A Abordagem APSEE para Modelagem e Gerência de Recursos em Ambientes de Processos de Software*

Carla A. Lima Reis[1,2] Rodrigo Quites Reis[1,2] Heribert Schlebbe[3] Daltro J. Nunes[2]
[1] Programa de Pós-Graduação em Computação - Universidade Federal do Rio Grande do Sul
[2] Departamento de Informática - Universidade Federal do Pará
[3]Institut für Informatik - Universität Stuttgart (Alemanha)

Resumo

Este artigo apresenta e discute a modelagem e gerência de recursos de apoio em um ambiente de gerência de processos de software (PSEE). Os PSEEs, tal como os sistemas de Workflow, coordenam tarefas que requerem recursos limitados. Todavia, a maioria dos modelos não permite a especificação precisa desses recursos, o que impede a análise e otimização da sua utilização. A abordagem APSEE é apresentada como contribuição neste tópico, através de um modelo de recursos, da integração do modelo a um PSEE existente e de um mecanismo de auxílio à instanciação de recursos para atividades de um processo. O artigo também discute aspectos da implementação do modelo proposto, além de comparações com trabalhos relacionados.
Palavras-chave: Gerência de processos de software; Gerência de Recursos; PSEE.

Abstract

This paper presents and discusses issues related to resource modeling and management in Process-Centered Software Engineering Environments (PSEEs). PSEEs and Workflow management systems constitute a special kind of software systems developed to coordinate tasks that frequently require limited resources. However, most of the underlying models do not allow precise resource specification, which constitute an obstacle for optimization and analysis of resource allocation. The APSEE approach is presented as a contribution to this field, describing the adopted resource model, the integration of this model to an existing PSEE, and a mechanism to assist resource instantiation for process activities. This paper also discusses some aspects related to the implementation of the proposed model, including comparisons to related work.
Keywords: Software process management; Resource Management, PSEE.

1. Introdução

ADSs Orientados ao Processo ou PSEEs (*Process-Centered Software Engineering Environments)* permitem a modelagem e a execução de processos de desenvolvimento de software [7]. O uso de PSEEs força a definição rigorosa da execução do processo e traz como benefícios a melhor comunicação entre as pessoas envolvidas, a consistência do que está sendo feito, e gerência dos componentes envolvidos na coordenação das atividades.

Os PSEEs, tal como sistemas de *Workflow* coordenam tarefas que necessitam de recursos limitados. A execução de processos depende da disponibilidade desses recursos, e a especificação precisa dos recursos permite a análise e otimização da sua utilização. A maioria dos PSEEs, citados em uma avaliação recente feita por Dermiane et al. [1] não trata em profundidade a definição rigorosa dos recursos necessários para execução das atividades [11][17], o que impede análise e otimização de sua alocação. E ainda, segundo Huff [9], a modelagem e gerência de recursos são pouco desenvolvidas da área de processos de software.

A pesquisa na área de sistemas operacionais tem demonstrado a importância do raciocínio sobre os recursos na paralelização, coordenação e otimização da execução de

* Trabalho apoiado pela CAPES, CNPq e DLR (Cooperação Internacional Brasil-Alemanha).

processos. Alguns atrasos podem ser evitados ou reduzidos através da análise dos recursos direcionada a evitar contenção [11] e manter o estado da realização do processo no mundo real consistente com o modelo sendo executado. Esta consistência entre definição e realização do processo, proposta por Dowson e Fernström [2], pode ser aumentada se o modelo de gerência de recursos for especificado de forma precisa em relação à realidade da organização.

Não existe um consenso na literatura acerca do conceito de recurso. Lonchamp [14] define recurso como sendo qualquer pessoa ou coisa necessária para a realização de um passo do processo. Enquanto Westfechtel [22] define um recurso como qualquer entidade que realiza ou apóia a realização de uma atividade, o que exclui os documentos utilizados. Levando em consideração estas e outras descrições [6], podem-se definir três tipos de recursos diferentes listados na literatura:

- **Agentes ou Recursos humanos**: são as pessoas envolvidas no processo, no desenvolvimento ou na gerência das atividades;
- **Artefatos de Software**: documentos, artefatos de software e qualquer informação necessária para a realização da atividade ou produzida por esta;
- **Recursos de Apoio**: são todos os recursos que apóiam a realização de uma atividade da organização e podem ser compartilhados, como impressoras; consumidos, como recursos financeiros; ou de uso exclusivo como salas e recursos computacionais.

Apesar das definições da literatura levarem a esta classificação genérica de recursos, não é possível tratar todos os tipos da mesma forma. Neste trabalho o termo recurso será utilizado para denotar os **recursos de apoio**, os quais também não podem ser tratados de forma genérica. Não se pode considerar que quaisquer recursos de apoio são alocados e liberados para atividades, pois existem recursos que são totalmente consumidos e deixam de existir após seu uso (como recursos financeiros, por exemplo). Os agentes e os artefatos de software, por terem características e necessidades específicas, merecem um tratamento conceitual e mecanismos de gerência diferenciados, o que está fora do escopo deste artigo.

O raciocínio sobre recursos requer um entendimento das similaridades e diferenças entre os mesmos, assim como as necessidades precisas das atividades a serem coordenadas. Com essa informação é possível analisar e identificar em quais situações apenas um recurso em particular pode ser adequado para a atividade e quais recursos são compatíveis com outro que precisa ser substituído. Além disso, a especificação precisa pode facilitar a decisão sobre quando é necessário acrescentar ou adquirir um determinado recurso para tornar uma atividade mais rápida; ou ainda para gerenciar a alocação de recursos muito usados.

Este artigo propõe a especificação precisa dos recursos como base para as análises indicadas no parágrafo anterior e para permitir a otimização do seu uso. Um modelo de recursos é proposto com o objetivo de definir todos os atributos necessários e todos os relacionamentos envolvidos, assim como os tipos existentes de recursos. É apresentada a integração deste modelo a um PSEE específico chamado APSEE. Porém o meta-modelo pode também ser aplicado a ambientes genéricos de gerência de tarefas. Foi construído também um mecanismo de instanciação de processos de software baseado em critérios definidos pelo usuário. A partir da definição de tipos de recursos necessários a uma atividade e de critérios de restrição e ordenação, são obtidos os recursos disponíveis mais adequados para aquela situação. A definição desses critérios é feita através de Políticas de Instanciação.

O artigo está organizado como segue. A seção 2 apresenta o modelo APSEE. A seção 3 apresenta o modelo de recursos proposto para o ambiente APSEE. A seção 4 apresenta a integração do modelo de recursos no APSEE. A seção 5 apresenta a gerência de recursos durante execução de processos de software. A seção 6 apresenta o auxílio à instanciação de recursos. A seção 7 discute trabalhos relacionados e a seção 8 apresenta as conclusões.

2. Modelo APSEE

APSEE é uma arquitetura que integra todas as fases do ciclo de vida de processos de software, incorporando as tecnologias necessárias para atender cada fase. Existe um meta-modelo único utilizado por todas as ferramentas do ambiente que tratam das fases do ciclo de vida (modelagem, instanciação, execução, etc.). Este meta-modelo, especificado formalmente, é uma evolução das propostas anteriores de apoio ao trabalho cooperativo e gerência de processos de software no ambiente PROSOFT [12]. O paradigma adotado pelo PROSOFT baseia-se na construção de ferramentas através da estratégia *data-driven* [16]. Como o ambiente é desenvolvido dentro de um projeto de cooperação internacional entre Brasil e Alemanha (apoiado por CNPq/DLR), as classes e o protótipo do APSEE (sendo desenvolvido e avaliado internamente) são especificados em inglês. As classes construídas integram os componentes do APSEE e possuem alguns componentes definidos pelos tipos primitivos do PROSOFT (inteiro, real, stringue, longue, texto, booleano).

Um modelo de processos é composto de atividades que podem ser decompostas e conexões entre essas atividades. A linguagem de modelagem de processos de software do APSEE (APSEE-PML) define rigorosamente os componentes necessários para execução de atividades. As atividades são representadas por elipses; as conexões simples por setas grossas (relação de dependência) e múltiplas por setas grossas com a especificação da conexão em um retângulo branco; as conexões de feedback por setas pontilhadas; os artefatos por retângulos 3D acinzentados; e as conexões de artefato por linhas simples. Um exemplo de modelo de processo APSEE é apresentado na figura 1 através de um processo de Análise Estruturada e dois sub-processos decompostos (fragmentos): *Req_Specification* e *SA_Analysis*. A atividade *SA_Analysis* depende do término de *Req_Specification* e de um artefato da anterior.

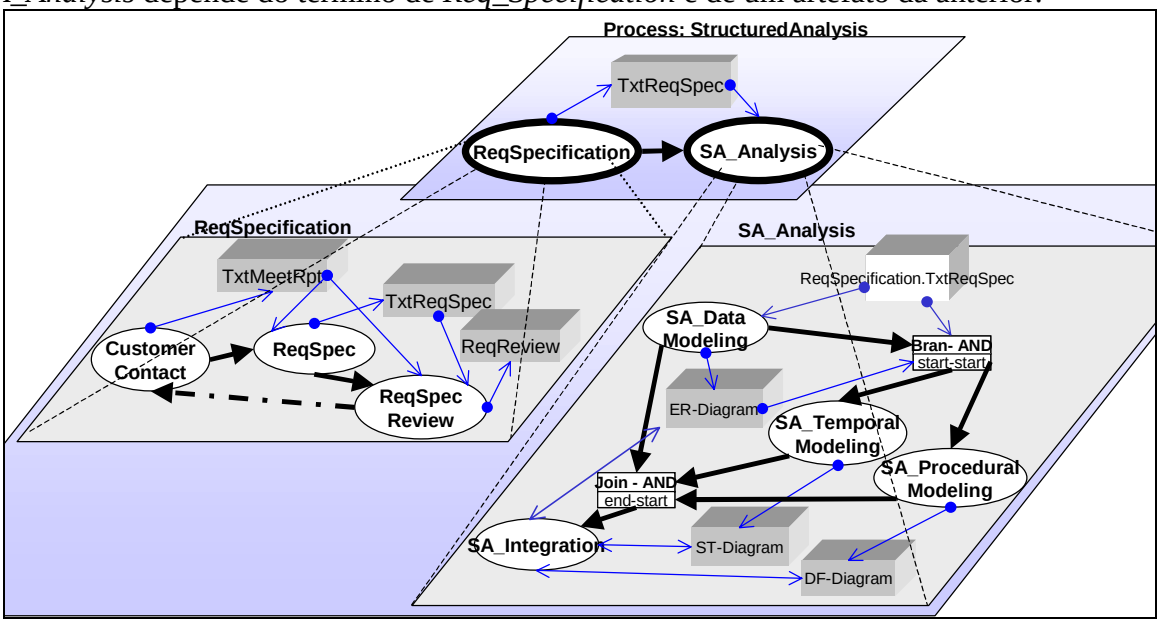

Figura 1. Exemplo de modelo de processo no APSEE.

O meta-modelo proposto para o APSEE é complexo devido à grande quantidade de relacionamentos entre seus componentes. *Apsee* é a classe principal do modelo, apresentada na figura 2 através da composição de vários elementos envolvidos na gerência de processos de software. O componente *ApseeTypes* contém as hierarquias de tipos utilizadas no ambiente, definidas através da classe *Types* (ambas apresentada na figura 2). Novas hierarquias sub-tipos podem ser definidas para especificar o modelo de tipos de recursos proposto no artigo. As seções a seguir mostram o modelo de recursos proposto.

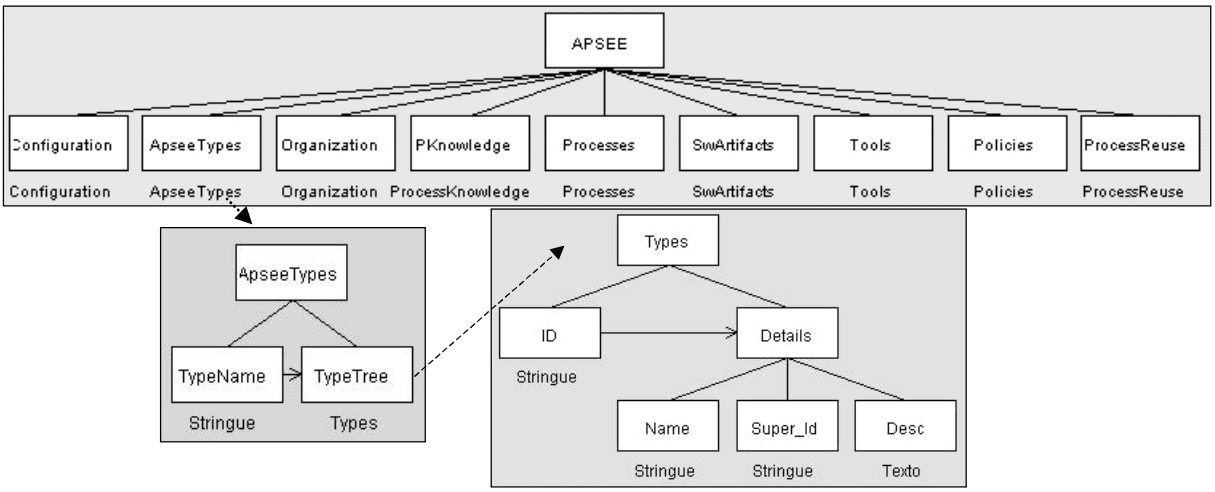

Figura 2. Classes *Apsee*, *ApseeTypes* e *Types*

3. Modelo de Recursos do APSEE

O modelo de recursos é descrito como uma coleção de tipos de recursos, instâncias de recursos e relacionamentos entre os mesmos. Além de prover um mecanismo para descrever e organizar os recursos disponíveis na organização, o modelo de recursos é usado para controlar dinamicamente o acesso concorrente aos recursos, escalonar sua utilização, analisar sua disponibilidade na organização e verificar consistência de seu uso.

O modelo de recursos é representado conforme a Figura 3 (notação UML). A hierarquia de tipos está representada pelo relacionamento *Has sub-type* da classe *Resource Type*. O relacionamento entre um recurso e seu tipo é representado pela associação *Has Type* e a conexão entre os recursos é definida pelas associações *Belongs_to* (um recurso pode ser componente de outro) e *Requires* (um recurso pode requerer outros recursos para ser alocado).

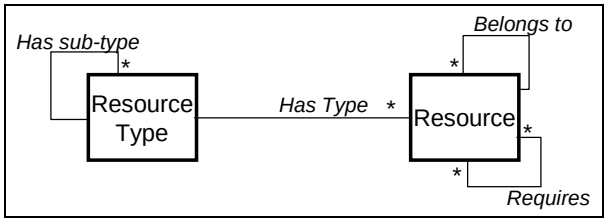

Figura 3. Modelo de Recursos em alto nível.

Para a modelagem dos tipos de recursos é utilizado o componente *ApseeTypes* apresentado na seção anterior. Uma hierarquia inicial de tipos de recursos foi incluída no meta-modelo, permitindo que novos tipos e subtipos sejam acrescentados dependendo das necessidades de recursos existentes. Para a modelagem das instâncias dos recursos é utilizado o componente *Resources* do meta-modelo APSEE, o qual permite a definição de vários relacionamentos entre recursos, além dos atributos específicos dos mesmos. As subseções a seguir detalham os componentes do modelo de recursos.

3.1 Hierarquia de Tipos de Recursos

Utilizando o componente *ApseeTypes* foi definida uma hierarquia de tipos de recursos cuja raiz é um tipo predefinido chamado *Resources*. Os atributos de cada tipo de recurso não são definidos na hierarquia de tipos e sim no modelo das instâncias dos recursos (apresentado na próxima seção). Com a definição de alguns tipos de recursos é possível obter a hierarquia da Figura 4. Os subtipos principais de recursos são:

- ***Exclusive*** (Exclusivos): São recursos que não podem ser alocados para várias atividades

simultaneamente. Salas e computadores são exemplos deste tipo;

- ***Shareable*** (Compartilháveis): Podem ser utilizados por várias atividades simultaneamente sem necessidade de alocação e liberação do recurso. Assim como os recursos exclusivos, os compartilháveis podem ficar indisponíveis em algumas ocasiões (defeito, por exemplo), impedindo a realização de uma atividade. Impressoras são exemplos deste tipo;
- ***Consumable*** (Consumíveis): Não podem ser utilizados novamente como os outros. Uma instância de recurso consumível pode ser utilizada total ou parcialmente por uma atividade. Dinheiro e papel são exemplos desse tipo.

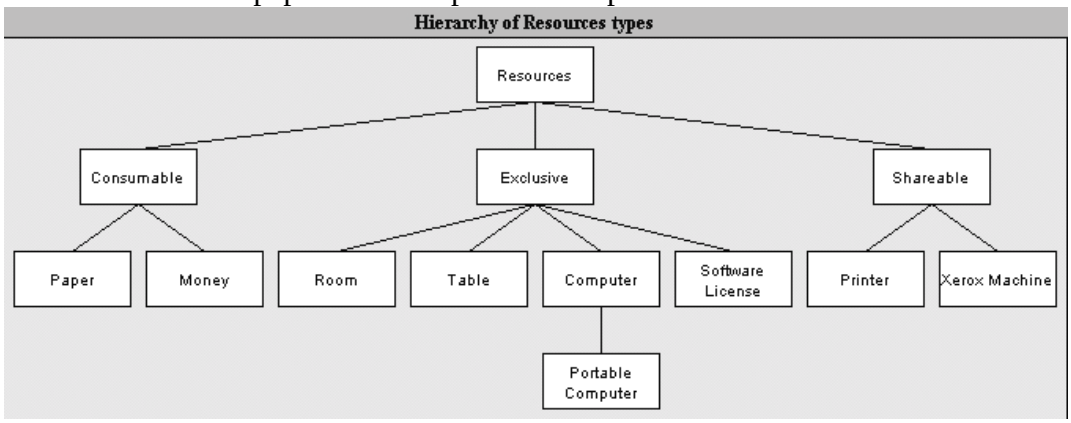

Figura 4. Exemplo de Hierarquia de tipos de recursos no APSEE.

3.2 Instâncias de Recursos

Cada recurso possui um tipo da hierarquia de tipos previamente definida. O componente que armazena os atributos comuns a todos os tipos de recursos da organização é *Resources*, cuja classe é apresentada na Figura 5 seguida da descrição dos atributos.

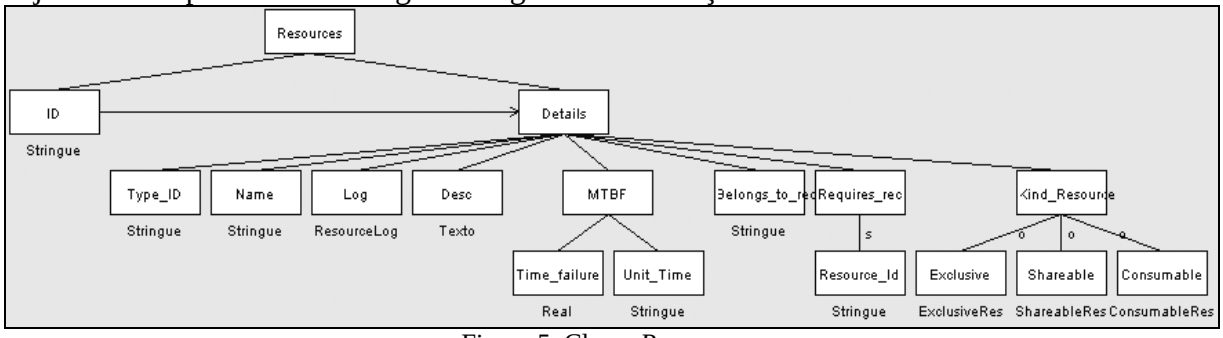

Figura 5. Classe *Resources*.

- Um **identificador** (*ID*) único no ambiente; Um **tipo** de recurso (*Type_ID*) que referencia um elemento da hierarquia de tipos; Um **nome** (*name*) e uma **descrição** textual (*desc*);
- Uma **lista de ocorrências** (*Log*), com as modificações feitas no recurso, indicando data e hora, evento (por exemplo, mudança de estado do recurso), nome do agente, identificador do processo e da atividade que o utilizou. Através do *log* é possível saber a taxa de utilização de um recurso, por exemplo. Essa informação pode ser visualizada por usuários do ambiente, porém apenas gerentes de processo (ou cargos equivalentes) podem manipular o estado de um recurso. A inclusão de novas ocorrências (p.ex., liberação de um recurso) é feita internamente pela máquina de execução de processos do APSEE.
- O **tempo médio entre falhas** (*MTBF)* armazena o tempo e a unidade de tempo entre falhas (ex.: 2 meses), podendo ser obtido a partir do *log* e permitindo que o projetista decida pelo recurso mais adequado em atividades críticas do processo, por exemplo;

- Relacionamento de **composição todo-parte** (atributo *belongs_to)*, que faz referência ao identificador de outro recurso. Este relacionamento pode alterar a disponibilidade e alocação dos recursos. Se uma sala composta de três computadores é alocada para uma atividade, então seus componentes também devem ser alocados. Caso apenas um dos computadores estiver alocado, isto não implica que a sala também deva estar;
- Relacionamento **requer** (atributo *requires)*, que contém um conjunto de identificadores de recursos requeridos para alocação. Este relacionamento também modifica a disponibilidade e alocação de recursos. Porém, neste caso, o fato de um recurso requerer outros não significa que os têm como componentes. Uma impressora pode requerer papel enquanto uma licença de software pode requerer um computador configurado para executar o software. Quando um recurso é alocado, são também alocados todos os recursos necessários especificados neste relacionamento.

Dependendo do tipo definido no atributo *Type_ID*, alguns atributos são acrescentados. Na classe *Resources* o atributo *Kind_resource* seleciona entre as alternativas *Exclusive*, *Shareable* e *Consumable*. O estado, custo e a unidade de custo do recurso são atributos com semântica diferente dependendo do tipo de recurso, portanto são definidos como atributos específicos. As características adicionais são detalhadas a seguir.

3.3 Recursos de uso Exclusivo (*Exclusive Resources*)

Possuem como características adicionais o conteúdo da classe *ExclusiveRes* mostrada na Figura 6, seguida da descrição dos seus componentes.

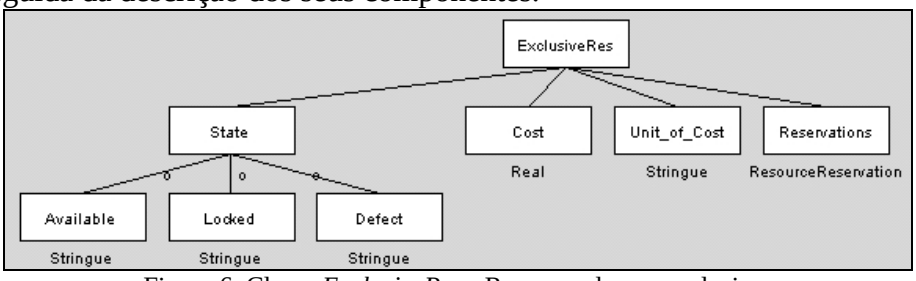

Figura 6. Classe *ExclusiveRes* - Recursos de uso exclusivo.

- **Estado** do recurso (*State*), que pode ser disponível (*available*), alocado (*Locked*) ou com defeito (*defect*). A alocação e liberação de um recurso são realizadas automaticamente pelo ambiente. A mudança para o estado *Defect* somente pode ser realizada por um usuário que gerencie o ambiente, tal como mostrado na Figura 7;

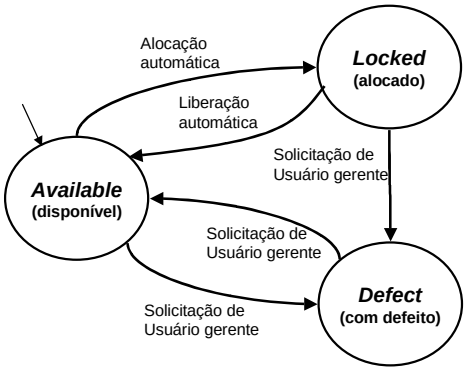

Figura 7. Estados de recursos de uso exclusivo.

- **Custo** e **Unidade de tempo do custo** (*Cost* e *Unit_Cost*), que definem o custo de utilização do recurso em função do tempo. Por exemplo, o custo de um equipamento é definido como R$100,00 por hora de utilização (*Cost* = 100,00 e *Unit_Cost* = "hora");
- **Reservas** de um recurso (*Reservations*), definidas pela classe *ResourceReservations* que define para qual processo, atividade e período o recurso está reservado. As reservas servem para planejamento de uso futuro de recursos e têm o potencial de melhorar a sua

utilização por garantir sua disponibilidade no momento da alocação.

3.4 Recursos Compartilháveis (*Shareable Resources*)

Possuem os atributos da classe mostrada na Figura 8 cujos componentes são:

- **Estado** do recurso (*State*), que pode ser disponível (*available*) ou não-disponível (*Not_Available*). Como este tipo de recurso não é alocado e liberado para atividades específicas, o seu estado permanece sempre disponível até que um usuário gerente o torne não disponível (podendo retornar para o estado anterior do mesmo modo). Entretanto, o *log* armazena a utilização do recurso pelas atividades e demais componentes do ambiente;
- **Custo** e **Unidade de tempo do custo** (*Cost* e *Unit_Cost*), definem o custo de utilização do recurso em função do tempo assim como os recursos de uso exclusivo da seção 3.3;

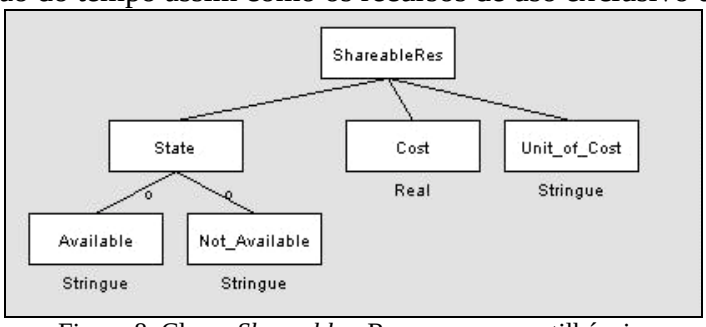

Figura 8. Classe *Shareable* - Recursos compartilháveis.

3.5 Recursos Consumíveis (*Consumable Resources*)

Possuem os atributos da classe da Figura 9 seguida da descrição de seus componentes.

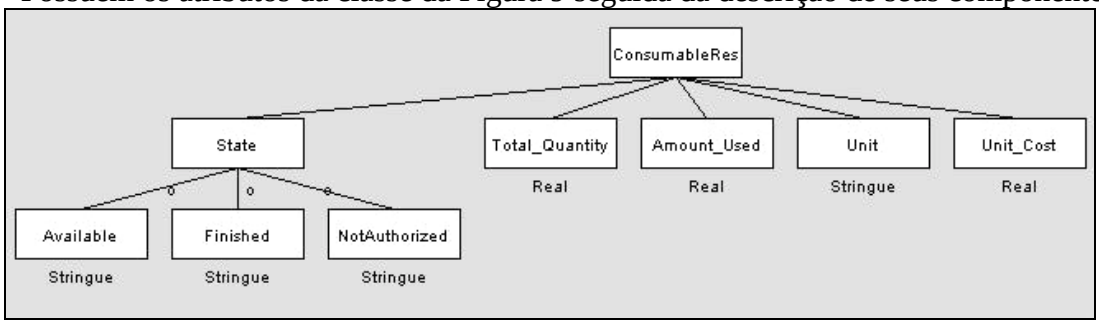

Figura 9. Classe *ConsumableRes* - Recursos consumíveis.

- **Estado** do recurso (*State*), que pode ser disponível (*available*), terminado (*finished*) ou não autorizado (*Not_Authorized*). Não existe alocação e liberação deste tipo de recurso, e sim uso ou consumo. Duas ou mais atividades podem estar usando o mesmo recurso consumível, desde que não ultrapassem o uso da quantidade total disponível do mesmo e desde que esteja disponível. Os estados *available* e *finished* são tratados automaticamente pelo ambiente. O estado *Not_Authorized* pode ser manipulado pelo usuário gerente;
- **Quantidade Total** (*TotalQuantity*) e **usada** (*amount used*) do recurso. O uso deste tipo de recurso é registrado através de adição no atributo *amount_used*;
- O atributo ***Unit*** guarda a unidade de medida do recurso. Por exemplo, para um papel, a unidade pode ser resmas ou folhas. E o atributo ***Unit_cost*** guarda o custo unitário do recurso (útil para recursos não financeiros).

Semelhante aos recursos *Shareable*, este tipo de recurso não requer alocação exclusiva, porém o *log* registra a sua utilização.

4. Integração com o Modelo de Processo de Software

Como apresentado na seção 2, o APSEE integra vários componentes relacionados à gerência de processos e às informações da organização que o utiliza. O componente *Processes* integrado ao APSEE contém todos os processos de software descritos em diferentes estados de modelagem e de execução. Atividades e conexões entre atividades são componentes de um processo de software. As próximas seções apresentam as atividades que compõem um processo no APSEE e os requisitos de recursos para atividades.

4.1 Atividades

A Figura 10 mostra a classe *Activities*, onde cada atividade possui um identificador e detalhes, dentre eles a descrição *Act_description* mostrada na mesma figura. Uma atividade pode ser simples (*plain*) ou fragmento (*fragment*). Se for fragmento, então *ActDescription* refere-se a um novo modelo de processo composto de novas atividades e conexões. Caso contrário, trata-se de uma atividade folha na decomposição do modelo de processo descrita através de seus requisitos (texto), uma descrição detalhada e informações de sua execução. Na descrição detalhada, a atividade pode ser definida como normal ou automática. Atividades automáticas não consomem recursos nem tempo e são realizadas através de chamadas a ferramentas integradas no ambiente.

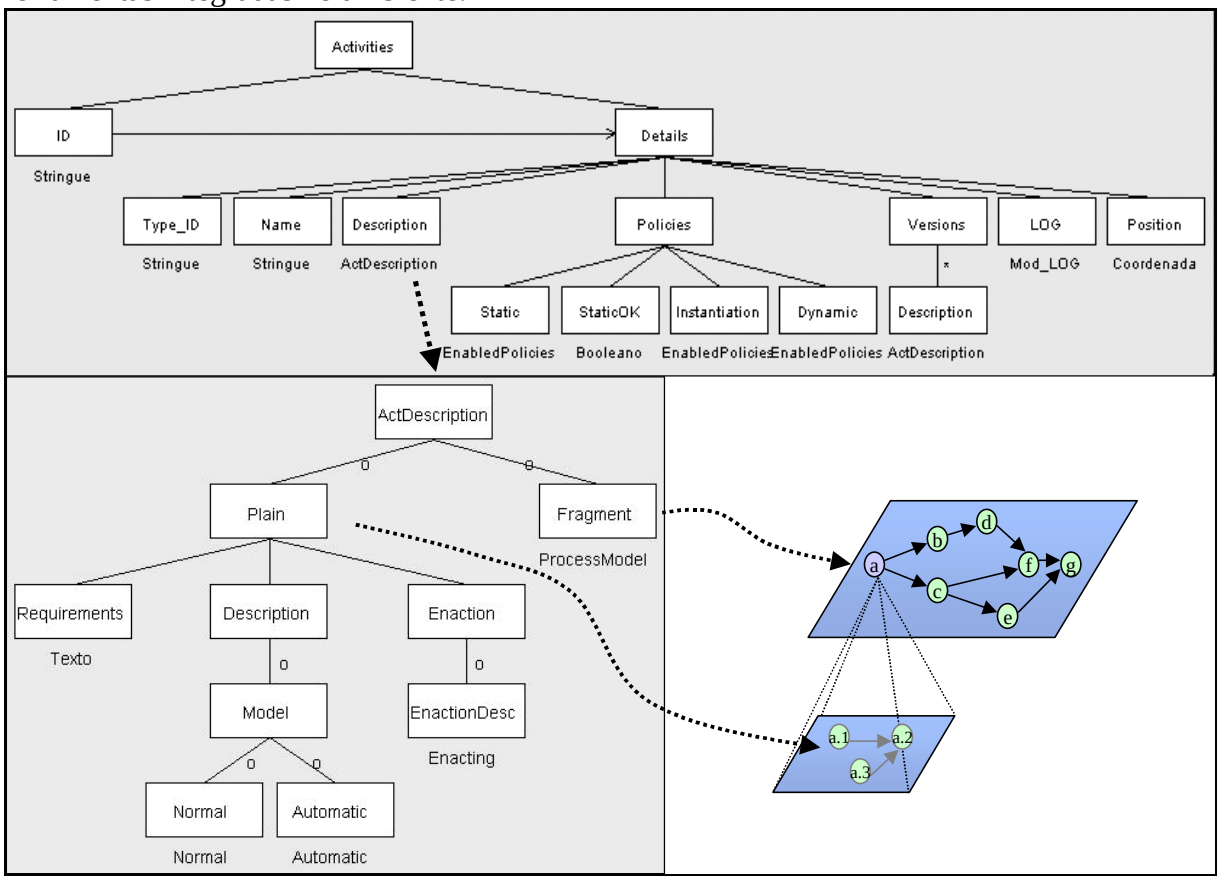

Figura 10. Classe *Activities* e *Act_Description*

4.2 Recursos de uma atividade

A classe que referencia recursos dentro de um processo de software é a classe *Normal* mostrada na Figura 11. Esta classe define detalhes de atividades realizadas por pessoas e

relaciona informações abstratas (tipos dos componentes) e instanciadas (os identificadores dos componentes). Além disso, pode-se observar todo o detalhamento dos requisitos de uma atividade para ser executada. O componente *People* define que agentes ou grupos trabalharão na atividade, *Resources* define os recursos necessários (a ser detalhado a seguir), *Artifacts* define artefatos de entrada e saída, *Schedule* contém o cronograma abstrato (duração em dias – *how long*) e concreto (*begin*, *end* – datas de início e fim previstas), *Script* permite definir os objetivos da atividade e *Conditions* com pré e pós condições para a sua execução.

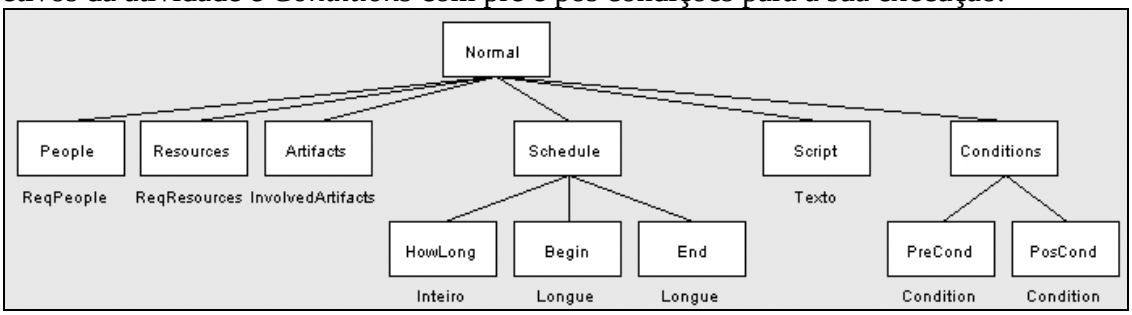

Figura 11. Classe *Normal* – com detalhes de uma atividade do processo.

O componente *Resources* é definido pela classe *ReqResources* (Recursos requeridos) mostrada na Figura 12. Esses recursos são definidos em uma lista onde cada elemento deve distinguir pelo menos o tipo do recurso e o identificador do recurso requerido (instância de recurso que referencia a classe Resources da Figura 5). Cada recurso requerido pode ser um consumível ou um recurso de outro tipo. Ambos têm atributos em comum, com exceção do atributo *AmountNeeded* dos consumíveis, que registra a quantidade a ser usada do recurso.

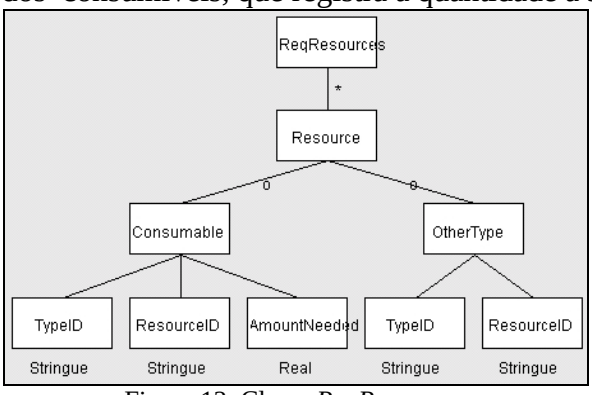

Figura 12. Classe *ReqResources*

O armazenamento dos tipos necessários na descrição da atividade é feito para aumentar a flexibilidade da definição do processo e permitir que o mesmo possa ser reutilizado em outro contexto, ou mesmo em outra organização [18]. Todos os componentes do ambiente são associados a um item da hierarquia de tipos do ambiente, o que permite adiar a decisão do projetista de processo sobre qual recurso vai ser efetivamente utilizado para o início da execução da atividade. Esta decisão será auxiliada pelo mecanismo de instanciação de modelos de processos, o qual a partir dos tipos necessários e de critérios definidos pelo usuário, sugere os recursos que devem ser alocados. Este mecanismo de instanciação será apresentado na seção 6.

5. Gerência de Recursos durante Execução de Processos de Software

O mecanismo de execução de processos do APSEE controla os elementos envolvidos com a realização das atividades. Para que uma atividade seja executada, sua descrição deve estar instanciada, isto é, todos os componentes (instâncias de tipos APSEE) necessários à sua

realização devem estar identificados (os tipos necessários não são obrigatórios).

Nesta seção será apresentada a gerência da execução de processo do ponto de vista de uma única atividade e seus recursos requeridos (A semântica completa de execução de uma atividade envolve também outros componentes do ambiente.). Um dos componentes da atividade mostrada anteriormente na Figura 10 é a descrição da sua execução (*EnactionDesc*). Quando um fragmento de processo é executado esta descrição é criada. O início da execução das atividades aguarda até que suas antecessoras concluam (estado *Waiting*). Quando isso ocorre, a atividade está pronta para começar (estado *Ready*), tendo que esperar manifestação dos agentes responsáveis para ser considerada ativa (estado *Active*).

Na ativação de uma atividade os recursos necessários devem ser alocados ou consumidos. No caso de indisponibilidade de um recurso, a atividade ficará bloqueada até que os recursos estejam disponíveis. A tabela 1 apresenta as operações de gerência de recursos que o mecanismo de execução do ambiente necessita. A Figura 13 descreve os estados de uma atividade em execução e o seu relacionamento com a gerência de recursos.

Tabela 1: Operações de Gerência de Recursos

Operações / Recursos	Para uso do recurso	Para liberação do recurso
Exclusivos	**EUso–Alocar_Recurso**: Se o recurso está disponível, muda estado para *locked* e registra início do uso no log.	**ELib-Liberar_Recurso**: Muda estado para *available* e registra fim do uso no log.
Compartilháveis	**SUso-Inicia_uso**: Se o recurso está disponível registra uso do recurso no log.	**SLib-Termina_Uso**: Registra o fim do uso do recurso no log.
Consumíveis	**CUso-Consome_Recurso**: Se recurso está disponível e é suficiente, aumenta *amount_used* com *amount_needed*. Registra consumo no log.	**CLib–Registra_fim_consumo**: Registra fim de consumo do recurso pela atividade no log.

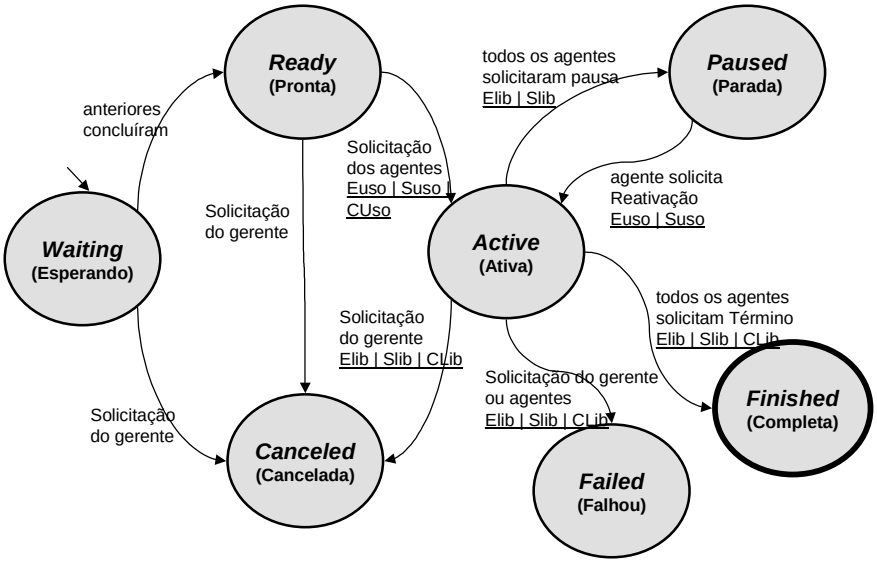

Figura 13. Transição de estados de uma atividade durante execução.

6. Auxílio à Instanciação de Recursos

A integração do modelo de recursos ao modelo de processos de software permite a instanciação do modelo de processos. A partir da definição dos tipos de recursos requeridos para uma atividade (parte abstrata da atividade) é sugerido, a pedido do usuário (projetista de processo), o recurso ou uma lista de recursos mais adequados. Esta sugestão leva em consideração o tipo do recurso, sua disponibilidade para o período da atividade, a disponibilidade dos recursos requeridos e dos recursos componentes. O usuário pode definir

critérios de restrição e ordenação na instanciação de recursos, como por exemplo, o critério de escolha dos recursos com menor custo.

6.1 Quando pode ocorrer instanciação

A Figura 14 apresenta o relacionamento entre os níveis abstrato e instanciado na definição de processos. No nível abstrato estão definidas referências aos tipos de recursos requeridos por uma atividade, enquanto que no nível instanciado podem ser definidas instâncias da organização. Na mesma atividade alguns elementos podem estar definidos nos dois níveis ou em um nível somente, não havendo restrição quanto a componentes já instanciados pelo usuário. O projetista pode solicitar auxílio à instanciação de recursos nos seguintes momentos:

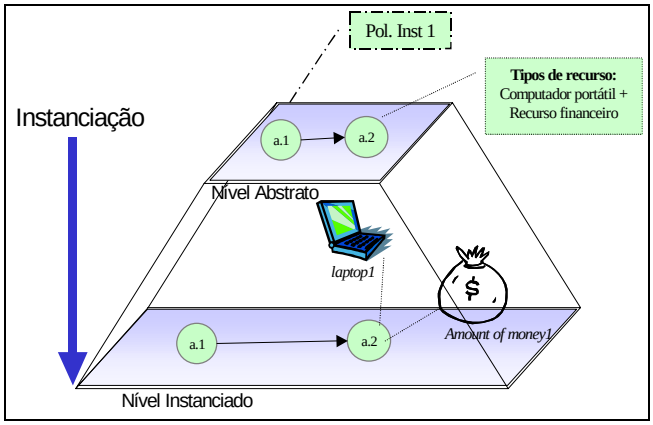

Figura 14. Instanciação de um fragmento de processo de software

- **Durante a modelagem da atividade** (antes de submeter o processo à execução): O mecanismo de instanciação verifica para cada tipo de recurso requerido se existem instâncias disponíveis. Os recursos exclusivos podem ser reservados para o período da atividade, garantindo sua disponibilidade no momento da execução. Os consumíveis devem ter uma quantidade suficiente para uso pela atividade. Enquanto que os compartilháveis não devem estar no estado *Defect*. Além disso, caso o recurso requeira ou seja composto de outros recursos, estes devem ser verificados quanto à possível disponibilidade (sem defeito e sem reservas para o mesmo período);
- **Durante a execução do processo** (imediatamente antes da atividade começar): Neste caso a atividade pode estar no estado *Waiting* ou *Ready*. Prevendo a necessidade de definição dos recursos, o mecanismo de execução ativa a instanciação que age da mesma forma que no item anterior, para que o início da atividade não seja atrasado. O projetista pode modificar a escolha do recurso selecionado se não concordar com a sugestão.

6.2 Políticas de Instanciação

Segundo Feiler [4], políticas de processos de software consistem em "princípios que conduzem o desenvolvimento e/ou a execução de processos de software". No modelo APSEE, uma política pode ser informalmente descrita como um conjunto de propriedades que atuam na formação e execução de modelos de processos de software, representando um conhecimento gerencial genérico e reutilizável para diferentes contextos.

O uso de tipos e especificação precisa dos componentes do ambiente tornou possível a construção de um mecanismo de instanciação guiado por políticas. Além do comportamento citado na seção anterior, o mecanismo de instanciação utiliza critérios para escolha do recurso baseado em políticas habilitadas naquela atividade. Uma política de instanciação consiste de uma especificação textual que contém critérios da organização definidos pelo usuário e pode estar habilitada em uma atividade, em um processo (estando habilitada em todas as atividades componentes), ou em uma organização (isto é, em todos os processos existentes na organização), permitindo a sua reutilização em diferentes contextos.

As políticas são definidas através de uma linguagem simples e compacta, que define

condições para que a política seja seguida e uma lista de critérios a serem adotados se as condições forem verdadeiras. A seguir, a linguagem de políticas de instanciação é descrita de forma sucinta e informal juntamente com um exemplo de seu uso.

6.3 Componentes da Política de Instanciação de Recursos

Os principais componentes de uma política de instanciação de recursos são:

- **Identificador** da política; **Nome** da política; e **Descrição** textual da política; **Tipo da política (*Police_TypeID*)**, na hierarquia de tipos (*ApseeTypes*);
- **Interface da Política (*Interface*).** Especifica o tipo de objeto APSEE que será tratado na condição da política, ou seja, o objeto cujas propriedades são avaliadas. A interface pode ser *Resource* ou *Activity*. O valor *default* deste atributo é *Activity*;
- **Tipo aplicável (*ApplySubtype*).** Especifica o tipo de recurso para o qual a política se aplica. Uma atividade pode ter várias políticas de instanciação de recursos habilitadas, porém cada uma instancia um tipo de recurso. Por exemplo, uma política que se aplica ao tipo "Handheld" somente será levada em consideração em atividades que necessitam de recursos desse tipo ou seus subtipos (PalmOS e PocketPC, por exemplo);
- **Condições (*Condition*).** Condições lógicas que devem ser satisfeitas para que os critérios da política sejam seguidos no momento da instanciação. Estas condições são formadas através de funções pré-definidas de acesso e verificação dos componentes do ambiente;
- **Critérios de Restrição (*RestrictBy*).** Se as condições forem satisfeitas, os critérios de restrição definidos pelo usuário são levados em consideração para instanciação. Estes critérios ajudam a diminuir o espaço de busca por recursos compatíveis a serem alocados;
- **Critério de Ordenação (*OrderBy*).** Após aplicação dos critérios de restrição, o critério de ordenação é utilizado para ordenar a sugestão de recursos para o usuário. Caso não haja critério de ordenação, a sugestão será ordenada pelo menor custo;

Se as condições não forem satisfeitas, a instanciação restringe os recursos pelo tipo e ordena-os pelo menor custo. Alguns critérios de restrição possíveis são mostrados na tabela 2, enquanto que alguns critérios de ordenação são mostrados na tabela 3.

Tabela 2 – Critérios de Restrição na instanciação de recursos

Critério (parâmetros)	Significado
Belongs_to (tipo_de_recurso)	Restringir recursos a sugerir para os que pertencem a algum recurso do tipo dado como parâmetro. Por exemplo, solicitar que sejam sugeridos apenas os recursos que pertençam a salas.
NotBelongs_to (tipo_de_recurso)	Oposto ao critério Belongs_to
No_Requirements	Sugerir apenas os recursos que não requerem nenhum outro recurso para serem utilizados.
Requires (tipo_de_recurso)	Sugerir recursos que requerem recurso do tipo dado como parâmetro.
NotRequires (tipo_de_recurso)	Oposto do critério Requires.
GetMetric(id_métrica, < | > | = | <= | >= | <>, valor)	Sugerir recursos que atendem à comparação com o valor de uma métrica existente.
Cost(< | > | = | <= | >= | <>, valor)	Sugerir recursos que atendem à comparação com o custo.
MTBF(< | > | = | <= | >= | <>, valor, unidade de tempo)	Sugerir recursos que atendem à comparação com o atributo MTBF (*médium time between failure*). Por exemplo: MTBF(>= , 60, "dias"),
Restrições específicas para recursos do tipo Consumables	
Consumable_New	Sugerir recursos consumíveis que nunca foram usados.
Max_PercentUsed(valor)	Sugerir recursos com percentual de uso menor que o valor fornecido.
AmountAvailable(< | > | = | <= | >= | <>, valor)	Sugerir recursos consumíveis que satisfazem a condição de disponibilidade expressa nos parâmetros dados.

Tabela 3 – Critérios de Ordenação na instanciação de Recursos

Critério	Significado
Low_Cost	Ordenar recursos pelo menor custo
High_Cost	Ordenar recursos pelo maior custo.
High_Mtbf	Ordenar pelos recursos que falham menos.
Low_Mtbf	Ordenar pelos recursos que falham mais.
Metric_higher(id_métrica)	Sugerir recursos em ordem decrescente do valor da métrica cujo identificador é passado como parâmetro.
Metric_lower(id_métrica)	Sugerir recursos em ordem crescente do valor da métrica cujo identificador é passado como parâmetro.

6.4 Gramática

A gramática da linguagem de instanciação de recursos é definida na Figura 15 com os componentes apresentados na seção 6.3. Classes Prosoft [16] foram construídas para mapear os componentes da política após análise léxica e sintática da mesma, e permitir que a análise semântica seja realizada (as classes não são apresentadas por questões de espaço).

<ResInstant_Policy> → **Id** <string> **(**[**Name** <string>] [**; Description** <string>] **;**
PolicyTypeID <type_id> **; Interface** <pol_interface> **; ApplyToType** <type_id>;
Conditions <condition> **; RestrictBy** <RestrictionCriteria> **; OrderBy** <OrderCriteria>)
<pol_interface> → <label> **:** <type>
<condition> → <policy_operand> [<opt_relation>] [<opt_connection>]
<opt_relation> → <comparison> <policy_operand>
<comparison> → **> | < | >= | <= | = | <> | contains | not_contains** | **sub_type_of**
<opt_connection> → <conn_type> <condition>
<conn_type> → **and** | **or**
<policy_operand> → <policy_object> **.** <operators>
<operators> → <policy_operator>*
<policy_operator> → <method_id> <parameters> | <reserved_word> | ∪ | ∩
<parameters> → <policy_operand>*
<reserved_word> → **any**| **all** | **no**
<RestrictionCriteria> → <string>* /* vários critérios de restrição podem ser definidos */
<OrderCriteria> → <string> /* apenas um critério de ordenação pode ser definido */
<label> → <string>
<type> → **Activity | Resource**
<policy_object> → <string>
<method_id> → <string>

Figura 15 Gramática da linguagem ResourceInstantiationPolicies

6.5 Funções embutidas na linguagem

A linguagem define um conjunto de funções aplicáveis aos tipos Activity, Artifact, Process, Resources, Agent, Groups, Roles e Connections, consultando informações sintáticas acerca do processo em questão. As funções são usadas na definição das condições da política de instanciação (no item method_id). O conjunto de funções disponíveis para o programador é fixo, determinado por primitivas disponíveis no mecanismo APSEE. Por questão de espaço, somente uma pequena lista de funções disponíveis é mostrada na Figura 16. Algumas dessas funções são utilizadas nos exemplos mostrados na seção 6.6.

Get_roles → set of roles	Is_active → bool
Is_performed_by_agents → bool	Get_sucessors → set of activities
Get_duration → real	Get_sucessors_oftype(type) → set of activities
Get_agents →set of agents	Get_artifact_connections_to(activity) → set of activities
Is_enacting →bool	Get_types_reqResources → set of string
Is_late_to_begin → bool	Get_input_artifacts → set of artifacts
Is_waiting → bool	Get_output_artifacts → set of artifacts

Figura 16. Operações sobre atividades para políticas de instanciação.

6.6 Exemplos de política de instanciação de recurso

Nas figuras 17 e 19 são mostrados exemplos de políticas de instanciação de recursos na sintaxe fornecida pela gramática da Figura 15. Os exemplos mostram a capacidade de reutilização das políticas em diferentes processos, domínios de aplicação e diferentes organizações. Métricas armazenadas sobre componentes do processo são consultadas nas políticas e podem servir de critério de ordenação para recursos.

ID:	**"**Pol_Inst1**"** **Name:** "Atividade atrasada envolvendo cliente e uso de computadores handheld"
Description:	"Se atividade está atrasada e é do tipo encontro com cliente, então obter computadores handheld que tenham taxa de utilização anual menor que 10% e tempo médio entre falhas maior que 30 dias e ordená-los pelo maior MTBF"
Interface:	a: Activity;
ApplyToType:	"handheld"
Conditions:	a.is_late_to_begin() **and** a.get_type() **sub_type_of** "Meet Customer"
RestrictBy:	getMetric("taxa_utilização_ano", <, 10) MTBF(>, 30, "days")
OrderBy:	High_MTBF

Figura 17. Política "Instanciação para handhelds"

ID:	**"**Pol_Inst2**"** **Name:** "Atividade de codificação requerendo impressora"
Description:	"Se atividade do tipo codificação produz código fonte então obter impressoras que não requeiram papel colorido e que requeiram papel A4 e ordenar pelas de maior velocidade"
Interface:	a: Activity;
ApplyToType:	"Printer"
Conditions:	a.get_output_artifacts.any.get_type() **sub_type_of** "SourceCode" **and** a.get_type() **sub_type_of** "Coding"
RestrictBy:	NotRequires("Color Paper") Requires("A4Paper")
OrderBy:	Metric_Higher("speed")

Figura 18. Política "Instanciação para impressora"

7. Trabalhos Relacionados

A maioria das abordagens que tratam coordenação de atividades não distingue entre agentes e recursos [17]. Agentes são normalmente tratados como recursos de apoio, enquanto que os recursos de apoio, quando considerados, nem sempre possuem informação detalhada sobre seu estado, impedindo verificação automática e adoção de estratégias para alocação. Mesmo em trabalhos na área de *workflow* [10][20], onde aspectos organizacionais são tratados com mais detalhes, pouca atenção é dada à gerência dos recursos não humanos.

Plekanova [24] propõe alocação de recursos humanos através de programação linear usando informações sobre recursos disponíveis e suas capacidades (pessoas e habilidades) e recursos requeridos. Exemplos de PSEEs que tratam a necessidade de modelar e gerenciar recursos são o MVP-L [19] e o APEL [3]. MVP-L permite modelagem de recursos, mas não controla o acesso aos mesmos. Da mesma forma, APEL trabalha com aspectos organizacionais de forma ortogonal às questões de processo (abordagem similar à proposta deste artigo), porém não incorpora aspectos de escalonamento. Em uma avaliação de PSEEs realizada por Lonchamp [5], o único ambiente avaliado a tratar o conceito de recursos sem denotar pessoas ou dados é o ambiente E3, e mesmo assim, apenas trata o conceito de alocação de tempo e máquinas para atividades, desconsiderando outros tipos de recursos.

Um trabalho que se preocupa com todos os tipos de recursos tratados neste artigo é o do grupo do Prof. Leon Osterweil [11][17]. Apesar de adotarem o conceito genérico de recursos, sua abordagem é motivada pela otimização da alocação de recursos. Por isso, seu modelo provê tipos e instâncias de recursos, reservas para quaisquer recursos, além de composição (*belongs_to*) e requisitos de recursos (*requires*). Uma desvantagem desta proposta é que

devido ao tratamento igual dado a tipos de recursos diferentes, o modelo permite que um recurso humano seja composto de uma máquina por exemplo, e para contornar esse problema são propostos critérios de participação em cada grupo de recurso através de regras.

No Brasil algumas propostas na área de processos de software permitem definição de recursos de apoio, porém não estabelecem formas de auxiliar a instanciação dos mesmos. O ambiente ExPSEE [8] permite definição precisa e instanciação de atividades do processo em relação a artefatos, atores (agentes) e ferramentas e a Estação TABA [21] possui um modelo recente para definição de processos que permite definir recursos humanos e ferramentas [15]. Mecanismos de coordenação de atividades modelados com redes de Petri foram propostos por Raposo, Magalhães e Ricarte [23]. Estes mecanismos tratam dependências temporais e de gerenciamento de recursos (de forma genérica, porém com uma classificação diferenciada) entre atividades e podem ser utilizados em ambientes de coordenação de atividades.

8. Conclusões

A qualidade na definição de processos de desenvolvimento de software é um dos elementos-chave para que uma organização possa atingir melhores níveis de maturidade. A tecnologia de processos de software permite, entre outros benefícios, um controle preciso na alocação e consumo de recursos durante o desenvolvimento de software. Porém nem sempre os ambientes existentes permitem modelagem precisa destes recursos.

Este artigo apresentou a modelagem e gerência de recursos de apoio em um PSEE. Primeiro, a terminologia adotada foi definida para tratar somente os recursos que fornecem apoio ao desenvolvimento de atividades, excluindo os agentes e os artefatos produzidos. Em seguida, o foco foi direcionado à apresentação detalhada do modelo de recursos e sua integração ao ambiente APSEE. Neste modelo são tratados recursos de uso exclusivo, compartilháveis e consumíveis.

O gerenciador de recursos proposto foi integrado a um ambiente específico de desenvolvimento de software orientado a processos, mas os autores acreditam que suas idéias podem também ser utilizadas em outros ambientes de coordenação de tarefas que requerem recursos. A integração permitiu a construção de um mecanismo de instanciação baseado na situação corrente do processo e em critérios definidos pelo usuário.

O mecanismo de instanciação fornece uma linguagem para definição dos critérios de instanciação de recurso (apresentada na seção 6). O mesmo conceito de instanciação está sendo desenvolvido também para agentes de uma atividade em outro trabalho do grupo [13], que também trata do conceito de políticas dinâmicas para execução do processo. Além disso, o conceito de políticas está sendo trabalhado na verificação estática do modelo de processo [25]. Trabalhos futuros incluem a experimentação da tecnologia proposta, através do protótipo sendo concluído a fim de validar e ajustar os conceitos apresentados, assim como aprimorar as métricas disponíveis a serem consultadas na execução das políticas de instanciação.

9. Referências Bibliográficas

1. Derniame, J.; Kaba, B.; Wastell, D.(eds.). **Software Process: Principles, Methodology and Technology**. Lecture Notes in Computer Science, vol. 1500. Springer-Verlag, 1998.
2. Dowson, M.; Fernström, C. Towards Requirements for Enactment Mechanisms. 3th European Workshop on Software Process Tech. **Proceedings...** Berlin: Springer, 1994.
3. Estublier, J. et al. **APEL: A graphical yet executable formalism for process modeling**. In: Automated Software Engineering, March 1997.
4. Feiler, P.; Humphrey, W. **Software Process Development and Enactment: Concepts**

and Definitions. 2nd International Conference on Software Process. Berlin, Feb./1993.
5. Finkelstein, A. et al. (Ed.). **Software Process Modelling and Technology**. 1994.
6. Fuggetta, A.; Wolf, A. (Eds.) Software Process. Wiley, 1996.
7. Gimenes, I.M. **Uma Introdução ao Processo de Engenharia de Software:** Ambientes e Formalismos. Jornada de Atualização em Informática, 13., Caxambu: SBC, 1994.
8. Gimenes, I. M. S. **ExPSEE: Um ambiente Experimental de Engenharia de Software Orientado a Processos**. Relatório de Projeto, Universidade Estadual de Maringá, 2000.
9. Huff, Karen. In: Fuggetta, A.; Wolf, A.(Eds.) **Software Process**.Wiley, 1996.
10. Kappel, G et al. **Coordination in workflow management systems: A rule based approach**. In Conen, W.; Neuman, G. (Eds.) Coordination technology for collaborative applications. LNCS, n. 1364, 1997.
11. Lerner, B. S. et al. **Modeling and Managing Resource Utilization in Process, Workflow and Activity Coordination**. Tech. Report. Dept. of C.S., Univ. of Massachusetts (UM-CS-2000-058) Aug., 2000
12. Reis, R.; Lima Reis, C.; Nunes, D. **Gerenciamento do Processo de Desenvolvimento Cooperativo de Software no Ambiente PROSOFT**. Simpósio Brasileiro de Engenharia de Software, 12, 1998, Maringá. Anais... SBC, 1998, p. 221-236.
13. Lima Reis, Carla A. APSEE: Uma abordagem baseada em conhecimento para gerência de processos de software evolutivos. PPGC-UFRGS, Proposta de Tese, 2001.
14. Lonchamp, J. **A Structured Conceptual and Terminological Framework for Software Process Engineering**. 2nd Int. Conference on the Software Process, Berlin. Feb, 1993.
15. Machado, L.F.C. **Modelo para Definição de Processos de Software na Estação TABA**, Dissertação de MSc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, mar 2000.
16. Nunes, D.J. **Estratégia Data-driven no Desenvolvimento de Software**. Simpósio Brasileiro de Engenharia de Software, 6. **Anais...** p. 81-95, Gramado, 1992.
17. Podorozhny, R.et al. **Modeling Resources for Activity Coordination and Scheduling**. 3th Intl. Conf. on Coordination Models and Languages. April 1999. LNCS, 1594.
18. Reis, Rodrigo. Lima Reis, C.; Nunes, D. Automated Support for Software Process Reuse: Requirements and Early Experiences with the APSEE model.7th International Workshop on Groupware. **Proceedings…** Darmstadt-Germany, September/2001. IEEE CS Press.
19. Rombach, D. Verlage, M. **How to assess a software process modeling formalism from a project member's point of view**. In: 2nd Int. Conference on the Software Process, 1993.
20. Schäl, Thomas. **Workflow Management Systems for Process Organizations**. 2. ed. Berlin. Lecture Notes in Computer Science Vol. 1096, Springer, 1998.
21. Travassos, G. H. **O Modelo de Integração de Ferramentas da Estação TABA.** Tese de Doutorado, Engenharia de Sistemas e Computação, COPPE/UFRJ, Março, 1994.
22. Westfechtel, B. **Models and Tools for Managing Development Processes**. Lecture Notes in Computer Science vol. 1646. 1999.
23. Raposo, A.B.; Magalhães, L.P.; Ricarte, I.L.M. **Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments**. *International Journal of Computer Systems Science & Engineerin*g. Special Issue on Flexible Workflow Technology Driving the Networked Economy. CRL Publishing. September 2000.
24. Plekhanova, Valentina. **Capability and Compatibility Measurement in Software Process Improvement**. Proceedings of the 2nd European Software Measurement Conference – FESMA'99. Amsterdam, The Netherlands, October, 1999.
25. Reis, Rodrigo Q.; Lima Reis, C. A.; Nunes, D. J. **Apsee-StaticPolicy: Verificação de políticas estáticas em modelos de processos de software**. Simpósio Brasileiro de Engenharia de Software - SBES'2001. Rio de Janeiro, Outubro 2001.