

# Um Ambiente para Modelagem e Execução de Processos

Marcus Felipe M. C. da Fontoura<sup>1</sup> e Carlos José P. de Lucena<sup>2</sup>

## Resumo

Este trabalho apresenta a arquitetura e as características principais de um ambiente para modelagem e análise de processos. A modelagem dos processos, tanto de software quanto de negócio, é feita através de uma linguagem de processos baseada em Redes de Petri que manipula os objetos relevantes à definição dos processos tais como artefatos, cargos e ferramentas.

Palavras-Chaves: Processos, Programação de Processos, Ambientes de Engenharia de Software, Linguagens de Processo.

## Abstract

This work presents the architecture and main features of an environment for process modeling and analysis. The software and business processes are modeled through a process language based on Petri Nets which handles all important objects for process definition such as artifacts, roles, and tools.

Keywords: Processes, Process Programming, Software Engineering Environments, Process Languages.

## 1. Introdução

Vários estudos têm sido feitos no sentido de se desenvolver “Fábricas de Software” [16] com o intuito de aprimorar o desenvolvimento e a qualidade dos produtos de software. Junto com a idéia de “Fábricas de Software” surge o conceito de controle do “Processo de Software”.

“Ambientes de Engenharia de Software dirigidos por Processo” (*PSEE*) caracterizam-se por dar suporte a descrição e execução de processos de modo a auxiliar e controlar todas as atividades envolvidas na produção e manutenção de um software operacional. Além disso, a interface do usuário com o ambiente é orientada pelo processo definido [15].

Este trabalho apresenta as características e a arquitetura de um *PSEE* contendo editores gráficos de processo, visualizadores e ferramentas de apoio à execução. A linguagem de representação de processos foi baseada em Redes de Petri incorporando alguns aspectos definidos em JIL [29], tais como tratamento de exceções local e global, existência de pré-condições, pós-condições e restrições e o conceito de subprocessos. O uso de Redes de Petri em linguagens de definição de processos é bastante aceito, principalmente pela facilidade na representação de sincronismo, necessária para esse tipo de aplicação.

O desenvolvimento de software é uma tarefa que deve ser realizada pelo conjunto “homem-máquina”. Portanto o ambiente proposto também dá suporte a uma descrição “informal” do processo mais facilmente entendida por seres-humanos. Assim sendo, todo processo definido terá uma representação formal que será entendida, instanciada e executada, e uma representação informal, que servirá como documentação para auxiliar os usuários no entendimento do processo. A modelagem da representação informal (documentação) dos

<sup>1</sup> mafe@inf.puc-rio.br, Laboratório de Engenharia de Software, Departamento de Informática, PUC-Rio.

<sup>2</sup> lucena@inf.puc-rio.br, Laboratório de Engenharia de Software, Departamento de Informática, PUC-Rio.

processos é baseada em hipertexto. Existe um mapeamento direto das representações formal e informal através de uma relação entre atividades ou tarefas, representadas na descrição formal, e nós de um hipertexto, representados na descrição informal do processo.

Uma outra característica do ambiente proposto é o fato de sua interface de execução ter sido desenvolvida em WWW (*World Wide Web*), permitindo assim que seus usuários possam estar espalhados em uma WAN (*Wide Area Network*).

A seção 2 apresenta a arquitetura e as características básicas do ambiente proposto. Em seguida, na seção 3, é apresentado um estudo de caso da utilização do ambiente. A seção 4 descreve alguns trabalhos relacionados e em seguida são apresentadas as conclusões e trabalhos futuros.

## 2. Arquitetura e Características Básicas

### 2.1 Arquitetura

PAGoDE (*Process Analysis design and Documentation Environment*) é um ambiente de engenharia de software orientado a processos [15, 18]. A idéia básica é permitir que se modelem processos, tanto de software quanto de negócio, e que os processos modelados possam ser executados e acompanhados com auxílio do ambiente.






O ambiente pode ser dividido em dois módulos distintos:

- Módulo de edição de processos: onde são definidos os processos através de uma linguagem de processos;
- Módulo de execução: que possui uma máquina de instanciação e execução (*enactment engine*) de processos. Esse módulo é responsável por interpretar a descrição do processo que se deseja executar, instanciá-la e controlar sua execução. O controle de uma interface baseada em agendas para execução de processos que necessitem de intervenção humana também faz parte deste módulo.

Ambos os módulos fazem uso de uma camada de gerenciamento de objetos, que é responsável por fornecer os objetos necessários para a implementação de cada um dos módulos.

O repositório de PAGoDE contém objetos de definição, utilizados na definição dos processos, objetos de execução, responsáveis pela instanciação, execução e controle dos processos, e objetos de métrica, que são objetos que contém informações sobre processos previamente executados no ambiente.

Objetos de definição são os objetos que são manipulados pela linguagem de processos do ambiente. A tabela abaixo apresenta todos os objetos de definição.

| Objeto  | Descrição  |
|---|--|
|  Processos     | <p>Cada objeto processo contém a descrição de um processo modelado. PAGO DE permite que processos possam ser decompostos em subprocessos, permitindo assim o aninhamento de processos em vários níveis. Os processos podem ser classificados como batch, que são processos que não necessitam intervenção humana e processos de usuário, que são processos que necessitam intervenção humana.</p>  |
|  Cargos        | <p>Os objetos cargos representam a estrutura organizacional modelada. A estrutura de cargos é hierárquica, ou seja, cargos podem conter outros cargos. Cada cargo pode possuir vários atores, que são objetos que representam os funcionários da organização sendo modelada. São os atores que vão interagir nos processos de usuário.</p>   |
|  Artefatos    | <p>Artefatos (ou documentos) são os produtos que são consumidos e/ou gerados pelos processos. Um processo de usuário especifica que um determinado cargo deve produzir um determinado artefato, por exemplo, uma requisição. Quando esse processo é executado, um ator daquele cargo deve ser designado para editar tal requisição, produzindo assim o artefato especificado. Os artefatos são compostos por campos. No exemplo acima a requisição poderia conter campos como Nome, Idade e Salário.</p>   |
|  Ferramentas | <p>PAGO DE possui dois tipos de ferramentas: usuário e batch. Ferramentas de usuário são utilizadas para auxiliar a edição de artefatos. O ambiente permite que cada campo de um determinado artefato tenha uma ferramenta de auxílio. Um artefato que representa a documentação de um sistema, por exemplo, pode ser editado com auxílio de um editor de diagramas. Ferramentas batch são utilizadas em processos batch para processar artefatos sem a necessidade de intervenção humana. Os dois tipos de ferramentas representam aplicações externas ao ambiente.</p> |
|  Advisors    | <p><i>Advisors</i> (ou <i>project advisors</i>) são um tipo especial de ferramenta batch que atua acompanhando a execução dos processos e realizando tarefas de auxílio ao gerenciamento do projeto. <i>Advisors</i> podem fazer uso dos objetos de métrica para obter informações sobre execuções anteriores de um determinado processo.</p>  |

A persistência dos objetos manipulados em PAGO DE é garantida pela camada de armazenamento. Essa camada torna os métodos de armazenamento e a estrutura da base de dados utilizada na implementação do ambiente totalmente transparentes para o resto do sistema. Isso permite que se possa alterar as características e a definição da base de dados utilizada sem que se altere a implementação das demais camadas do sistema.

A figura abaixo ilustra essa arquitetura.



**Figura 1: Arquitetura do ambiente PAGoDE**

## **2.2 Tipos de Usuários**

O sistema possui dois tipos de usuários bem distintos, a saber:

- Engenheiro de negócio: responsável pela modelagem e acompanhamento dos processos;
- Operador: funcionário da organização (ator) que interage com o sistema para a execução de processos de usuário.

O engenheiro de negócio faz a modelagem dos processos baseado no seu conhecimento do negócio da organização. No caso de organizações de desenvolvimento de software o papel do engenheiro de negócio é feito por engenheiros de software que possuam conhecimento dos processos de desenvolvimento utilizados.

Em geral as organizações possuem algum tipo de documentação dos seus processos. Neste caso a função do engenheiro de negócio é a de modelar os processos através da linguagem de processos do ambiente, com base nessa documentação. No caso em que não exista nenhuma documentação dos processos, essa documentação deve ser gerada para que então os processos possam ser modelados.

Além da modelagem o engenheiro de negócio é também responsável pelo acompanhamento da execução dos processos. Obviamente uma organização de porte médio deverá possuir vários engenheiros de negócio, responsáveis, por exemplo, por diferentes departamentos (modelados em PAGoDE pela estrutura de cargos).

Os operadores de uma organização são os responsáveis pela execução de processos do usuário. Na instanciação de um processo do usuário, o operador (modelado por um objeto ator) responsável pela execução de tal processo deve ser especificado. A interface vista por cada operador é baseada em agendas onde o operador pode verificar quais processos pendentes ele tem para realizar bem como instanciar novos processos. A idéia de interface baseada em agendas é usada em vários ambientes tais como ProcessWeaver [Bourdon 92] e IPSE 2.5 [9].

A interface de agendas foi implementada em WWW (World Wide Web), permitindo assim que a execução dos processos seja distribuída. O uso de WWW também permite que sejam modelados processos que transcendam os limites organizacionais. Um exemplo simples é a modelagem de processo de compra entre uma empresa fornecedora e seus clientes. A empresa fornecedora pode modelar esse processo criando um subprocesso de pedido de compras que é executado por um ator denominado “Comprador”. Portanto, quando alguma empresa cliente for realizar a compra basta que um operador desta empresa se identifique como “Comprador” e instancie um novo processo de pedido de compras.

### **2.3 Documentação de Processos (Representação Informal)**

Conforme foi dito anteriormente um dos trabalhos do engenheiro de negócios é o de converter a documentação de processos já existente na organização para uma modelagem baseada na linguagem de processos do ambiente. No entanto, o ambiente também permite que uma representação textual (que é o tipo de documentação de processos geralmente encontrada nas organizações) possa ser associada a cada processo modelado.

Portanto o sistema é capaz de armazenar duas representações para cada processo: uma representação formal, descrita na linguagem de processos do ambiente, e uma representação textual (ou informal). A representação formal será usada pela máquina de instanciação e execução enquanto a representação textual será usada como documentação pelos operadores.

Cada operador poderá consultar a documentação (representação informal) dos processos para obter informações relevantes tais como objetivos do processo, tarefas que devem ser realizadas, artefatos que manipula e relacionamentos entre processos. Em [11] encontra-se uma descrição de atributos relevantes para documentação de processos.

Como a estrutura de processos é hierárquica, ou seja, processos podem ser compostos por outros processos (subprocessos) em vários níveis de profundidade, essa hierarquia deve ser refletida também na documentação. Outra característica importante da representação dos processos em PAGOGE é que cada operador que pertença a um determinado cargo é capaz de visualizar apenas os processos que podem ser executados por aquele cargo (ou por seus subcargos). Isso significa que operadores que pertençam a cargos diferentes podem possuir visões diferentes das hierarquias de processos do ambiente.

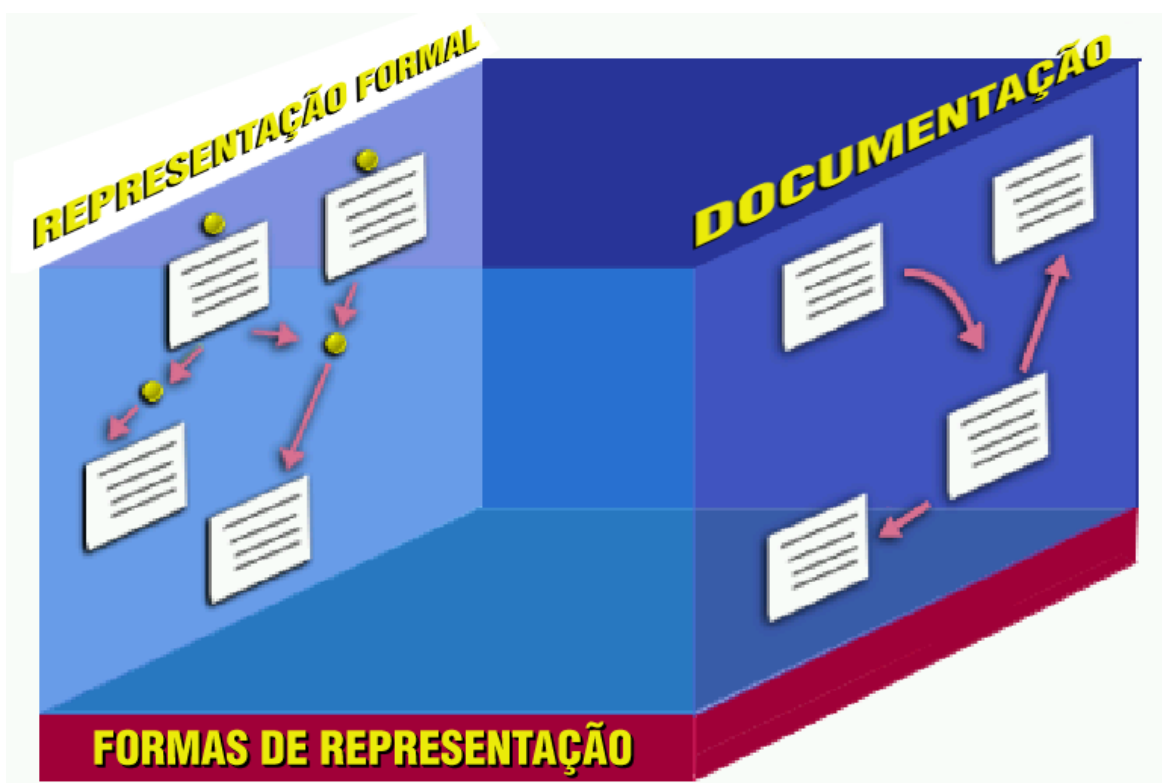
A necessidade de representação de hierarquias e controle de visões por cargo são dois requisitos importantes que são atendidos pelo sistema de documentação do ambiente. Para atender tais requisitos a forma de representação escolhida para a documentação dos processos foi a hipertextual baseada no Modelo de Contextos Aninhados (*NCM - Nested Context Model*) [27].

O *NCM* baseia-se nos conceitos de nós de conteúdo e nós de composição, onde os nós de conteúdo representam a informação (no caso do PAGOGE cada nó de conteúdo representa a documentação de um processo) e os nós de composição representam relacionamentos entre

nós (tanto de conteúdo quanto de composição). A implementação dessa estrutura baseia-se no *Design Pattern Composite* [14].

O sistema de documentação foi construído criando-se um nó de composição para cada cargo representando sua visão da hierarquia de processos. Obviamente esses nós de composição são compostos por outros nós de composição representando a estruturação hierárquica de cada processo (decomposição em subprocessos).

Em PAGoDE as duas formas de representação de processos, formal e informal, são integradas. O operador pode consultar a documentação de processos e a partir daí executá-lo, bem como enquanto estiver executando um processo consultar sua documentação. A figura abaixo representa esse mapeamento entre as formas de representação de processos do sistema.



**Figura 2: Mapeamento entre as representações de processos**

A consulta da documentação do sistema é feita na mesma interface baseada em agendas vista pelos operadores.

#### **2.4 Linguagem de Modelagem de Processos (Representação Formal)**

A linguagem de processos do ambiente PAGoDE (PAGoDE Language) é baseada na linguagem JIL [29]. O fato de JIL<sup>3</sup> ser uma linguagem textual torna muito difícil o acompanhamento (*trace*) da execução de processos escritos nesta linguagem. Essa dificuldade vem do fato de que processos podem ser executados em paralelo, necessitam de pontos de

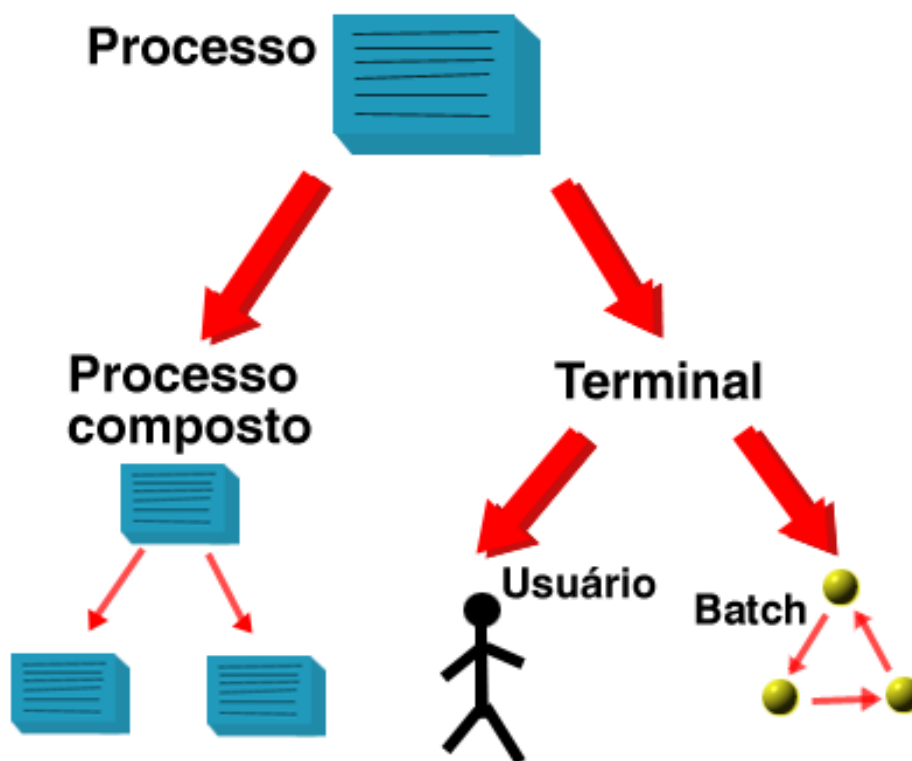
<sup>3</sup> O ambiente Julia apresenta uma representação gráfica para JIL, contudo essa representação ainda está em desenvolvimento e, portanto, não é tão bem aceita e conhecida quanto, por exemplo, Redes de Petri.

sincronismo e possuem dependências entre si [6]. Essas características são mais facilmente entendidas através de uma representação gráfica.

Uma representação visual que atende aos requisitos descritos acima é a de Redes de Petri [22]. Redes de Petri têm sido bastante utilizadas em linguagens de processos [5, 10]. A execução de processos pode ser vista como um sistema de tempo real onde existem restrições temporais para a execução de tarefas. Portanto o modelo de Redes de Petri utilizado em PAGoDE Language é o de Redes de Petri Temporizadas (*Time Petri Nets*) [20, 21].

PAGoDE Language é uma linguagem para programação de processos que integra os conceitos existentes em JIL com uma representação gráfica baseada em Redes de Petri Temporizadas. PAGoDE Language manipula os objetos de definição (processos, cargos, artefatos, ferramentas e *project advisors*) apresentados anteriormente.

Processo é o elemento principal na linguagem. A figura abaixo mostra a hierarquia de classes que modelam os processos no ambiente PAGoDE.



**Figura 3: Hierarquia de classes de processo**

Conforme discutido anteriormente, processos de usuário são processos que necessitam de intervenção de atores para sua realização enquanto processos batch são executados apenas por ferramentas.

Cada processo é representado em PAGoDE Language por uma transição em uma Rede de Petri Temporizada. Para cada processo composto (formado por subprocessos que podem ser de usuário, batch ou compostos) a transição representa uma outra Rede de Petri Temporizada, formando assim uma Rede de Petri Temporizada Hierárquica (RPTH) [30].

Cada transição da RPTH contém a especificação do processo modelado naquela transição. Essa especificação pode ser feita tanto em tempo de modelagem quanto em tempo de execução. A forma da especificação varia com o tipo de processo sendo modelado (usuário, batch ou composto) porém existe uma parte comum referente à especificação dos seguintes atributos:

- Pré-condições, restrições e pós-condições<sup>4</sup>: assim como em JIL, representam as condições que devem ser atendidas para o início, durante a execução e para o término de um processo. Em PAGO DE Language as pré-condições habilitam o disparo das transições da RPTH;
- Definição do intervalo de tempo de disparo do processo (TDI, Tempo de Disparo Inicial, e TDF, Tempo de Disparo Final): o intervalo de disparo funciona como pré e pós-condições na execução dos processos;
- Tratamento de exceções: o tratamento de exceções de PAGO DE Language é exatamente igual ao de JIL. A violação de pré-condições, restrições e/ou pós-condições pode ser tratada localmente ou globalmente através do mecanismo de controle de fluxo reativo;
- Controle reativo: o controle reativo de PAGO DE é baseado no mecanismo de eventos, onde a ocorrência de um determinado evento pode disparar a execução de uma nova instância de processo que irá tratar esse evento.

#### 2.4.1 Processos de Usuário

Processos de usuário especificam a edição de um artefato por um ator de um determinado cargo. Portanto, para cada processo de usuário descrito em PAGO DE Language os seguintes atributos devem ser especificados:

- Artefato: representando um artefato previamente cadastrado no ambiente PAGO DE;
- Cargo: indicando o cargo (também previamente cadastrado em PAGO DE) que deve conter o ator que irá executar o processo;
- Ator: operador que irá executar o processo.

A edição do artefato poderá fazer uso de ferramentas de usuário que tenham sido cadastradas para auxiliar a edição de campos daquele artefato.

#### 2.4.2 Processos Batch

Processos batch são executados sem a intervenção humana. Processos batch representam o processamento de artefatos por ferramentas e são especificados em PAGO DE Language através dos seguintes atributos:

- Ferramenta: ferramenta batch (ou *advisor*) que irá realizar o processamento;
- Mapeamento: especificação do mapeamento entre os parâmetros de entrada e saída da ferramenta e os artefatos que serão processados e gerados.

##### 2.4.2.1.1 Project Advisors

---

<sup>4</sup> O atributo restrições não é utilizado em processos batch, uma vez que seu propósito é o de validar a execução do processo.



*Project advisors* são tipos especiais de ferramentas batch que podem fazer uso dos objetos de métrica do ambiente. A finalidade dos objetos de métrica é a de formar um repositório de informações sobre execuções prévias de processos. Esse repositório contém várias informações relevantes tais como tempo médio de duração de processos e comparação de desempenho de processos quando executados por atores diferentes.

Um exemplo de *Advisor* pode ser uma ferramenta que otimiza a distribuição das tarefas entre os atores levando em conta o desempenho desses atores em execuções prévias desses mesmos processos.

Na versão atual de PAGoDE não existe nenhum suporte para criação de novos *advisors*, contudo pretende-se incorporar ao ambiente assistentes (*wizards*) para auxiliar essa tarefa.

## 2.5 Biblioteca de Padrões de Processo

Padrões de design (*design patterns*) foram introduzidos por Christopher Alexander [4] para a área de arquitetura. Ele definiu um padrão de design como um elemento que descreve uma solução genérica para um problema que ocorre repetidamente em um determinado cenário. Esta solução genérica pode ser usada diversas vezes, sem a necessidade de repensar o problema. O que Alexander definiu pode ser facilmente utilizado no design de software [14, 1].

O uso de *design patterns* no desenvolvimento de software tem sido proposto também como uma forma de documentar *frameworks* [17, 13], afim de facilitar o seu entendimento e conseqüente uso.

Um padrão de design é composto por quatro partes essenciais [14, 21]:

- Nome: identificador que descreve o problema de design. Nomear um padrão de design facilita o trabalho do projetista, pois o aumento de vocabulário permite um design em alto nível. A comunicação de idéias aos companheiros de trabalho também é facilitada. Outra vantagem, é a possibilidade de documentar projetos de forma mais detalhada;
- Problema: descreve a situação onde pode ser aplicado um padrão de design. Explica o problema e o contexto do mesmo. Em geral esta parte inclui uma lista de condições que devem ser satisfeitas, afim de se aplicar corretamente o padrão de design;
- Solução: descreve os elementos que fazem parte da solução, seus relacionamentos, responsabilidades, e colaboradores. Esta parte deve descrever uma solução genérica, na medida em que um padrão de design deve poder ser aplicado em diversas situações;
- Consequências: são os prós e contras de aplicar-se um padrão de design. Esta parte contém a avaliação entre as alternativas de design, e a descrição dos custos e benefícios em aplicá-las.

O conceito de *design patterns* pode ser facilmente estendido para modelagem de processo (*process patterns*).

Uma grande diferença entre *process patterns* e *design patterns* é que os padrões de processo são sempre dependentes de domínio enquanto que os padrões de design não o são necessariamente. Os padrões de design apresentados em [14], por exemplo, não são dependentes de nenhum domínio de aplicação.

PAGoDE possui uma biblioteca de padrões de processo classificadas por domínio e permite que novos domínios sejam cadastrados assim como novos padrões. Dessa forma o engenheiro

de negócio pode fazer uso de processos bem estabelecidos no seu domínio de trabalho para modelar os processos de sua organização.

O estabelecimento de padrões de processo depende do conhecimento vasto do domínio de aplicação daquele processo. No caso do domínio de desenvolvimento de software vários padrões são estabelecidos. Em geral esses padrões são baseados em modelos de ciclo de vida de software, tais como Booch [7] e Rumbaugh [24], aliados aos aspectos gerenciais do desenvolvimento, como alocação da equipe e controle do cronograma do projeto.

## 2.6 Análise de Processos (Trace)

A análise de processos em PAGoDE é feita através de um mecanismo de *trace*, onde cada transição da RPTH é representada por cores diferentes identificando estados de execução diferentes. Os seguintes estados de execução são possíveis para cada processo:

- Ativo: o processo está executando naquele momento;
- Inativo ou bloqueado: está aguardando o início de sua execução;
- Abortado: teve sua execução interrompida devido à violação de alguma restrição.

A figura abaixo mostra a visualização de um *trace* em um determinado instante da execução de um processo em PAGoDE.

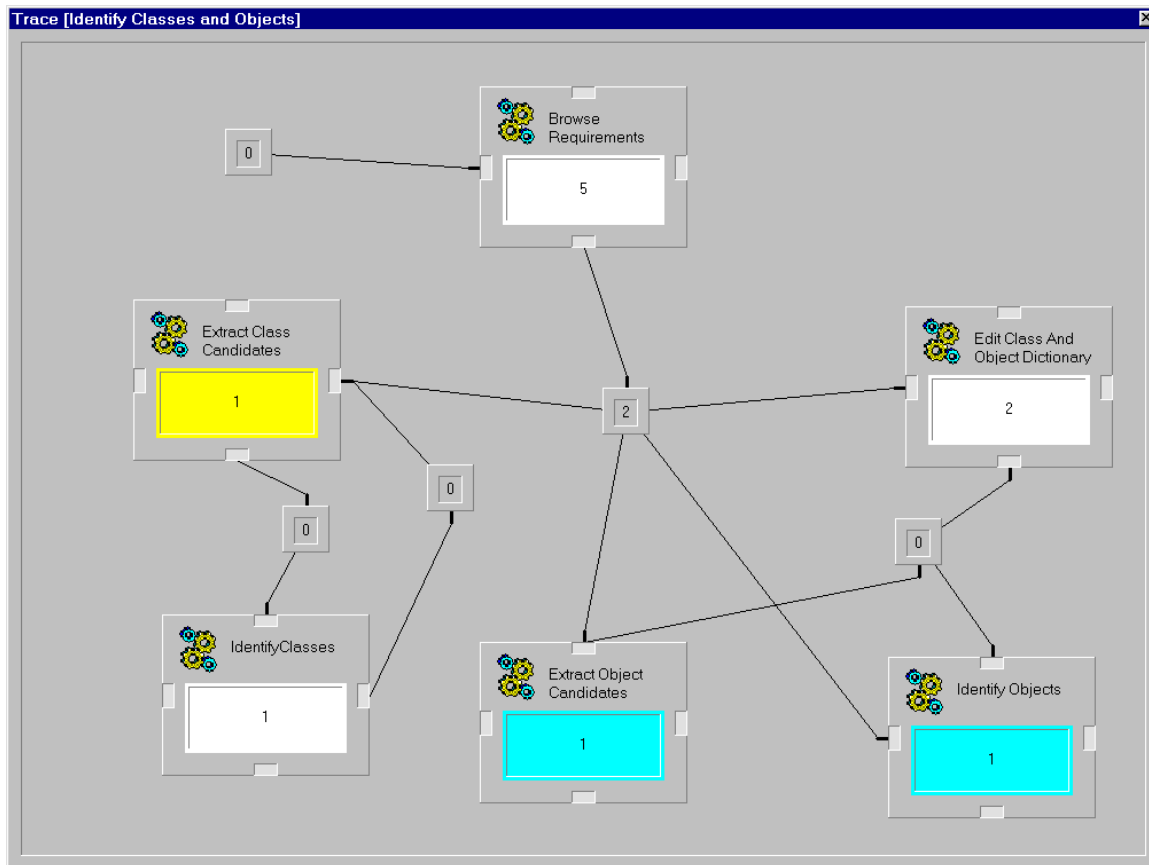


Figura 4: Acompanhamento de processos (*Trace*)

### 3. Estudo de Caso

Essa seção descreve um estudo de caso da utilização do ambiente PAGO DE na modelagem de um sistema de controle de processo industrial.

#### 3.1 Sistema de Informatização de Laboratório da Petrobrás Distribuidora (SILAB)

SILAB é o sistema de informatização do laboratório de análises de lubrificantes da Petrobrás Distribuidora que foi desenvolvido pela Fundação Padre Leonel Franca (PUC-Rio). O laboratório de lubrificantes é responsável por realizar análises de toda a produção de lubrificantes da Petrobrás.

Durante sua produção, um lubrificante deve ser analisado diversas vezes para se verificar se não houve contaminação em nenhuma etapa do processo. Lubrificantes são produzidos através da mistura de óleos básicos e aditivos. Logo após a mistura o lubrificante é armazenado e em seguida é envasado em frascos (por exemplo frascos de um litro) para ser distribuído. O laboratório é responsável pela análise dos óleos básicos e aditivos e pela análise do lubrificante nas seguintes etapas do processo: após a mistura, após o armazenamento e antes do envasamento.

A análise de um lote (seja de óleo básico, aditivo ou lubrificante) consiste na realização de vários ensaios (testes) em uma amostra deste lote e na comparação dos resultados desses ensaios com uma faixa de valores padrões. Caso os resultados estejam fora dos valores especificados a amostra é recusada e é feita uma reamostragem, ou seja, os ensaios são realizados novamente em uma nova amostra daquele lote. A aprovação ou não do lote depende dos resultados da reamostragem. Para cada amostra analisada no laboratório pode ser emitido um laudo, contendo os resultados dos ensaios daquela amostra.

Caso um lote de lubrificantes seja reprovado é emitido um relatório técnico chamado “Relatório de não conformidade” ou RNC informando a causa da reprovação do lote. Caso o lote reprovado seja de óleos básicos ou aditivos um outro relatório é emitido, “Relatório de não conformidade do fornecedor” ou NCF. A diferença básica é que óleos básicos e aditivos não são produzidos pela Petrobrás, portanto o relatório deve ser enviado para o fornecedor que teve seu lote recusado.

#### 3.2 Organização Hierárquica do Laboratório

O laboratório possui os seguintes tipos de funcionários:

- **Técnicos:** são os funcionários responsáveis pela realização das análises das amostras. Vários técnicos participam da análise de uma amostra pois cada técnico é responsável pela realização de um tipo específico de ensaio. Se para a análise de uma amostra forem necessários a realização de ensaios de “Viscosidade” e “Densidade”, os técnicos responsáveis pela realização desses ensaios participarão da análise desta mesma amostra;
- **Supervisores:** são os funcionários responsáveis pela aprovação ou não de uma amostra através da análise dos resultados dos seus ensaios. Note que o sistema consegue comparar os resultados de uma amostra com os valores padrão, contudo para que uma amostra seja aprovada ou reprovada é necessário que o supervisor dê seu parecer, pois mesmo que os resultados estejam fora dos limites especificados ele pode aprovar a amostra, ou por exemplo, solicitar a realização de alguns ensaios adicionais. A escolha de que ensaios devem

ser realizados em que amostras depende do tipo da amostra (óleo básico, aditivo ou lubrificante) e do produto propriamente dito (por exemplo: óleo básico “Bright Stock”, aditivo “MIV”, lubrificante “Lubrax Supra”). Contudo para alguns produtos os ensaios realizados não são sempre os mesmos. Neste caso, quando uma amostra deste tipo de produto chega no laboratório para ser analisada cabe ao supervisor a escolha de que ensaios serão realizados naquela amostra;

- Profissionais: são os funcionários responsáveis pela aprovação ou não de um lote. Mesmo que uma amostra de um lote seja reprovada, o supervisor responsável solicite uma reamostragem do lote e esta também seja reprovada, o supervisor não tem autoridade suficiente para reprovar o lote. O profissional responsável deve ser avisado, e ele sim, pode reprovar ou aprovar o lote. O profissional também pode solicitar que ensaios adicionais sejam realizados nas amostras já testadas ou que uma nova amostragem seja realizada. Um lote cuja amostra tenha sido reprovada pelo supervisor pode ser aprovado, contudo, neste caso diz-se que o lote foi aprovado com concessão técnica e um RNC é emitido;
- Auxiliar de Entrada: é o funcionário responsável pelo controle da entrada de amostras no laboratório.

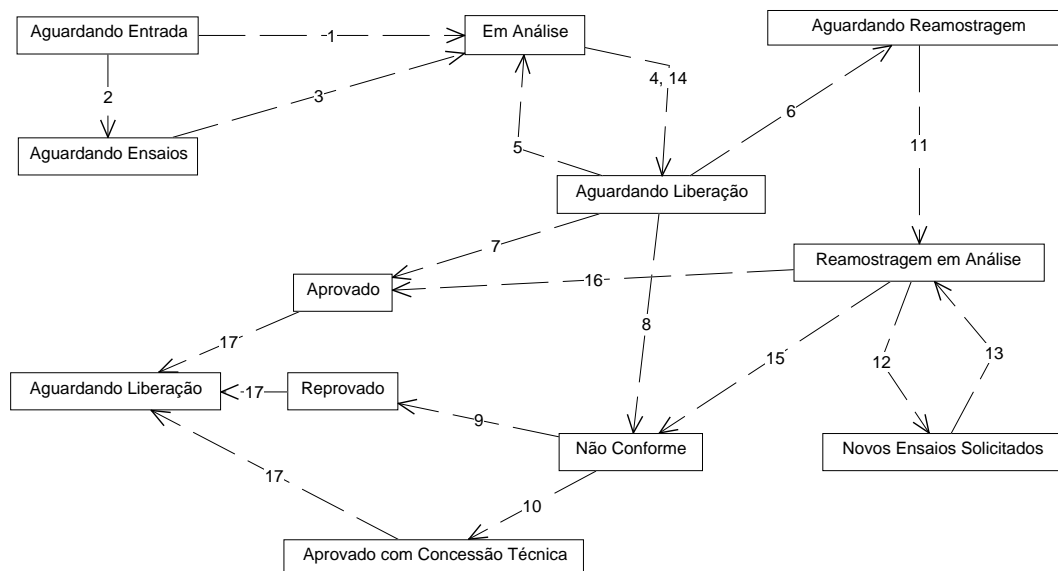
### **3.3 A Implementação Atual**

O SILAB atualmente está implementado em Microsoft Visual Basic, utiliza o gerenciador de banco de dados relacional Progress e está dividido em cinco módulos distintos:

- Cadastro: módulo utilizado para o cadastramento dos tipos de produtos, ensaios, que ensaios devem ser realizados em cada produto, faixas de valores padrões por ensaio, fornecedores e funcionários;
- Entrada: módulo utilizado pelo Auxiliar de Entrada para cadastrar a entrada das amostras no laboratório;
- Técnico: módulo utilizado pelos Técnicos para realização dos ensaios;
- Supervisão: módulo utilizado pelos Supervisores e Profissionais para liberação/reprovação de amostras e lotes, emissão de laudos, RNC e NCF;
- Estatístico: módulo que fornece informações estatísticas, tais como número de ensaios realizados, percentual de amostras reprovadas, número de RNC/NCF emitidos e tempo médio de duração de ensaios.

#### **3.3.1 O Fluxo de Análises**

O processo de análise de lotes (seja de lubrificantes, óleos básicos ou aditivos) está modelado pela máquina de estados apresentada na figura abaixo. Para a implementação atual do SILAB foram criados campos de *status* nas tabelas de amostra para controlar as transições da máquina de estados. O *status* NC, por exemplo, indica que a amostra está não conforme enquanto que o estado AP indica que a amostra está aprovada.



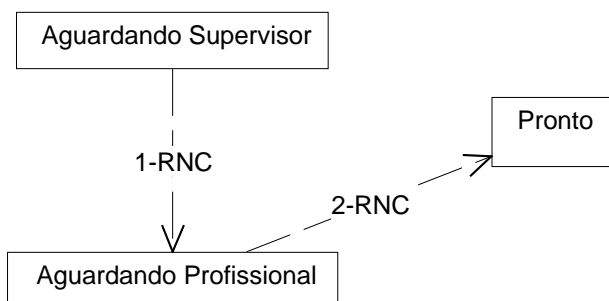
**Figura 5: Máquina de estados do fluxo de análises**

| Transição | Descrição   |
|-----------|---|
| 1         | Módulo de Pedidos disponibiliza a amostra para análise (caso este produto tenha ensaios cadastrados).   |
| 2         | Módulo de Pedidos verifica que o produto não tem nenhum ensaio cadastrado e coloca-o disponível ao supervisor para adicionar os ensaios.                    |
| 3         | Modulo Supervisão cadastra os ensaios e disponibiliza amostra.  |
| 4         | Módulo Técnico verifica que todos os ensaios da amostra foram realizados e retira esta da análise deixando-a disponível ao supervisor.                      |
| 5         | Modulo Supervisão cadastra novos ensaios para a amostra.  |
| 6         | Modulo Supervisão pede reamostragem.  |
| 7         | Modulo Supervisão aprova amostra (só se for amostragem e der tudo O.K.).  |
| 8         | Modulo Supervisão declara a amostra não conforme.   |
| 9         | Modulo Supervisão reprova amostra emitindo RNC.   |
| 10        | Modulo Supervisão aprova amostra com concessão técnica emitindo RNC.  |
| 11        | Modulo de Pedidos cadastra a reamostragem, mudando o estado da amostragem.  |
| 12        | Modulo Supervisão pede novos ensaios da amostragem.   |
| 13        | Modulo Técnico realiza ensaios solicitados.   |
| 14        | Modulo Supervisão tira amostra de análise.  |
| 15        | Modulo Supervisão declara amostra NC através da reamostragem, logo a reamostragem termina, e a amostra original ganha o <i>status</i> NC (caso amostragem). |

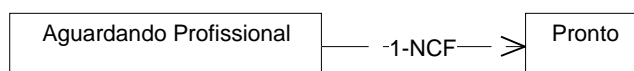
|    |   |
|----|---|
| 16 | Modulo Supervisão declara amostra AP através da reamostragem, logo a reamostragem termine, e a amostra original ganha o status AP (caso amostragem).    |
| 17 | Modulo Supervisão Permite que profissional corrija uma transição dada erroneamente na amostra, com isto são apagados qualquer laudo, RNC e NCF gerados. |

### 3.3.2 O Fluxo dos Relatórios (RNC e NCF)

A emissão de relatórios no sistema SILAB é um processo que envolve tanto profissionais quanto supervisores. Esse processo está especificado na máquina de estados da figura abaixo.



**Figura 6: Máquina de estados do fluxo de relatórios (RNC)**



**Figura 7: Máquina de estados do fluxo de relatórios (NCF)**

| Transição | Descrição   |
|-----------|---|
| 1-RNC     | Supervisor preenche RNC e envia ao profissional.                            |
| 2-RNC     | Profissional preenche RNC.  |
| RNC       | Profissional pode imprimir, atualizar todos os dados ou emitir RNC.         |
| Pronto    |   |
| 1-NCF     | Profissional preenche NCF.  |
| NCF       | Profissional pode imprimir, atualizar todos os dados ou emitir NCF por fax. |
| Pronto    |   |

### 3.3.3 Falhas na Implementação Atual

A implementação atual do SILAB é falha em alguns aspectos:

- Processos são *hardwired*: tanto o processo de análise quanto a emissão de relatórios estão implementados diretamente no código fonte do sistema. Isto gera dois problemas graves:
  1. Caso haja alguma alteração nesses processos o código do sistema tem que ser alterado e recompilado;

2. Tanto o código quanto a definição da base de dados do sistema se tornam extremamente complexos. Isso é gerado pelo controle dos campos de *status*, que são campos adicionais que tiveram que ser criados em várias tabelas, em diversos pontos do sistema.
- Estrutura organizacional *hardwired*: assim como no caso dos processos, caso haja alguma alteração na estrutura organizacional do laboratório o sistema deve ser alterado e recompilado;

### **3.4 Modelagem em PAGOde**

A modelagem do SILAB com o uso do ambiente PAGOde, apresentada a seguir, visa resolver os problemas encontrados na implementação atual do sistema.

#### **3.4.1 Modelagem da Estrutura Organizacional**

A modelagem da estrutura organizacional do laboratório em PAGOde é bastante simples. São criados cargos para cada uma das funções exercidas no laboratório. Note que com essa modelagem não seria mais necessária a criação de um cadastro de funcionários, simplificando a implementação do módulo de cadastros.

#### **3.4.2 Modelagem dos Artefatos**

Cada ensaio é modelado em PAGOde por um campo, e cada conjunto de ensaios que devem ser realizados por um determinado produto é modelado por um artefato. Essa modelagem também diminui dois cadastros do módulo de cadastros: o de ensaios e o de ensaios realizados por produto.

Os laudos e relatórios (RNC e NCF) também são modelados como artefatos.

A entrada de uma amostra no laboratório será feita pela edição de um artefato cujos campos são os dados da amostra. Alguns exemplos de campos são Tipo (se é óleo básico, lubrificante ou aditivo), Produto e Data/Hora de entrada no laboratório.

Além desses artefatos serão criados dois artefatos auxiliares que servirão para conter o parecer do supervisor ou do profissional em relação à amostra ou lote, respectivamente. Esses dois artefatos, “Parecer do Supervisor” e “Parecer do Profissional”, possuem dois campos, um chamado “Parecer” que conterá a informação sobre a aprovação ou reprovação da amostra/lote e outro chamado de “Explicações” que conterá uma explicação técnica sobre o porque da aprovação/reprovação da amostra.

#### **3.4.3 Reformulação dos Módulos do Sistema**

Com essa modelagem uma nova estruturação de módulos é requerida. Como PAGOde trabalha com ferramentas externas essa reformulação fará com que cada módulo seja enxergado pelo ambiente PAGOde como uma ferramenta.

O módulo de cadastro continua sendo necessário, porém ele é simplificado pois alguns cadastros, conforme visto anteriormente, não são mais necessários.

O módulo do técnico passa a funcionar como uma ferramenta para realização de ensaios. Ela auxilia o técnico durante a realização do ensaio e compara o resultado encontrado com os limites padrões;

As operações de supervisão, realizadas pelos supervisores e profissionais, passam a ser completamente modeladas pelo ambiente. Portanto o módulo de supervisão passa a não ser mais necessário.

O módulo estatístico pode ser completamente implementado com o uso de *advisors*, já que os *advisors* tem acesso aos objetos de métrica do sistema. Portanto para cada análise estatística necessária é criado um novo *advisor*.

### 3.4.4 Melhorias Apresentadas

Essa nova modelagem apresenta diversas melhorias em relação a implementação atual do SILAB:

- Processos e estrutura organizacional não são mais *hardwired*, permitindo que os processos sejam alterados sem que o sistema precise ser recompilado;
- Menos módulos precisam ser implementados (apenas parte do módulo de cadastro e o módulo do técnico);
- A implementação do módulo do técnico fica bastante simplificada já que não é mais necessário o controle de *status* de amostras/lotos;
- A definição da base de dados do sistema também fica bastante simplificada, já que várias tabelas passam a ser desnecessárias, facilitando assim a manutenção do sistema;

A desvantagem apresentada pela utilização de PAGoDE é o tempo de aprendizado da utilização deste ambiente, tanto pela equipe de desenvolvimento do sistema quanto pelos funcionários do laboratório.

## 4. Trabalhos Relacionados

O ambiente Articulator [25] baseia-se na descrição de processos através da definição de classes de objetos, que possuem atributos, relações, restrições, controle de fluxo, regras e métodos. A modelagem é baseada na captura de descrições informais do processo e na passagem dessas descrições para representações formais de processos e instâncias de processos. As grandes contribuições desse ambiente são o suporte visual à execução, capacidade de simulação e o poderoso sistema de análises de processo.

ProcessWeaver [8] é um modelo que utiliza Redes de Petri para a representação dos processos. Ele dá suporte a descrição de atividades em função dos seus recursos, entradas, saídas e parâmetros de controle. As transições da rede representam as atividades enquanto a existência de marcadores nos lugares habilitam sua execução. O ambiente de execução controla uma interface baseada em agendas, vista pelos membros da equipe do projeto. A partir dessa interface cada usuário pode executar suas tarefas com o auxílio de ferramentas externas ao ambiente. Os vários tipos de processo possuem representações visuais diferentes, proporcionando uma poderosa representação gráfica para o ambiente.

O ambiente Julia [29] é um ambiente integrado de controle de processos. A modelagem dos processos é feita com base na linguagem JIL (Julia Input Language), que serviu de base para a definição da linguagem do ambiente PAGoDE. Julia fornece vários serviços de suporte a execução de processos, que são baseados em um interpretador da linguagem e em agentes (incluindo agentes humanos). Durante a execução dos programas o interpretador se comunica com um gerenciador de objetos para solicitar a aquisição de artefatos e com um gerenciador de



recursos, para solicitar a aquisição de recursos. Os gerenciadores de objeto e de recurso, por sua vez, fazem uso de um gerenciador de *lock*, que é responsável pelo controle de concorrência. Os conflitos de aquisição de objetos e de recursos são resolvidos por um objeto, PCA (*Process Control Authority*), que pode necessitar de intervenção humana para tal.

Uma discussão de outros ambientes de engenharia de software orientados a processo pode ser vista em [15, 18].

## 5. Conclusões e Trabalhos Futuros

A partir da análise de estudos de caso realizados (outros estudos de caso são descritos detalhadamente em [12]) e com base na avaliação dos ambientes orientados a processo existentes algumas conclusões podem ser tiradas a respeito do ambiente apresentado neste trabalho:

- A separação dos tipos processo em batch e de usuário facilita bastante a modelagem do processo. Essa característica não é apresentada de forma explícita em nenhum dos ambientes estudados;
- Processos compostos também facilitam bastante a modelagem e possibilitam que processos possam ser reutilizados para compor processos mais complexos;
- O suporte à duas representações de processo, formal e informal, integradas é de fundamental importância para a utilização efetiva do ambiente por parte dos operadores. Os ambientes estudados também não apresentam essa característica;
- O suporte visual ao acompanhamento dos processos (*trace*), encontrado apenas no ambiente Articulator [25], facilita bastante o gerenciamento dos projetos, permitindo que os engenheiros de negócio tenham controle total do andamento de cada projeto sob sua responsabilidade;
- *Project Advisors* se mostraram de fundamental importância para a modelagem dos processos, conforme demonstrado nos estudos de caso apresentados;
- A introdução do conceito de padrões de processo (ainda não encontrado na literatura) é importante tanto sob aspectos teóricos quanto no auxílio ao trabalho do engenheiro de negócios;
- A utilização de WWW para o desenvolvimento da interface dos operadores torna a utilização do sistema bastante flexível, permitindo que operadores trabalhem espalhados em uma WAN;
- O uso de WWW também é importante na padronização da interface do ambiente, diminuindo assim a curva de aprendizado por parte dos operadores;
- Uma característica interessante da utilização do ambiente é o fato do módulo de edição (engenheiro de negócios) poder ser cadastrado no ambiente como uma ferramenta, permitindo assim que se modelem processos que possuam etapas (subprocessos) relativas à modelagem de processos;
- Em relação à modelagem da implementação atual do ambiente, a utilização do relacionamento de visualização (*Views a*) [3], tornou bastante clara a estrutura do modelo de execução.

Um ponto que pode ser considerado deficiente na versão atual do ambiente é a dificuldade de modelagem de processos por engenheiros de negócio que não sejam programadores. Na próxima versão serão desenvolvidos assistentes (*wizards*) com o intuito de facilitar essa tarefa.

Uma outra deficiência do ambiente é a falta de suporte à outras linguagens de representação de processo. Para diferentes utilizações do ambiente diferentes linguagens de representação são necessárias. Um exemplo seria a utilização do ambiente para *Process Scheduling* [26], onde uma representação da linguagem de processos baseada em diagramas PERT seria mais adequada. O *framework* para construção de *Process Life-Cycle Applications* apresentado em [2] visa solucionar esse problema.

A seguir é apresentada uma lista de trabalhos para melhoria e complementação do ambiente PAGoDE. Parte desses trabalhos já esta sendo desenvolvida em projetos do Laboratório de Engenharia de Software (LES) da PUC-Rio.

- Desenvolvimento de ferramentas para análise e otimização de processos. Essa ferramenta seria responsável por vários tipos de análises tais como *what-if*, checagem de inconsistências e checagem de *deadlocks*;
- Desenvolvimento de uma camada de suporte à *EDI (Electronic Data Interchange)* [23], permitindo que processos inter-organizacionais sejam realizados de acordo com esse padrão;
- Modelagem e desenvolvimento da camada de controle e gerenciamento de processos do ambiente de engenharia de software TOTEM [28], atualmente em desenvolvimento no LES.

Existem também algumas extensões teóricas deste trabalho:

- Modelagem de um *framework* para construção de ambientes orientados a processos com o uso de *Viewpoints* e *Design Patterns* [2];
- Estudos para estabelecimentos de padrões de processo (*process patterns*) na área de engenharia de software;
- Realização de estudos de caso para avaliação de métricas com a utilização do ambiente;
- Formalização da semântica da linguagem de processos.

## 6. Referências

- [1] Alencar, P. S. C., Cowan, D. D., Lucena, C. J. P., “Integrating Design Patterns an Subject-Oriented Programming within the ADV Framework”, *Submitted to the 18th International Conference on Software Engineering, ICSE’95*, 1995.
- [2] Alencar, P. S. C., Cowan, D. D., Fontoura, M. F. M. C., Lucena, C. J. P., Nelson, T.: “A Framework Development Approach based on Viewpoints”, *Technical Report*, Computer Systems Group, University of Waterloo, Ca, 1997.
- [3] Alencar, P. S. C., Cowan, D. D., Nova, L. C. M., Lucena, C. J. P., “Interface Modeling with ADVs” , *Technical Report*, Computer Systems Group, University of

Waterloo, Ca, 1997.

- [4] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S., “*A Pattern Language*”, Oxford University Press, New York, 1977.
- [5] Bandinelli, S., Fuggetta, A., Grigolli, S.: “Process modeling in-the-large with SLANG”, *In Proc. of the Second International Conference on Software Engeneering*, pp. 144-154, 1993.
- [6] Barthelmess, P. & Wainer, J.: “Workflow Modeling”, *CYTED-RITOS International Workshop on Groupware*, Lisbon, Portugal, Setembro 1996
- [7] Booch, G.: “*Object-Oriented Analisis and Design with Applications*”, The Benjamin/Cummings Publishing Company, Inc., segunda edição, 1994.
- [8] Bourdon, M.: “Process Weaver: Process Modeling Experience Report”, *Technical Report*, CAP Gemini Innovation, 1992.
- [9] Bruynooghe, R. F., Parker, J. M., Rowles, J. S.: “PSS: A system for Process Enactment”, *Proceedings of the First International Conference on Software Process*, Los Angeles, California, USA, IEEE Computer Society Press, Outubro 1991.
- [10] Deiters, W. & Gruhn, V.: “Managing software process in the enviroment melmac”, *In Proc. of the Fourth ACM SIGSOFT Symposium on Practical Software Development Enviroments*, pp. 193-205, 1990. Irvine, California.
- [11] Fiorini, S. T.: “Processos de Negócio e Hipertextos: Uma Proposta para Elicitacao de Requisitos”, *Dissertação de Mestrado*, Departamento de Informática, PUC-Rio, Rio de Janeiro, 1995.
- [12] Fontoura, M. F.: “Um Ambiente para Modelagem e Execução de Processos”, *Dissertação de Mestrado*, Departamento de Informática, PUC-Rio, Rio de Janeiro, 1997.
- [13] Gamma, E., Helm, R., Vlissides, J., Johnson, R. E., “Design Patterns: Abstraction and Reuse of Object-Oriented Design”, *Proceedings ECOOP*, ed. O. Nierstrasz, LNCS, Springer-Verlag, Kaiserslautern, Germany, 1993.

- [14] Gamma, E., Helm, R., Johnson, R. E., Vlissides, J.: “*Design Patterns, Elements of Reusable Object-Oriented Software*”, Addison-Wesley 1995.
- [15] Gimenes, I. M. S.: “Uma Introdução ao Processo de Engenharia de Software”, *Relatório Técnico*, Universidade Estadual de Maringá, Paraná, Brasil, Julho 1994.
- [16] Hubert, L.: “An Environment for Software Process Modeling integrated with Project Management and Product Management facilities”, *First European Workshop on SPM*, Milan, Italy, May 1991.
- [17] Johnson, R. E., “Documenting Frameworks using Patterns”, *Proceedings OOPSLA*, ACM SIGPLAN Notices, 27(10) pp. 63-76, 1992.
- [18] Karrer, A. S. & Scacchi, W.: “Meta-Environments for Software Production”, *Technical Report*, Information and Operation Management Department, University of Southern California, 1994.
- [19] Merlin, P. M. & Farber, D. J.: “Recoverability of Communications”, *IEEE Transactions on Communications*, Vol. COM-24, No. 9, pp. 1036-1043, Setembro 1976
- [20] Merlin, P. M.: “A Methodology for the Design and Implementation of Communication Protocols”, *IEEE Transactions on Communications*, Vol. COM-24, No. 6, pp. 614-621, Junho 1976.
- [21] Moura, L.: “Um Sistema de Programação Visual”, *Dissertação de Mestrado*, Departamento de Informática, PUC-Rio, Rio de Janeiro, 1995.
- [22] Peterson, J. L.: “*Petri Net Theory and the Modeling of Systems*”, Prentice-Hall, N.J., 1981.
- [23] Premenos Technology Corporation: <http://www.premenos.com>
- [24] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorenson, W.: “*Object-Oriented Modeling and Design*”, Prentice Hall, Englewood Cliffs, NJ, 1991.
- [25] Scacchi, W.: “Modeling, Simulating and Enacting Complex Organizational Processes: A Lyfe Cycle Approach”, *Technical Report*, Information and Operation

Management Department, University of Southern California, 1996.

- [26] Schlenoff, C., Knutilla, A., Ray, S.: “Unified Process Specification Language: Requirements for Modeling Process”, U. S. Department of Commerce, Technology Administration, National Institute of Standards and Technology (NIST), Manufacturing Engineering Laboratory, Manufacturing Systems Integration Division, Gaithersburg, MD 20899, Setembro 1996.
  
- [27] Soares, L. F. G., Casanova, M. A., Rodriguez, N. R.: “Nested Composite Nodes and Version Control in an Open Hypermedia System”, *International Journal on Information Systems; Special issue on Multimedia Information Systems*, vol 20, No. 6. Elsevier Science Ltd. England, pp.501-520, Setembro de 1995.
  
- [28] Staa, A. & Cowan D.: “An Overview of TOTEM Software Engineering Meta-Environment”, *Monografias em Ciência da Computação*, Departamento de Informática, PUC-Rio, Novembro 1995.
  
- [29] Sutton, S. M. & Osterweil, L.: “The Design of a Next-Generation Process-Language”, *CMPSCI Technical Report 96-30*, University of Massachusetts at Amherst, Computer Science Department, Amherst, Massachusetts 01003, Janeiro 1997 .
  
- [30] Willrich, R., Senac, P., Sannes, P. D., Diaz, M.: “Towards Hypermedia Documents Design”, *14º Simpósio Brasileiro de Redes de Computadores, SBRC’ 96*, Fortaleza, Ceara, Brasil, 1996.