento da arquitetura é opcional ou não. O elemento obrigatório <*connectionKind*> pode conter os valores “FIM_INICIO”, “FIM_FIM”, “INICIO_INICIO”, “INICIO_FIM” e “SIMPLES” que representa o tipo da conexão entre os elementos <*sourceElement*> e <*targetElement*>. O elemento obrigatório <*targetElement*> possui a mesma estrutura do elemento <*sourceElement*>. O elemento <*type*> com o valor “Componente” ou “Atividade” diz se o elemento da arquitetura é um Componente de Processo ou uma Atividade. Atividades são parecidas com Componentes de Processo, porém com algumas diferenças, como discutido no Capítulo 3, por isso possui seu próprio XML Schema.

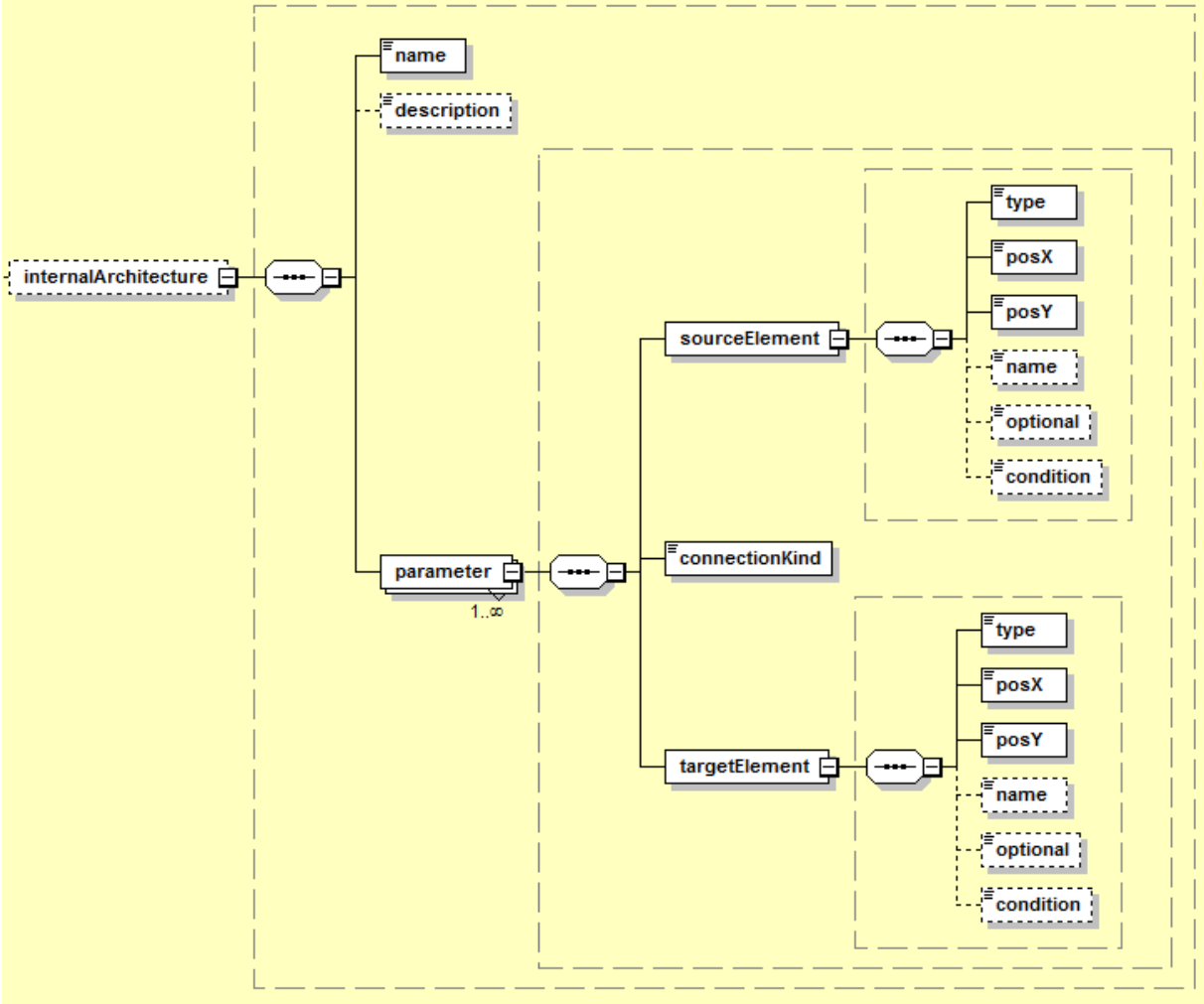

Figura 20: Estrutura do elemento <*internalArchitecture*>

No XML Schema para Atividades, mostrado na Figura 21, o elemento raiz é <*processComponent*> e é definido pela sequência de elementos: <*name*>, <*description*>, <*inputCriterion*>, <*outputCriterion*>, <*responsible*>, <*inputParameter*>, <*outputParameter*>, <*humanResourcesProfile*> e <*supportTools*>. Todos esses elementos são definidos igualmente como os elementos de Componente de Processos. Uma Atividade não pode ser exportada diretamente, isto é, não pode ser selecionada para exportação através da inteface. Atividades são exportadas quando estão dentro de uma arquitetura interna de um Componente de Processo que está sendo exportado, assim quando uma Atividade é encontrada dentro de um componente, o Exportador é chamado para exportar essa Atividade ao invés de um componente. Semelhantemente ocorre com o Importador, que diferencia se está importando um componente ou uma Atividade através do elemento <type> da arquitetura interna. O XML Schema completo para Atividades pode ser obtido no anexo B.

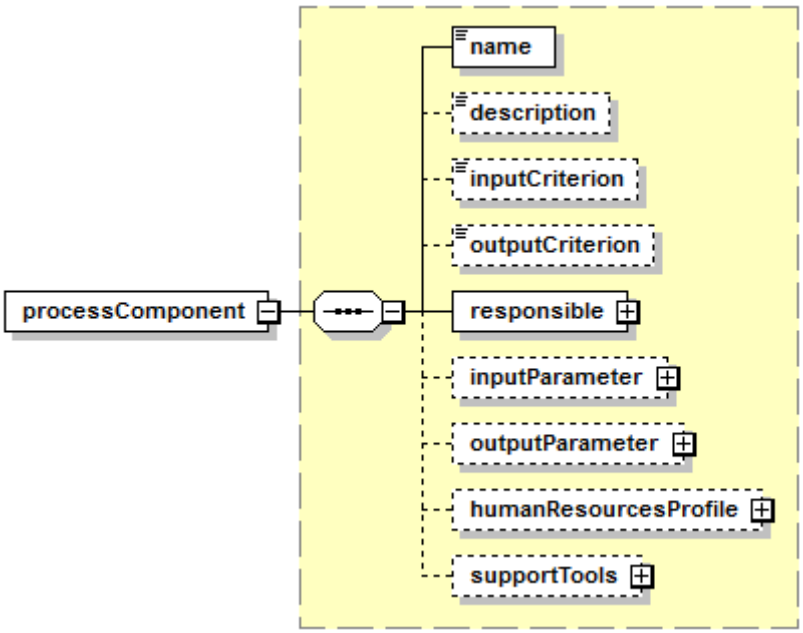

Figura 21: XML Schema para Componentes de Processo de Atividades

Com essas regras de formação estabelecidas no XML Schema dos Componentes de Processo pode-se representar, por exemplo, o seguinte componente:

```
<processComponent>
    <identifier>COP.ABS.0001</identifier>
     <type>Abstrato</type>
    <name>Planejar Riscos Abstrato</name>
    <definingOrganization>
            <name>II-COPPE</name>
            <description>Instituição Implementadora de Processos
            COPPE</description>
            <isImplementingInstitution>true</isImplementingInstitution>
    </definingOrganization>
     <responsible>
            <name>Gerente de Projeto</name>
    </responsible>
    <inputParameter>
          <parameter>
                <name>Plano do Projeto</name>
                <description>Plano que apresenta as diretrizes do
  Projeto</description>
          </parameter>
    </inputParameter>
    <outputParameter>
          <parameter>
                <name>Plano do Projeto</name>
```

```
                <description> Plano que apresenta as diretrizes do
            Projeto</description>
            </parameter>
        </outputParameter>
        <supportTools>
            <parameter>
                <name>MS Word</name>
                <description>Editor de texto da Microsoft</description>
            </parameter>
            <parameter>
                 <name>RiscManager</name>
                <description>Ferramenta para gerência de riscos.</description>
            </parameter>
        </supportTools>
        <variations>
            <parameter>PlanejarRiscos.xml</parameter>
            <parameter>PlanejarRiscosConcreto.xml</parameter>
        </variations>
    </processComponent>
```

Com a finalidade de evitar repetições de conteúdos de elementos, como o elemento <*parameter*> em <*inputParameter*> e <*outputParameter*> no exemplo anterior, economizar espaço e exercitar o reuso, na representação de um Componente de Processo faz-se o uso de Entidades XML, que permite referenciar um conteúdo definido externamente. Assim, cada elemento passível de repetição é definido como uma entidade no cabeçalho do componente e referenciado exatamente onde o elemento deveria estar. O conteúdo referenciado também é um arquivo XML. Atualmente os seguintes elementos utilizam entidades: <*definingOrganization*>, <*inputParameter*>, <*outputParameter*>, <*humanResourcesProfile*>, <*supportTools*>, <*matchingFeatures*>, <*conflictingFeatures*> e <*internalArchitecture*>. O exemplo anterior ficará como a seguir:

```
<!DOCTYPE processComponent SYSTEM "componentType.dtd" [
<!ENTITY definingOrganization SYSTEM "II-COPPE.xml">
<!ENTITY PlanodoProjeto SYSTEM "PlanodoProjeto.xml">
<!ENTITY MSWord SYSTEM "MSWord.xml">
<!ENTITY RiscManager SYSTEM "RiscManager.xml"> ]>

<processComponent>
    <identifier>COP.ABS.0001</identifier>
    <type>Abstrato</type>
    <name>Planejar Riscos Abstrato</name>
    &definingOrganization;
    <responsible>
```

```
            <name>Gerente de Projeto</name>
        </responsible>
        <inputParameter>&PlanodoProjeto;</inputParameter>
        <outputParameter>&PlanodoProjeto;</outputParameter>
        <supportTools>&MSWord;&RiscManager;</supportTools>
        <variations>
            <parameter>PlanejarRiscos.xml</parameter>
            <parameter>PlanejarRiscosConcreto.xml</parameter>
        </variations>
    </processComponent>
```

As entidades definingOrganization, PlanodoProjeto, MSWord e RiscManager são definidas previamente no cabeçalho, dentro da marcação DOCTYPE e referenciadas ao longo da descrição do componente, como por exemplo, o termo "&definingOrganization;" faz referência à entidade "definingOrganization" que por sua vez referência o arquivo externo "II-COPPE.xml", que será como o descrito a seguir:

```
    <definingOrganization>
        <name>II-COPPE</name>
        <description>Instituição Implementadora de Processos</description>
        <isImplementingInstitution>true</isImplementingInstitution>
    </definingOrganization>
```

O mesmo não se aplica ao elemento <*variations*> cuja sequência de elementos <*parameter*> contém o nome do componente XML variante. Os elementos <*parameter*> não utilizam referências uma vez que não se deseja que o conteúdo do componente XML variante seja inserido dentro do elemento <*parameter*>, apenas saber o nome do componente XML variante.

Ainda no cabeçalho dos Componentes de Processo em XML há a definição da versão do XML e da codificação de caracteres utilizada (<?xml version="1.0" encoding="UTF-8"?>) e a definição de um XSLT (<?xml-stylesheet type="text/xsl" href="componentView.xsl"?>) cuja finalidade é ajudar na visualização do Componente de Processo de uma maneira mais amigável através de um browser. Um exemplo dessa visualização pode ser vista na Figura 22. Há a definição de um XSLT para Componentes de Processo e um para Atividades separadamente e ambos são exportados quando necessário pelo Exportador. Quando o componente ou a Atividade XML é aberto com um visualizador compatível, o XSL transforma a árvore de elementos XML em uma tabela em HTML. O XSL completo para

Componentes de Processo pode ser obtido no anexo D. É apresentado a seguir um trecho do código XSLT:

```
<xsl:template match="supportTools">
  <tr style="text-align: left;">
    <td width="35%" class="celula">
      <b>Ferramentas:</b>
    </td>
    <td>
      <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
        <xsl:for-each select="parameter">
          <tr>
            <td width="35%" class="celula">
              <xsl:value-of select="name/ text()"/>
            </td>
            <td width="65%" class="celula">
              <xsl:value-of select="description/ text()"/>
            </td>
          </tr>
        </xsl:for-each>
      </table>
    </td>
  </tr>
</xsl:template>
```

Este trecho é responsável por adicionar uma linha com duas colunas na tabela final gerada pelo XSL, uma à esquerda contendo a palavra “Ferramentas” e na coluna à direita é inserida outra tabela onde o XSL transforma cada <*parameter*> existente de <*supportTools*> em duas linhas, uma com o conteúdo de <*name*> e a outra com o conteúdo de <*description*>.

Identificador:	COP.ABS.0001	
Nome:	Planejar Riscos Abstrato	
Descrição:	-	
Tipo de Componente:	Abstrato	
Responsável:	Gerente de Projeto	-
Artefatos Requeridos:	Plano do Projeto	Plano que apresenta as diretrizes do Projeto
Artefatos Produzidos:	Plano do Projeto	Plano que apresenta as diretrizes do Projeto
Ferramentas:	MS Word	Editor de texto da Microsoft
	RiscManager	Ferramenta para gerência de riscos.
Organização de Definição:	II-COPPE	Instituição Implementadora de Processos COPPE
	-	Instituição Implementadora
É Variante de:	PlanejarRiscos.xml	
	PlanejarRiscosConcreto.xml	

Figura 22: Exemplo de representação de um Componente XML utilizando XSLT

4.3 VALIDADOR

Para verificar que um arquivo representado em XML é valido, ou seja, segue as regras definidas em um XML Schema, utiliza-se um Validador. O Validador recebe dois parâmetros, o arquivo XML e o XML Schema, retornando verdadeiro, caso o arquivo XML seja validado, assegurando que o arquivo XML segue as regras definidas no XML Schema passado como parâmetro, ou retorna falso caso contrário. O algoritmo do Validador monta uma árvore com os elementos do XML passado como parâmetro e verifica sua sintaxe de acordo com o XML Schema passado com parâmetro. Este algoritmo é eficiente na validação, porém apresenta algumas limitações quando o arquivo representado em XML passado como parâmetro faz uso de Entidades XML.

O algoritmo do Validador substitui as referências das entidades externas pelos seus conteúdos correspondentes antes da verificação da validação, porém uma limitação ocorre ao tentar verificar a validade de um componente XML que utiliza Entidades XML. As definições de Entidades XML devem ser feitas dentro da marcação DOCTYPE no cabeçalho do XML. Esta marcação é utilizada para a definição de uma DTD, que é similar ao XML Schema, que serve para definir um conjunto de regras às quais um XML deve seguir para ser válido, porém a DTD é mais antiga e menos robusta que o XML Schema. Devido a uma limitação tecnológica, quando o algoritmo do Validador encontra a declaração DOCTYPE, este espera um arquivo DTD para validar o XML, mesmo que posteriormente seja utilizado um XML Schema para verificar a validade do XML.

Para contornar essa limitação do algoritmo Validador e tornar o uso de Entidades XML possível na representação de Componentes de Processo, uma vez observada os ganhos obtidos com seu uso, a tecnologia DTD é adicionada ao cabeçalho da representação de Componentes de Processo em XML. A DTD utilizada define, basicamente, a mesma estrutura de elementos para Componentes de Processo e Atividades já definidas nos XML Schemas respectivos, porém apresenta definições mais fracas devido às limitações de definições de regras da DTD. Por esse motivo o uso do XML Schema faz-se necessário. Como o uso da DTD foi simplesmente para contornar essa situação, foi definido apenas uma DTD para Componentes de Processo e Atividades, uma vez que sua real validação se dá através de seus respectivos XML Schemas. É apresentado a seguir o trecho onde há a definição da DTD:

```
<!DOCTYPE nomeDocType SYSTEM "caminho.dtd" [
<!ENTITY nomeEntidade1 SYSTEM "caminhoEntidade1.xml">
<!ENTITY nomeEntidade2 SYSTEM "caminhoEntidade2.xml">
...
<!ENTITY nomeEntidadeN SYSTEM "caminhoEntidadeN.xml"> ] >
```

4.4 EXPORTADOR

O Exportador e o Importador de Componentes de Processo são os principais componentes do Process Broker. Eles fazem uso do XML Schema, dos XMLs, do Indexador, do Validador, do Compactador e de dados da base de dados local através da API do ambiente A2M. O Exportador tem a função de extrair os Componentes de Processo da base de dados local e convertê-los em componentes representados em XML, garantindo que estes sejam válidos. O Exportador exporta um documento XML para cada um dos Componentes de Processo selecionados na base de dados local, verificando sua validade. Posteriormente, gera um arquivo de índices com estes componentes exportados e os compacta num arquivo ZIP que poderá ser utilizado como entrada para um Importador de outra base de dados.

A Figura 23 mostra a interface do Exportador. A caixa, à esquerda, possui o Importador (Importação de Componentes de Processo) e o Exportador (Exportação de Componentes de Processo). As caixas acima são ferramentas do ambiente A2M. Na caixa central, há uma lista onde pode-se escolher qual componente exportar. Para exportá-lo, basta selecioná-lo e clicar no botão “Exportar”, o algoritmo do Exportador é responsável por exportar cada componente escolhido e, recursivamente, os componentes que fazem parte deste.

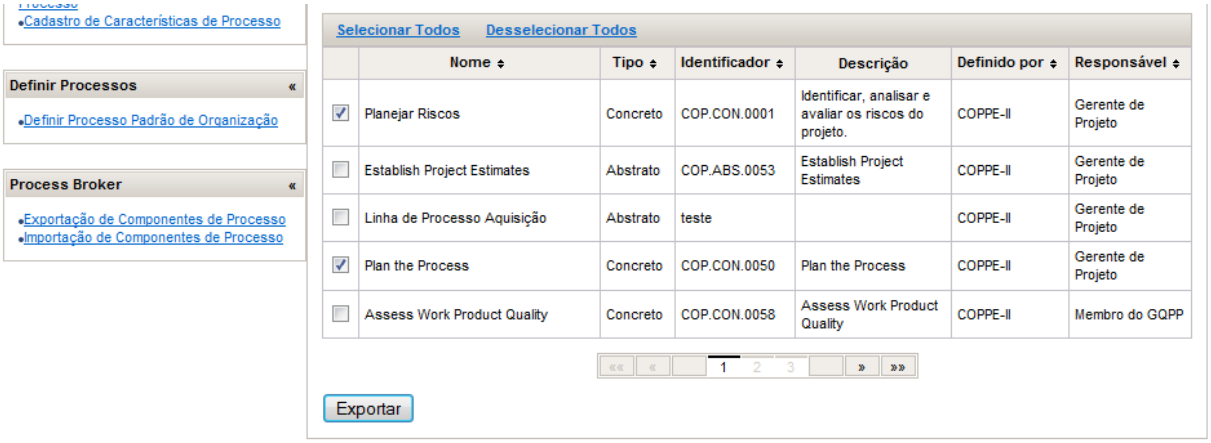

Figura 23: Interface do Exportador – componentes selecionados para exportação.

O algoritmo do Exportador recebe como parâmetro o Componente de Processo selecionado através da interface que deverá ser exportado da base de dados local. Como os Componentes de Processo podem conter em sua definição outros Componentes de Processo, o algoritmo do Exportador pode utilizar recursão para a exportação. Estas definições que necessitam de recursão podem estar nas Variantes e na Arquitetura Interna. Na Arquitetura Interna de um Componente de Processo pode haver Atividades. Estas também serão exportadas pelo Exportador recursivamente.

A partir do Componente de Processo recebido como parâmetro, o algoritmo do Exportador monta uma árvore de elementos XML contendo as informações do componente de processo, seguindo as regras definidas no XML Schema de Componentes de Processo. As informações para a montagem dessa árvore de elementos XML são extraídas da base de dados local usando-se funções disponíveis pelo ambiente A2M. Essas funções retornam as informações necessárias para o algoritmo do Exportador construir cada elemento da árvore. Essa árvore é salva como um arquivo XML e então é verificada a sua validade com o Validador. Caso apresente em sua definição outros componentes ou Atividades estes serão recursivamente exportados e validados. Um exemplo de um componente exportado pelo Exportador pode ser obtido no anexo E.

O algoritmo do Exportador também é responsável por gerar os arquivos XML e montar as referências XML das Entidades XML dos Componentes de Processo exportados. O algoritmo verifica se o arquivo referente à Entidade XML já não foi criado previamente e depois monta a referência para ele. Os arquivos XML são salvos na mesma pasta temporária onde os Componentes de Processo são salvos. Ao termino da exportação de todos Componentes de Processo selecionados, é gerado e validado o arquivo de índices pelo

Indexador e todos os arquivos são compactados em um arquivo ZIP, sendo dada ao usuário a opção de salvá-lo em algum local.

4.5 INDEXADOR

Para auxiliar na importação de Componente de Processo em XML para a base de dados local pelo Importador é gerado pelo Indexador um índice de Componentes de Processo exportados. O índice de componentes também é representado na tecnologia XML e possui um XML Schema que possui as suas regras de formação, sendo assim passível de ser validado pelo algoritmo do Validador. O índice possui uma lista de parâmetros, cada um contendo nome, descrição e nome do arquivo do componente XML exportado. A Figura 24 mostra o diagrama do XML Schema para o índice de componentes.

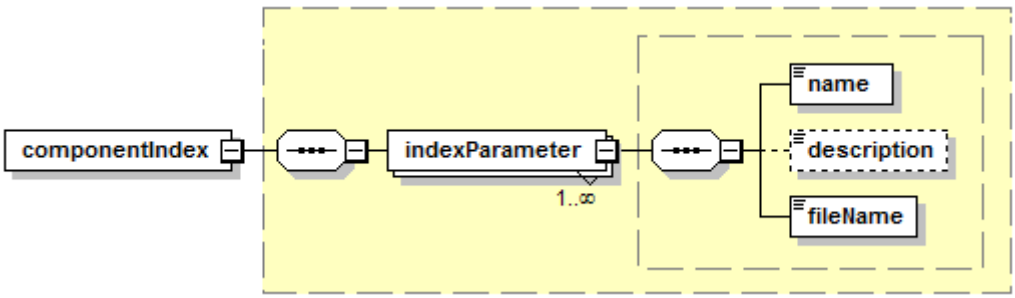

Figura 24: Diagrama do XML Schema para índice de componentes

O XML Schema para o índice de componentes define o elemento raiz <*componentIndex*> como uma sequência de um ou mais elementos <*indexParameter*>. O elemento <*indexParameter*> é definido como uma sequência dos elementos <*name*>, <*description*> e <*fileName*>. O elemento obrigatório <*name*> é uma cadeia de caracteres que representa o nome do componente indexado. O elemento opcional <*description*> é uma cadeia de caracteres que possui a descrição componente indexado. O elemento obrigatório <*fileName*> é uma cadeia de caracteres que representa o nome do arquivo do componente indexado. O XML Schema completo para o índice de componentes pode ser obtido no anexo C. Segue o exemplo com a estrutura de elementos do índice de componentes XML:

```
<componentIndex>
    <indexParameter>
        <name>Planejar Riscos</name>
        <description>Identificar, analisar e avaliar os riscos do
    projeto.</description>
        <fileName>PlanejarRiscos.xml</fileName>
    </indexParameter>
    <indexParameter>
        <name>Realizar Estimativas por Ponto de Função</name>
        <fileName>RealizarEstimativasPorPontoDeFuncao.xml</fileName>
    </indexParameter>
</componentIndex>
```

Para verificar a validade do arquivo de índices, o Exportador ou Importador utiliza o mesmo Validador utilizado para validar Componentes de Processo ou Atividades. Nesse caso, o Validador recebe como parâmetro o arquivo de índices XML e o XML Schema para índices de componentes.

O algoritmo do Indexador recebe como parâmetro cada componente que foi exportado pelo Exportador. Para cada um desses componentes extrai as informações necessárias e monta a árvore do elemento *<indexParameter>* segundo a regra de formação do XML e o coloca na árvore de elementos *<componentIndex>*. Quando não houver mais componentes para serem exportados pelo Exportador, o arquivo de índices XML é gerado, validado e compactado junto com os Componentes de Processo em XML pelo Compactador. No arquivo de índices somente são listados os Componentes de Processo escolhidos para exportação, componentes que forem exportados como variação ou por estarem inclusos em uma arquitetura interna não são incluídos nessa lista. São também excluídos dessa lista Componentes de Processo que foram escolhidos para serem exportados maus que são variantes e/ou estão inclusos em uma arquitetura interna de um componente que já está arquivo de índice. Isso se deve pelo fato de como o Importador utiliza o arquivo de índices XML para fazer a importação.

Na importação de Componentes de Processo em XML, o Importador valida o arquivo de índices e o utiliza para localizar os componentes que podem ser importados. Cada componente da lista do arquivo de índices pode ser escolhido para ser importado ou não. Por esse motivo, componentes que são variantes ou estão inclusos na arquitetura interna de um componente são excluídos do índice, uma vez que obrigatoriamente devem ser importados.

4.6 COMPACTADOR

O Compactador tem como função compactar em um arquivo ZIP os Componentes de Processo em XML exportados pelo Exportador e descompactar um arquivo ZIP contendo Componentes de Processo para serem utilizados pelo Importador.

O algoritmo de compactação recebe dois diretórios como parâmetros e compacta todos os arquivos dentro do primeiro diretório de arquivos e o salva em arquivo ZIP no segundo diretório de arquivos. Esse algoritmo é utilizado após a exportação de todos Componentes de Processo em XML, o índice de componentes XML, DTDs e XSLs.

O algoritmo de descompactação recebe como parâmetro um arquivo ZIP e um diretório de arquivos, todo o conteúdo do arquivo ZIP passado como parâmetro é descompactado no diretório de arquivo passado como parâmetro. Uma limitação desse algoritmo ocorre na nomenclatura dos arquivos, pois não pode conter os caracteres como til e cedilha, por exemplo. Por isso os nomes dos arquivos exportados pelo Exportador são normalizados.

4.7 IMPORTADOR

O Importador tem a função de importar um Componente de Processo em XML para a base de dados local, isto é, uma vez que estes componentes em XML são válidos o Importador extrai as informações de cada componente representado em XML e os insere na base de dados local do ambiente A2M. A Figura 25 mostra a interface do Importador. Nessa interface pode-se escolher o arquivo ZIP para a importação pressionando “Adicionar”. Após o *upload* do mesmo, basta clicar em “Prosseguir com a importação”.

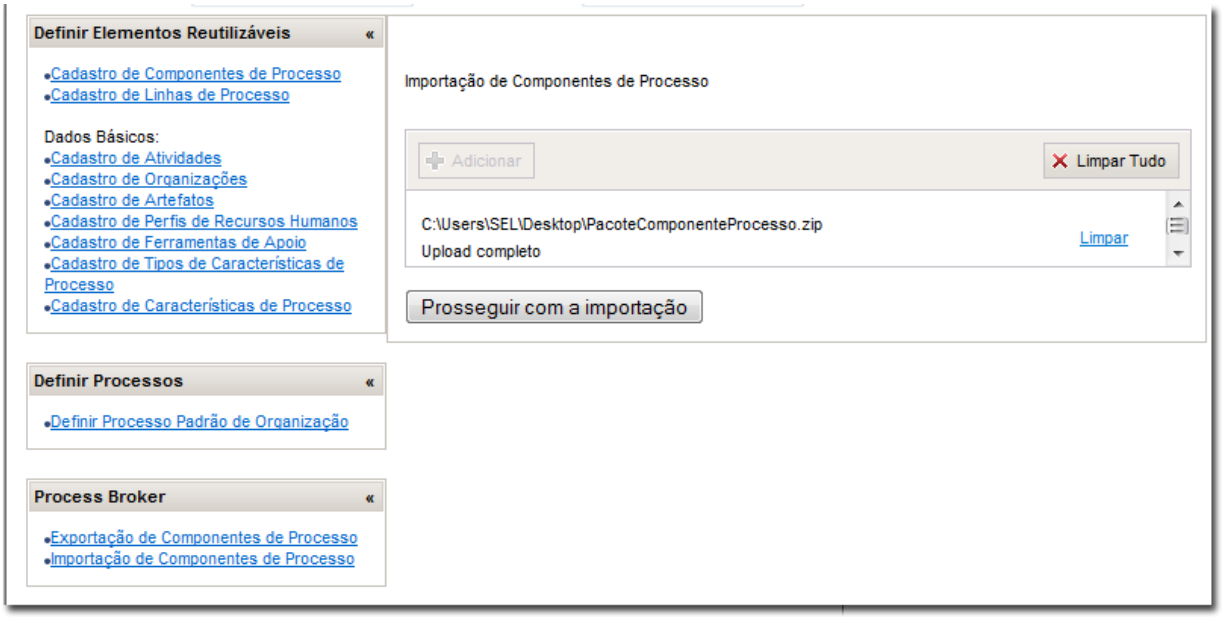

Figura 25: Interface do Importador – arquivo ZIP selecionado para importação.

Internamente, o Compactador descompacta os arquivos do ZIP e o Indexador valida a o arquivo de índices usando o Validador e gera a lista de componentes que poderão ser importados. Em seguida (Figura 26), pode-se escolher os componentes que estão no arquivo ZIP para a importação e, utilizando os botões centrais, pode-se selecionar os componentes, que ficam na caixa à direita. Para importá-los, basta clicar no botão “Importar”. O algoritmo do Importador é então chamado, validando os componentes e os inserindo na base de dados local.

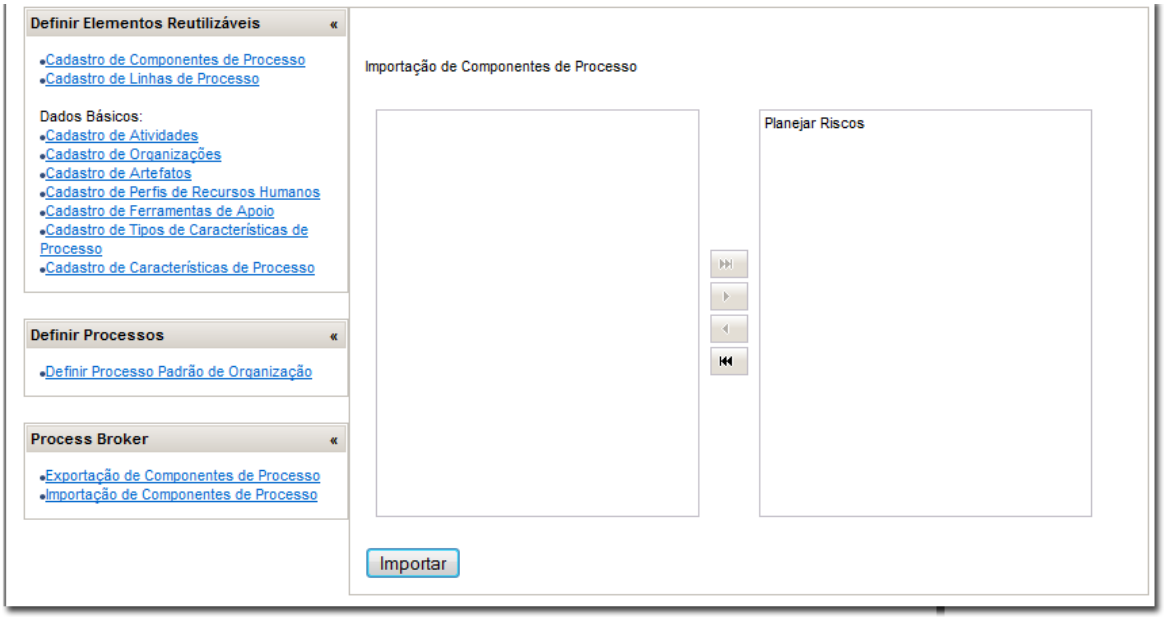

Figura 26: Interface do Importador – seleção dos componentes para importação

O algoritmo do Importador recebe como parâmetro um endereço de um Componente de Processo em XML. O arquivo de índices contém a lista de endereços que serão usados como parâmetros. Este componente é validado com o Validador utilizando o XML Schema de Componentes de Processo. Caso a validação ocorra satisfatoriamente, o algoritmo prossegue com a importação, caso contrário, o algoritmo do Importador interrompe a importação. Similarmente ao Exportador, o Importador monta uma árvore de elementos XML, porém a árvore agora é montada completamente logo nos primeiros passos do algoritmo com todos os elementos do arquivo do Componente de Processo em XML passado como parâmetro. Todas as referências utilizadas nas Entidades XML são substituídas antes da montagem dessa árvore. O algoritmo percorre essa árvore para incluir cada informação na base de dados local, verificando em alguns casos se a informação já não está contida na base de dados. O algoritmo do Importador também pode usar recursão quando o componente que está sendo importado tiver variações em sua definição e/ou uma arquitetura interna.

Quando o Importador entra em uma recursão dentro da arquitetura interna, o Importador pode vir a importar uma Atividade ao invés de um componente. O algoritmo diferencia se está importando um componente ou uma Atividade através do elemento <type> da arquitetura interna, que indica se o próximo elemento a ser importado é um componente ou Atividade.

4.8 DETALHAMENTO DA IMPLEMENTAÇÃO

O Process Broker está integrado ao ambiente A2M, cuja implementação é feita através da linguagem de programação orientada a objetos Java. Por esse motivo o Process Broker também é implementado na linguagem Java. A estrutura do Process Broker apresentada na Figura 7 foi implementada como mostra o diagrama de implantação UML da Figura 27. As relações entre as estruturas serão discutidas a seguir.

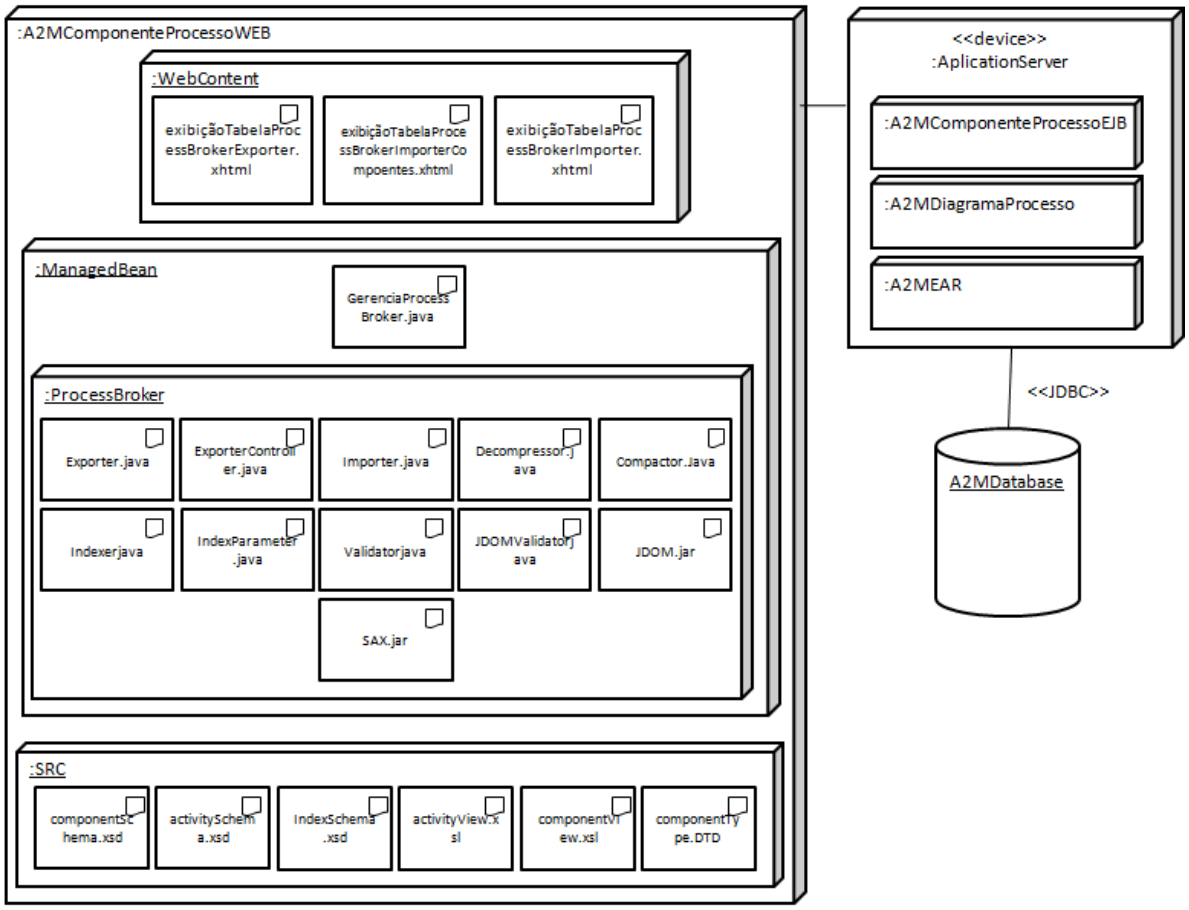

Figura 27: Diagrama de implantação UML do Process Broker

As interfaces do Process Broker, mostradas na Figura 23, Figura 25 e Figura 26, são construídas através dos XHTML *exibicaoTabelaProcessBrokerExporter.xhtml*, *exibicaoTabelaProcessBrokerImporter.xhtml* e *exibicaoTabelaProcessBrokerImporterComponentes.xhtml*, respectivamente. O uso das interfaces do Process Broker, do Exportador, do Importador, do Indexador e do Compactador

são controladas através do ManagedBean *GerenciaProcessBroker.Java*. Esta classe Java é responsável por preencher a tabela com os Componentes de Processo disponíveis na base de dados na interface do Exportador, por preparar os diretórios temporários para receber os arquivos XML, instanciar as classes do Exportador com os componentes selecionados para exportação, instanciar o Importador, passar para ele como parâmetros os componentes XML que serão importados, instanciar o Compactador para compactar ou descompactar o arquivo ZIP e instanciar o Indexador. Esta classe utiliza algumas das ferramentas do ambiente A2M.

O Validador é implementado através de duas classes Java, *JDomValidator.Java* e *Validator.Java.* Essas classes utilizam a biblioteca SAX para verificar a validade do XML com DTD e, assim como as classes do Exportador e Importador, utilizam a biblioteca JDOM para a montagem da árvore XML.

O Compactador é implementado através de duas classes Java, *Compactor.Java*, responsável pela compressão dos arquivos exportados em um arquivo ZIP e *Decompressor.Java,* responsável pela descompressão do arquivo ZIP importado.

O Indexador é implementado através de duas classes Java, *Indexer.Java* e *IndexerParameter.Java.*

O Importador é implementado através de uma classe Java, *Importer.Java*. O Exportador é implementado através de duas classes Java, *Exporter.Java* e *ExporterController.Java. Exporter.Java* é responsável por exportar cada Componente de Processo enquanto o *ExporterController.Java* controla quais componentes, Atividades e entidades foram exportados. Tanto o Exportador quanto o Importador utilizam muitas ferramentas do ambiente A2M, especialmente ferramentas para acessar a base de dados local.

Os XML Schemas *componentSchema.xsd* e *activitySchema.xsd* são utilizados pelo Validador através do Exportador e Importador, já o *indexSchema.xsd* é utilizado pelo Validador através do Indexador.

Os XSLs *componentView.xsl* e *activityView.xsl* e a DTD *componentType.dtd* não são usadas diretamente entretanto são passados pelo Exportador para o diretório temporário para serem exportados e compactados no arquivo ZIP pelo Compactador.

4.9 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os conceitos utilizados no Process Broker e a representação dos Componentes de Processo através da tecnologia XML e XML Schema.

A abordagem apresentada pelo Process Broker utiliza da metodologia de reuso de processos expresso através da troca de Componentes de Processo definidos e estáveis. Essa troca de Componentes de Processo é feita através do Importador e Exportador do Process Broker que deve garantir a integridade dos Componentes de Processo que são expressos através da tecnologia XML.

CAPÍTULO 5 - CONCLUSÃO

5.1 CONTRIBUIÇÕES

Este trabalho apresentou a revisão bibliográfica e toda a abordagem adotada no ambiente Process Broker, que tem por objetivo fornecer um apoio à definição de Processos de Software através da troca de Componentes de Processo de Software reutilizáveis entre as diferentes organizações envolvidas. A seguir, são listadas as principais contribuições deste trabalho:

- Definição um vocabulário através da tecnologia XML para a representação de Componentes de Processo e a definição de um XML Schema para garantir a validade de Componentes de Processo nessa representação;
- Construção do Exportador de Componentes de Processo, responsável pela exportação de componentes da base de dados local na representação XML garantindo que estes sejam válidos;
- Construção do Importador de Componentes de Processo, responsável pela importação de componentes válidos na representação XML para a base de dados local;
- Integração do Process Broker com o ambiente A2M, o qual proporciona uma maior visibilidade para o Process Broker;
- Definição de um XML Schema para o arquivo de índices através da tecnologia XML. Construção do Indexador de Componentes de Processo que gera um

índice dos componentes exportados pelo Exportador e consulta esse índice para a importação de componentes pelo Importador

- Definição da DTD para Componentes de Processo, possibilitando que o Validador valide corretamente os componentes exportados e importados através do XML definido;
- Construção do XSL para Componentes de Processo e Atividades, possibilitando uma visualização mais amigável dos Componentes de Processo representados em XML;
- Construção do Compactador para a compactação dos arquivos exportados pelo Exportador em um arquivo ZIP e descompactação de um arquivo ZIP para o Importador.
- Construção das interfaces de importação e exportação do Process Broker;
- Construção do *ManagedBean* que controla as interações das interfaces do Process Broker com o Exportador, Importador, Indexador e Compactador;

5.2 LIMITAÇÕES

Um dos pontos críticos na elaboração do Process Broker foi a dependência do ambiente A2M. Por um lado foi positivo, pois foram reaproveitadas a interface e alguns métodos implementados, como consultas e persistência na base de dados. Por outro lado era um ambiente que também ainda estava em fase de desenvolvimento, portanto algumas alterações no seu código também impactavam no desenvolvimento do Process Broker. Infelizmente, não houve uma disponibilidade de tempo para que o desenvolvimento de ambas as aplicações acontecessem de forma sincronizada. Portanto, a forma como o Process Broker está implementado destoa um pouco da realidade atual do ambiente A2M. Foi verificado também que possíveis *bugs* no ambiente A2M impactaram na realização de alguns casos de testes, e com isso há a demanda pela realização de mais casos de testes, bem como eventuais correções.

Outra limitação encontrada foi a necessidade da introdução de uma DTD em conjunto com o XML Schema, devido a uma limitação do algoritmo do Validador, uma vez que o uso da DTD não era necessário. Essa limitação ocorreu somente após o inicio da utilização de

Entidades XML na representação de Componentes de Processo em XML. Observados os ganhos obtidos com o uso de Entidades XML e XML Schema, o uso da DTD se fez necessário somente para contornar essa limitação do algoritmo.

5.3 TRABALHOS FUTUROS

Pode-se citar como possíveis trabalhos futuros para o Process Broker:

- Realização de mais testes de Exportação e Importação de Componentes de Processo, com diferentes instâncias do ambiente A2M em diferentes organizações;
- Implementação de um controle de acesso ao Exportador e Importador de acordo com o usuário;
- Implementação de opções avançadas de Exportação e Importação, como por exemplo, a opção de exportar apenas a arquitetura interna de um componente, exportar somente as variações de um componente;
- Implementação de uma ferramenta de edição de Componentes de Processo em XML, para auxiliar na Exportação e Importação;
- Implementação de uma ferramenta para validação de Componentes de Processo em XML, possibilitando que estes componentes possam ser validados sem que sejam importados ou exportados;
- Definição de regras de negócio do compartilhamento de Componentes de Processos de Software disponibilizados;
- Definição de um meio de coleta de métricas relacionadas ao uso dos Componentes de Processo software e ao Process Broker;
- Estudo de possíveis abordagens de implantação do Process Broker, como por exemplo, uma base central de dados com todos Componentes de Processo, que ao ser atualizada com novos componentes, enviaria automaticamente estes componentes para as outras bases de dados.
- Estudo de possíveis aprimorações dos XMLs Schemas definidos.

REFERÊNCIAS BIBLIOGRÁFICAS

ACUÑA, S. T., ANTONIO, A., FERRÉ, X. The Software Process: Modelling, Evaluation and Improvement. In: CHANG, S.K. *Handbook of Software Engineering and Knowledge Engineering*. Singapore: World Scientific Publishing Company, 2000.

BARRETO, Ahilton. *Uma Abordagem para Definição de Processos de Software Baseada em Reutilização*, Exame de Qualificação para o Doutorado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, Brasil, 2007.

BARRETO, Ahilton; MURTA, Leonardo; ROCHA, Ana Regina. *Software Process Definition: a Reuse-based Approach.* In: XXXIV CONFERENCIA LATINOAMERICANA DE INFORMÁTICA (CLEI'08), Santa Fe, Argentina, 2008.

BARRETO, Ahilton; MURTA, Leonardo; ROCHA, Ana Regina. *Componentizando Processos Legados de Software Visando a Reutilização de Processos.* In: SIMPÓSIO BRASILEIRO DE QUALIDADE EM SOFTWARE, 2009.

CAMPOS, F.B., CONTE, T.U., KATSURAYAMA, A.E., *et al. Gerência Quantitativa para o Processo de Desenvolvimento de Requisitos*. In: VI SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 2007, p. 125-139.

CARD, D. N. *The RAD Fad: Is Timing Really Everything?* IEEE Software 12, 5, 1995, p.19–22.

COSTA, Anderson; SALES, Ernani; REIS, Carla A. L.; REIS, Rodrigo Q. *Apoio a Reutilização de Processos de Software através de Templates e Versões.* In: SIMPÓSIO BRASILEIRO DE QUALIDADE EM SOFTWARE, 2007.

CHRISSIS, M. B., KONRAD, M., SHRUM, S. *CMMI: Guidelines for Process Integration and Product Improvement.* 2 ed. Nova York, Estados Unidos: Addison-Wesley, 2006.

ELLMER, E., MERKL, D., QUIRCHMAYR, G., et al. *Process Model Reuse to Promote Organizational Learning in Software Development*. In: ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC), 1996, p. 21-26.

FLORAC, W. A., CARLETON, A .D. *Measuring the Software Process*. Reading, Estados Unidos: Addison-Wesley, 1999.

FLORAC, W.A., PARK, R. E., CARLETON, A.D. *Pratical Software Measurement: Measuring for Process Management and Improvement.* Pittsburg, Estados Unidos, 1997.

FUGGETTA, Alfonso. *Software process: a roadmap*. In: CONFERENCE ON THE FUTURE OF SOFTWARE ENGINEERING. Limerick, Irlanda. Junho, 2000, p. 25-34.

GARG, A., CRITCHLOW, M., CHEN, P., et al. *An Environment for Managing Evolving Product Line Architectures*. In: INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, Amsterdan, Netherlands, 2003, p. 358-367.

GARY, K.A., LINDQUIST, T.E. *Cooperating Process Components*. In: INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC), Phoenix, United States, 1999, p. 218-223.

HOLLENBACH, C., FRAKES, W. *Software Process Reuse in an Industrial Setting*. In: 4TH INTERNATIONAL CONFERENCE ON SOFTWARE REUSE, Orlando, Estados Unidos, 1996, p. 22-30.

HUMPHREY, W.S. *Managing the Software Process*. Boston, Estados Unidos: Addison-Wesley, 1989.

ISO/IEC 15504, *Information Technology – Software Process Assessment*. The International Organization for Standardization and the International Electrotechnical Commission, Parts 1-9, 2003.

ISO/IEC-12207, *Information Technology - Amendment 2 to ISO/IEC 12207*. The International Organization for Standardization and the International Electrotechnical Commission, 2004.

ISO/IEC-24774, *System and software engineering - Life Cycle Management - Guidelines for process definition*. The International Organization for Standardization and the International Electrotechnical Commission, 2006.

MADHAVJI, N.H. The process cycle. *Software Engineering Journal*, 1991, v. 6, n.5, p. 234-242.

MURTA, Leonardo. *Gerência de Configuração no Desenvolvimento Baseado em Componentes*, Tese de D.Sc., Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, Brasil, 2006.

OSTERWEIL, Leon. *Software Process are Software too.* Colorado, Estados Unidos, 1987.

PRESSMAN, Roger S. *Engenharia de Software.* Ed. 6ª, Mc Graw-Hill, 2006.

RU-ZHI, X., HE Tao, *CHU Dong-Sheng, XUE Yun-Jiao, QIAN Le-Qiu. Reuse-Oriented Process Component Representation and Retrieval. International* Conference on Computer and Information Technology. Shangai, China: IEEE Computer Society. September, 2005.

SOFTEX - Associação para Promoção da Excelência do Software Brasileiro. *MPS.BR - Melhoria de Processo do Software Brasileiro, Guia Geral (v1.2)*, 2007.

OSTERWEIL, Leon. *Software Process are Software too.* Colorado, Estados Unidos, 1987.

SOLINGEN, R. V. *Measuring the ROI of Software Process Improvement*. In: IEEE Software, 2004, v.21, n.3, p.32-38.

SPEM - Software Process Engineering Metamodel, Object Management Group, 2006.

WANG, Y., KING, G. *Software Engineering Processes: Principles and Applications*. Estados Unidos: CRC Press, 2000.

WHEELER, D.J., CHAMBERS, D.S. *Understanding Statistical Process Control*. Knoxville, Estados Unidos: SPC Press, 1992.

ANEXOS

ANEXO A - XML SCHEMA PARA COMPONENTES DE PROCESSO

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:complexType name="tDataParameters">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tParameters">
  <xs:sequence>
    <xs:element name="parameter" type="tDataParameters" minOccurs="1"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tVariationParameters">
  <xs:sequence>
    <xs:element name="parameter" type="xs:string" minOccurs="1"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tDataFeatures">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="featureType" type="tDataParameters" minOccurs="0"
maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tFeatures">
  <xs:sequence>
```

```
    <xs:element name="parameter" type="tDataFeatures" minOccurs="1"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="tType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Abstrato"/>
    <xs:enumeration value="Concreto"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="tFlag">
  <xs:restriction base="xs:string">
   <xs:enumeration value="true"/>
   <xs:enumeration value="false"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="tOrganization">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="mission" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="isImplementingInstitution" type="tFlag" minOccurs="1"
maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="tConnectionKind">
  <xs:restriction base="xs:string">
    <xs:enumeration value="FIM_INICIO"/>
    <xs:enumeration value="FIM_FIM"/>
    <xs:enumeration value="INICIO_INICIO"/>
    <xs:enumeration value="INICIO_FIM"/>
    <xs:enumeration value="SIMPLES"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="tElementType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Componente"/>
    <xs:enumeration value="Atividade"/>
```

```
      <xs:enumeration value="ItemInicioArquitetura"/>
      <xs:enumeration value="ItemFimArquitetura"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="tArchitectureElementItem">
  <xs:sequence>
    <xs:element name="type" type="tElementType" minOccurs="1" maxOccurs="1"/>
    <xs:element name="posX" type="xs:float" minOccurs="1" maxOccurs="1"/>
    <xs:element name="posY" type="xs:float" minOccurs="1" maxOccurs="1"/>
    <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="optional" type="tFlag" minOccurs="0" maxOccurs="1"/>
    <xs:element name="condition" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tConnection">
   <xs:sequence>
     <xs:element name="sourceElement" type="tArchitectureElementItem" minOccurs="1"
maxOccurs="1"/>
     <xs:element name="connectionKind" type="tConnectionKind" minOccurs="1"
maxOccurs="1"/>
     <xs:element name="targetElement" type="tArchitectureElementItem" minOccurs="1"
maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tInternalArchitecture">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="parameter" type="tConnection" minOccurs="1"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tProcessComponent">
  <xs:sequence>
    <xs:element name="identifier" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="type" type="tType" minOccurs="1" maxOccurs="1"/>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="inputCriterion" type="xs:string" minOccurs="0" maxOccurs="1" />
```

```
    <xs:element name="outputCriterion" type="xs:string" minOccurs="0" maxOccurs="1" />
    <xs:element name="definingOrganization" type="tOrganization" minOccurs="1"
maxOccurs="1" />
    <xs:element name="responsible" type="tDataParameters" minOccurs="1"
maxOccurs="1" />
    <xs:element name="inputParameter" type="tParameters" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="outputParameter" type="tParameters" minOccurs="0"
maxOccurs="1" />
    <xs:element name="humanResourcesProfile" type="tParameters" minOccurs="0"
maxOccurs="1" />
    <xs:element name="supportTools" type="tParameters" minOccurs="0" maxOccurs="1"/>
    <xs:element name="matchingFeatures" type="tFeatures" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="conflictingFeatures" type="tFeatures" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="variations" type="tVariationParameters" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="internalArchitecture" type="tInternalArchitecture" minOccurs="0"
maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="processComponent" type="tProcessComponent"/>

</xs:schema>
```

ANEXO B - XML SCHEMA PARA ATIVIDADES

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:complexType name="tDataParameters">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tParameters">
```

```
  <xs:sequence>
    <xs:element name="parameter" type="tDataParameters" minOccurs="1"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tVariationParameters">
  <xs:sequence>
    <xs:element name="parameter" type="xs:string" minOccurs="1"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tDataFeatures">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="featureType" type="tDataParameters" minOccurs="0"
maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tFeatures">
  <xs:sequence>
    <xs:element name="parameter" type="tDataFeatures" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="tType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Abstrato"/>
    <xs:enumeration value="Concreto"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="tFlag">
  <xs:restriction base="xs:string">
    <xs:enumeration value="true"/>
    <xs:enumeration value="false"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:complexType name="tProcessComponent">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="inputCriterion" type="xs:string" minOccurs="0" maxOccurs="1" />
    <xs:element name="outputCriterion" type="xs:string" minOccurs="0" maxOccurs="1" />
    <xs:element name="responsible" type="tDataParameters" minOccurs="1"
maxOccurs="1" />
    <xs:element name="inputParameter" type="tParameters" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="outputParameter" type="tParameters" minOccurs="0"
maxOccurs="1" />
    <xs:element name="humanResourcesProfile" type="tParameters" minOccurs="0"
maxOccurs="1"/>
    <xs:element name="supportTools" type="tParameters" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="processComponent" type="tProcessComponent"/>

</xs:schema>
```

ANEXO C - XML SCHEMA PARA ARQUIVO DE ÍNDICES

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:complexType name="tIndexParameters">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="fileName" type="xs:string" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tComponentIndex">
  <xs:sequence>
    <xs:element name="indexParameter" type="tIndexParameters" minOccurs="1"
maxOccurs="unbounded"/>
  </xs:sequence>
```

```
</xs:complexType>

<xs:element name="componentIndex" type="tComponentIndex"/>

</xs:schema>
```

ANEXO D - XSL PARA COMPONENTES DE PROCESSO

```
<?xml version="1.0" encoding="windows-1252"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html"/>

<xsl:template match="/">
  <html>
    <head>
      <style type = "text/css">
        .corpo { font-family:Verdana, Arial, Helvetica, sans-serif;font-size:10px;font-
style:normal; }
        .tabela { font-family:Verdana, Arial, Helvetica, sans-serif;font-size:10px;font-
style:normal;border-right:1px solid #375481;border-bottom:1px solid #375481; }
        .celula { border-left:1px solid #375481;border-top:1px solid
#375481;padding:0.2em;}
      </style>
      <title>"Visualização do Componente de Processo"</title>
    </head>
    <body class="corpo">
      <xsl:apply-templates select="processComponent"/>
   </body>
  </html>
</xsl:template>

<xsl:template match="processComponent">
  <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
    <tr style="text-align: left;">
      <td width="35%" class="celula">
        <b>Identificador:</b>
      </td>
      <td width="65%" class="celula">
        <xsl:value-of select="identifier"/>
```

```
        </td>
      </tr>
      <tr style="text-align: left;">
        <td width="35%" class="celula">
          <b>Nome:</b>
        </td>
        <td width="65%" class="celula">
          <xsl:value-of select="name"/>
        </td>
      </tr>
      <tr style="text-align: left;">
        <td width="35%" class="celula">
          <b>Descrição:</b>
        </td>
        <td width="65%" class="celula">
          <xsl:value-of select="description"/>
        </td>
      </tr>
      <tr style="text-align: left;">
        <td width="35%" class="celula">
          <b>Tipo de Componente:</b>
        </td>
        <td width="65%" class="celula">
          <xsl:value-of select="type"/>
        </td>
      </tr>
      <xsl:apply-templates select="inputCriterion"/>
      <xsl:apply-templates select="outputCriterion"/>
      <xsl:apply-templates select="responsible"/>
      <xsl:apply-templates select="humanResourcesProfile"/>
      <xsl:apply-templates select="inputParameter"/>
      <xsl:apply-templates select="outputParameter"/>
      <xsl:apply-templates select="supportTools"/>
      <xsl:apply-templates select="definingOrganization"/>
      <xsl:apply-templates select="matchingFeatures"/>
      <xsl:apply-templates select="conflictingFeatures"/>
      <xsl:apply-templates select="variations"/>
      <xsl:apply-templates select="internalArchitecture"/>
    </table>
  </xsl:template>

  <xsl:template match="inputCriterion">
    <tr style="text-align: left;">
```

```
      <td width="35%" class="celula">
        <b>Critérios de Entrada:</b>
      </td>
      <td width="65%" class="celula">
        <xsl:value-of select="inputCriterion"/>
      </td>
    </tr>
</xsl:template>

<xsl:template match="outputCriterion">
    <tr style="text-align: left;">
      <td width="35%" class="celula">
        <b>Critérios de Saída:</b>
      </td>
      <td width="65%" class="celula">
        <xsl:value-of select="outputCriterion"/>
      </td>
    </tr>
</xsl:template>

<xsl:template match="internalArchitecture">
    <tr style="text-align: left;">
      <td width="35%" class="celula">
        <b>Arquitetura Interna:</b>
      </td>
      <td>
        <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
          <tr>
            <td width="35%" class="celula">
              <xsl:value-of select="name/ text()"/>
            </td>
            <td width="65%" class="celula">
              <xsl:value-of select="description/ text()"/>
            </td>
          </tr>
        </table>
        <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
          <tr>
            <td width="33%" class="celula" align="center">
              <b>Elemento</b>
            </td>
            <td width="34%" class="celula" align="center">
              <b>Conexão</b>
```

```
            </td>
            <td width="33%" class="celula" align="center">
              <b>Elemento</b>
            </td>
          </tr>
            <xsl:for-each select="parameter">
            <tr>
              <td width="33%" class="celula" align="center">
                <xsl:if test="sourceElement/type/ text() = 'ItemInicioArquitetura'">
                  INICIO<br/>
        <xsl:if test="sourceElement/condition/ text() != 'null'">
                    <xsl:value-of select="sourceElement/condition/ text()"/><br/>
                  </xsl:if>
                </xsl:if>
                <xsl:if test="sourceElement/optional/ text() = 'true'">
                  OPC<br/>
                </xsl:if>
                <xsl:value-of select="sourceElement/name/ text()"/>
              </td>
              <td width="34%" class="celula" align="center">
                <xsl:value-of select="connectionKind/ text()"/>
              </td>
                <td width="33%" class="celula" align="center">
                  <xsl:if test="targetElement/type/ text() = 'ItemFimArquitetura'">
                    FIM<br/>
                    <xsl:if test="targetElement/condition/ text() != 'null'">
                      <xsl:value-of select="targetElement/condition/ text()"/><br/>
                    </xsl:if>
                  </xsl:if>
                  <xsl:if test="targetElement/optional/ text() = 'true'">
                    OPC<br/>
                  </xsl:if>
                  <xsl:value-of select="targetElement/name/ text()"/>
                </td>
              </tr>
            </xsl:for-each>
          </table>
        </td>
      </tr>
    </xsl:template>

    <xsl:template match="responsible">
      <tr style="text-align: left;">
```

```
      <td width="35%" class="celula">
        <b>Responsavél:</b>
      </td>
      <td>
        <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
          <tr>
            <td width="35%" class="celula">
              <xsl:value-of select="name/ text()"/>
            </td>
  <td width="65%" class="celula">
              <xsl:value-of select="description/ text()"/>
            </td>
          </tr>
        </table>
      </td>
  </tr>
</xsl:template>

<xsl:template match="humanResourcesProfile">
  <tr style="text-align: left;">
    <td width="35%" class="celula">
      <b>Participantes:</b>
    </td>
    <td>
      <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
        <xsl:for-each select="parameter">
          <tr>
            <td width="35%" class="celula">
              <xsl:value-of select="name/ text()"/>
            </td>
            <td width="65%" class="celula">
              <xsl:value-of select="description/ text()"/>
            </td>
          </tr>
        </xsl:for-each>
      </table>
    </td>
  </tr>
</xsl:template>

<xsl:template match="inputParameter">
  <tr style="text-align: left;">
    <td width="35%" class="celula">
```

```
        <b>Artefatos Requeridos:</b>
      </td>
      <td>
        <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
          <xsl:for-each select="parameter">
            <tr>
              <td width="35%" class="celula">
                <xsl:value-of select="name/ text()"/>
              </td>
              <td width="65%" class="celula">
                <xsl:value-of select="description/ text()"/>
              </td>
            </tr>
          </xsl:for-each>
        </table>
      </td>
    </tr>
</xsl:template>

<xsl:template match="outputParameter">
  <tr style="text-align: left;">
    <td width="35%" class="celula">
      <b>Artefatos Produzidos:</b>
    </td>
    <td>
      <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
        <xsl:for-each select="parameter">
          <tr>
            <td width="35%" class="celula">
              <xsl:value-of select="name/ text()"/>
            </td>
            <td width="65%" class="celula">
              <xsl:value-of select="description/ text()"/>
            </td>
          </tr>
        </xsl:for-each>
      </table>
    </td>
  </tr>
</xsl:template>

<xsl:template match="supportTools">
  <tr style="text-align: left;">
```

```
    <td width="35%" class="celula">
      <b>Ferramentas:</b>
    </td>
    <td>
      <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
        <xsl:for-each select="parameter">
          <tr>
            <td width="35%" class="celula">
              <xsl:value-of select="name/ text()"/>
            </td>
            <td width="65%" class="celula">
              <xsl:value-of select="description/ text()"/>
            </td>
          </tr>
        </xsl:for-each>
      </table>
    </td>
  </tr>
</xsl:template>

<xsl:template match="definingOrganization">
  <tr style="text-align: left;">
    <td width="35%" class="celula">
      <b>Organização de Definição:</b>
    </td>
    <td>
      <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
        <tr>
          <td width="35%" class="celula">
            <xsl:value-of select="name/ text()"/>
          </td>
          <td width="65%" class="celula">
            <xsl:value-of select="description/ text()"/>
          </td>
        </tr>
        <tr>
          <td width="35%" class="celula">
            <xsl:value-of select="mission/ text()"/>
          </td>
          <td width="65%" class="celula">
            <xsl:choose>
              <xsl:when test="isImplementingInstitution='true'">
                Instituição Implementadora
```

```
                </xsl:when>
                <xsl:otherwise>
                  Instituição Não Implementadora
                </xsl:otherwise>
              </xsl:choose>
            </td>
          </tr>
        </table>
      </td>
    </tr>
</xsl:template>

<xsl:template match="conflictingFeatures">
  <tr style="text-align: left;">
    <td width="35%" class="celula">
      <b>Características Conflitantes:</b>
    </td>
    <td>
      <xsl:for-each select="parameter">
        <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
          <tr>
            <td width="35%" class="celula">
              <xsl:value-of select="name/ text()"/>
            </td>
            <td width="65%" class="celula">
              <xsl:value-of select="description/ text()"/>
            </td>
          </tr>
          <xsl:for-each select="featureType">
            <tr>
              <td width="35%" class="celula">
                <xsl:value-of select="name/ text()"/>
              </td>
              <td width="65%" class="celula">
                <xsl:value-of select="description/ text()"/>
              </td>
            </tr>
          </xsl:for-each>
        </table>
      </xsl:for-each>
    </td>
  </tr>
</xsl:template>
```

```
<xsl:template match="conflictingFeatures">
  <tr style="text-align: left;">
    <td width="35%" class="celula">
      <b>Características Atendidas:</b>
    </td>
    <td>
      <xsl:for-each select="parameter">
        <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
          <tr>
            <td width="35%" class="celula">
              <xsl:value-of select="name/ text()"/>
            </td>
            <td width="65%" class="celula">
              <xsl:value-of select="description/ text()"/>
            </td>
          </tr>
          <xsl:for-each select="featureType">
            <tr>
              <td width="35%" class="celula">
                <xsl:value-of select="name/ text()"/>
              </td>
              <td width="65%" class="celula">
                <xsl:value-of select="description/ text()"/>
              </td>
            </tr>
          </xsl:for-each>
        </table>
      </xsl:for-each>
    </td>
  </tr>
</xsl:template>

<xsl:template match="variations">
  <tr style="text-align: left;">
    <td width="35%" class="celula">
      <b>É Variante de:</b>
    </td>
    <td>
    <table cellpadding="0" cellspacing="0" width="100%" class="tabela">
      <xsl:for-each select="parameter">
        <tr>
          <td width="100%" class="celula">
```

```
            <xsl:value-of select="text()"/>
          </td>
        </tr>
        </xsl:for-each>
      </table>
    </td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

ANEXO E - UM EXEMPLO DE COMPONENTE DE PROCESSO EXPORTADO PELO EXPORTADOR

Ao selecionar o Componente de Processo “Planejar Riscos Abstrato” para exportação através do Exportador, obtemos um arquivo ZIP contendo os seguintes arquivos:

PlanejarRiscosAbstrato.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="componentView.xsl"?>
<!DOCTYPE processComponent SYSTEM "componentType.dtd" [
<!ENTITY definingOrganization  SYSTEM "II-COPPE.xml">
<!ENTITY PlanodoProjeto SYSTEM "PlanodoProjeto.xml">
<!ENTITY MSWord SYSTEM "MSWord.xml">
<!ENTITY RiscManager SYSTEM "RiscManager.xml"> ]>

<processComponent>
<identifier>COP.ABS.0001</identifier>
<type>Abstrato</type>
 <name>Planejar Riscos Abstrato</name>
&definingOrganization;
<responsible>
        <name>Gerente de Projeto</name>
 </responsible>
<inputParameter>&PlanodoProjeto;</inputParameter>
<outputParameter>&PlanodoProjeto;</outputParameter>
<supportTools>&MSWord;&RiscManager;</supportTools>
<variations>
        <parameter>PlanejarRiscos.xml</parameter>
```

```
        <parameter>PlanejarRiscosConcreto.xml</parameter>
 </variations>
</processComponent>
```

II-COPPE.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<definingOrganization>
        <name>II-COPPE</name>
        <description>Instituição Implementadora de Processos COPPE</description>
        <isImplementingInstitution>true</isImplementingInstitution>
</definingOrganization>
```

PlanodoProjeto.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<parameter>
        <name>Plano do Projeto</name>
        <description>Plano que apresenta as diretrizes do Projeto</description>
</parameter>
```

RiscManager.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<parameter>
        <name>RiscManager</name>
        <description>Ferramenta para gerência de riscos.</description>
</parameter>
```

MSWord.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<parameter>
        <name>MS Word</name>
        <description>Editor de texto da Microsoft</description>
</parameter>
```

Por uma questão de espaço, foram suprimidos aqui os seguintes arquivos também exportados pelo Exportador: PlanejarRiscos.xml (variaçao), PlanejarRiscosConcreto.xml (variaçao), componentView.xsl e componentType.dtd.