Gerenciamento do Processo de Desenvolvimento Cooperativo de Software no Ambiente PROSOFT[1]

Carla Alessandra G. de Lima[*,#] *Rodrigo Quites Reis*[*,#] *Daltro José Nunes*[#]

[*]Universidade Federal do Pará - UFPA
Departamento de Informática
Belém - PA

[#]Universidade Federal do Rio Grande do Sul
Programa de Pós-Graduação em Computação
Instituto de Informática
Porto Alegre - RS

E-mail: {clima, quites, daltro}@inf.ufrgs.br

Resumo

A capacidade de gerenciar processos de desenvolvimento de software com envolvimento cooperativo dos desenvolvedores constitui um aspecto importante na construção de ambientes de desenvolvimento de software. Este artigo apresenta extensões ao ambiente PROSOFT para o desenvolvimento cooperativo e gerência de processos de software. O ambiente PROSOFT apóia a construção formal de ferramentas de software sob o paradigma algébrico, proporcionando sua integração ao ambiente.

O gerenciador de processos e o PROSOFT Cooperativo são apresentados, enfatizando a coordenação de desenvolvedores, grupos de desenvolvedores e gerentes em um ambiente distribuído, bem como características de gerência de projetos, reutilização de processos de software e gerência de versões de objetos cooperativos. A máquina de processos de software que compõe o gerenciador coordena processos modelados e instanciados permitindo modificação dinâmica dos mesmos, dentre outras características, e utiliza serviços fornecidos pelo componente PROSOFT Cooperativo. Através desta especificação, é possível visualizar de forma integrada a execução de processos de software, a interação entre desenvolvedores e ferramentas do ambiente e a cooperação entre desenvolvedores atuando em processos de desenvolvimento.

Palavras-chave: ambiente de desenvolvimento de software, processo de software, desenvolvimento cooperativo de software e especificação formal.

Abstract

Managing software development process involving developers in a cooperative way is an important aspect for software engineering environment construction. This paper shows extensions for PROSOFT environment to support cooperative development and software process management. The PROSOFT provides integration of tools specified with the use of an algebraic paradigm.

The Process Manager and Cooperative PROSOFT are presented, emphasizing developers, groups and managers' coordination in a distributed environment. Project management, software process reuse and cooperative object version management characteristics are also described. The process manager uses Cooperative PROSOFT services and has a process machine which enacts modeled and instantiated processes, providing support for dynamic process modification. Through this specification the software process enaction, the interaction between tools and developers, and developers' cooperation during the software process can be visualized in an integrated way.

Keywords: software development environment, software process, cooperative software development, formal specification.

[1] Trabalho apoiado pela CAPES, CNPq e FAPERGS.

1. Introdução

A busca da qualidade do processo de construção de software vem absorvendo bastante interesse na área de Engenharia de Software. A crescente utilização de software determina que esta qualidade seja uma meta crucial da área. Entretanto, os desenvolvedores de software enfrentam vários problemas de natureza técnica e gerencial. Tais problemas englobam características inerentes de um processo de construção de software como por exemplo, a complexidade dos produtos de software, as dificuldades no estabelecimento e estabilização dos requisitos, as dificuldades de manutenção de software e as dificuldades na comunicação entre os desenvolvedores.

O processo de desenvolvimento de software, ou processo de software [GIM94], corresponde ao conjunto de atividades relacionadas que são desempenhadas pelos desenvolvedores desde a concepção até a liberação do produto. Uma forma de analisar e amadurecer tal processo é através da sua descrição, a qual consiste de um modelo de processo de software. A descrição formal de um processo de software é a atividade que permite que o mesmo seja analisado, compreendido e automatizado (executado).

Um Ambiente de Desenvolvimento de Software (ADS) deve proporcionar um alto nível de integração entre suas ferramentas e ser capaz de executar um modelo de processo de software, através da coordenação dos desenvolvedores na execução de suas tarefas, permitindo coleta de métricas, execução automática de algumas atividades e mudança do processo durante sua execução.

Os ADS que suportam a modelagem e execução de processos de software são denominados ADS orientados ao processo. Nestes ambientes, um modelo de processo é interpretado por um mecanismo de execução, o qual interage com os desenvolvedores e com as ferramentas do ambiente a fim de controlar o processo de software. Existem diversos ADS orientados ao processo, como por exemplo EPOS [CON94], Spade [BAN92] e Adele [BEL94].

A crescente complexidade do software desenvolvido atualmente teve como consequência o aumento do número de profissionais envolvidos no processo de software. Isto ocorre porque este processo possui atividades criativas, desempenhadas por pessoas com preferências, formações e experiências diferenciadas que precisam cooperar com um objetivo comum: o desenvolvimento de um produto ou artefato de software [BOR95]. O apoio ao desenvolvimento cooperativo de software corresponde à disponibilização de técnicas e ferramentas que forneçam assistência a grupos de desenvolvedores de software nas tarefas que são realizadas cooperativamente.

A literatura descreve vários ADS orientados a processo, dando ênfase à sua linguagem de modelagem e ao seu mecanismo de execução. Da mesma forma são apresentadas várias iniciativas de incorporar recursos que permitam cooperação de desenvolvedores em ADS. Entretanto, nem sempre estes componentes são tratados formalmente e nem sempre estão presentes no mesmo ambiente. A maioria apresenta apenas uma descrição informal do mecanismo de execução e do mecanismo que propicia a cooperação.

Em função dos benefícios do uso de métodos formais [COH86] e da necessidade de construção de mecanismos de execução e cooperação para ADS orientados ao processo, o objetivo deste artigo é apresentar extensões realizadas no ADS PROSOFT para permitir desenvolvimento cooperativo de software (PROSOFT Cooperativo), bem como gerência de processos de software modelados no ambiente (Gerenciador de Processos para o PROSOFT). Estas extensões foram baseadas em um estudo aprofundado dos requisitos necessários e estão direcionadas a aumentar a qualidade do processo de desenvolvimento de software no referido ambiente. São levados em consideração aspectos de distribuição do ambiente, gerência de versões de objetos cooperativos, gerência e autorização de acesso a objetos, manipulação síncrona de objetos, gerência de projetos e adaptação dinâmica do processo de software.

O artigo é organizado como segue. A seção 2 apresenta o ambiente PROSOFT, sua arquitetura, tipos de dados e distribuição. A seção 3 apresenta o PROSOFT Cooperativo. A seção 4 apresenta o gerenciador de processos de software para o PROSOFT. A seção 5 apresenta alguns trabalhos relacionados e comparações com outros ambientes. Finalmente, na seção 6 são apresentadas as conclusões.

2. Ambiente PROSOFT

O PROSOFT tem como objetivo principal apoiar o desenvolvimento formal de software, fornecendo integração de dados, controle e de apresentação entre suas ferramentas. Sua construção foi influenciada pela estratégia *data-driven* [NUN92], conceito de modelos [BJØ78], tipos abstratos de dados, orientação a objetos (OO), método algébrico [WAT91], dentre outros conceitos.

Os componentes do PROSOFT são os ATOs - Ambientes de Tratamento de Objetos. Todo ATO especifica algebricamente um tipo abstrato de dados e é composto, essencialmente, de uma classe e de um conjunto de operações que atuam sobre os objetos dessa classe. Além disso, a ICS - Interface de Comunicação de Sistema, provê o mecanismo básico de comunicação dos ATOs PROSOFT. O desenvolvimento de um sistema, sob a ótica PROSOFT, consiste na definição de um ou mais ATOs. A estrutura de um ATO é ilustrada na figura 1.

Todo ATO possui um nome, e na sua definição são especificados a sua **classe** (instanciação), representada graficamente e definida através da composição de tipos de dados, a **interface** das operações e a especificação (semântica) das **operações** [NUN94]. Cada ATO trata apenas termos do *sort* (tipo) por ele definido. Os ATOs comunicam-se através de mensagens. A troca de mensagens entre ATOs é feita através da ICS, cujo formato é apresentado a seguir:

ICS(nome do ATO, nome da operação, seletor, <lista-de-argumentos>)

Figura 1: Estrutura de um ATO

Em uma chamada ICS, o seletor é um objeto do ATO chamado que será transformado pela operação definida na sua interface. A figura 2 apresenta um modelo esquemático da comunicação dos ATOs através da ICS [NUN94].

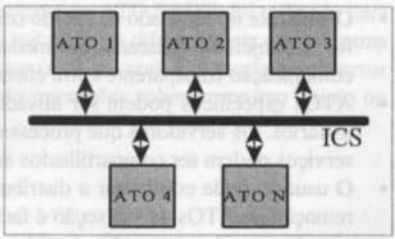

Figura 2: Estrutura do PROSOFT

As instâncias de uma classe (objetos) são entidades passivas, que não têm capacidade de responder a estímulos (mensagens). Armazenam dados mas não podem manipulá-los. Deve-se observar que o PROSOFT é um ambiente homogêneo (fechado). Portanto, o ambiente reconhece somente os componentes (ATOs) desenvolvidos sob o seu paradigma.

2.1. Tipos de dados PROSOFT

Os tipos de dados presentes no PROSOFT estão classificados em: **primitivos** (Integer, String, Real, Char, Boolean, Date, Time); **compostos** (construtores definidos no método algébrico: conjunto, lista, mapeamento, registro e união disjunta) e **definidos pelo usuário** (construídos a partir de outros tipos). A seguir, serão apresentados os tipos compostos, bem como sua representação diagramática na figura 3 (compiladas em [MOR97] e [NUN92]).

- **Tipo Conjunto**: Representa uma coleção de objetos obedecendo às características de conjuntos;
- **Tipo Mapeamento:** Expressa uma função entre objetos de um Tipo-Domínio com os objetos de um Tipo-Imagem;
- **Tipo Lista**: Representa uma sequência (ordenada) de zero ou mais componentes;
- **Tipo Registro**: Define tuplas de tipos heterogêneos e é usado para expressar o produto cartesiano de objetos;
- **Tipo União Disjunta**: Define a união de tipos diferentes de elementos. Útil na definição de classes de dados que necessitem expressar valores alternativos.

Figura 3: Exemplos da instanciação de ATOs PROSOFT [REI98]

2.2. PROSOFT Distribuído

Para que vários usuários utilizem o ambiente PROSOFT simultaneamente, através de mecanismos de compartilhamento das informações, foi desenvolvido o PROSOFT Distribuído, apresentado em [SCH95]. De uma forma geral, as características que distinguem o PROSOFT Distribuído do PROSOFT tradicional são:

- O ambiente foi projetado de acordo com um modelo cliente/servidor. A implementação da ICS foi reprojetada (baseada no mecanismo *Remote Procedure Call*) para permitir uma comunicação transparente entre clientes e servidores;
- ATOs específicos podem ser ativados por demanda, de acordo com as necessidades dos usuários. Os servidores que processam as requisições são chamados de ATO-Server e seus serviços podem ser compartilhados simultaneamente por vários usuários;
- O usuário pode configurar a distribuição do sistema de forma flexível, isto é, a adição ou remoção de ATOs da sua seção é facilitada.

Apesar das inovações trazidas para o ambiente, algumas características ainda eram necessárias para considerá-lo um ADS cooperativo. Estas características são apresentadas na seção a seguir.

3. PROSOFT Cooperativo

O principal objetivo de um ADS cooperativo envolve o uso concorrente de objetos de software (diagramas, códigos, etc.), permitindo que um mesmo objeto seja manipulado por vários desenvolvedores. O PROSOFT Distribuído representa um primeiro passo para esse objetivo, provendo mecanismos para compartilhamento de informações e distribuição dos ATOs. Entretanto, a manipulação síncrona de objetos não estava disponível.

PROSOFT Cooperativo é uma extensão ao núcleo do PROSOFT especificada com o formalismo apresentado em [NUN94] na forma de ATOs desenvolvidos e integrados ao ambiente. Em [REI97] e [REI98] são apresentadas as visões estática, funcional e dinâmica dessa extensão, sendo que esta última foi especificada através de ISO/LOTOS (*Language of Temporal Ordering Specification*) e foi influenciada por We-Met [REK92].

Do ponto de vista arquitetural, o PROSOFT Cooperativo é um componente do PROSOFT Distribuído que provê um meta-ambiente cooperativo que gerencia usuários, estações de trabalho e objetos cooperativos. A arquitetura do PROSOFT Cooperativo é apresentada a seguir através da descrição informal da sua funcionalidade.

3.1. Arquitetura do PROSOFT Cooperativo

O PROSOFT Cooperativo constitui um conjunto de 8 ATOs especificados para gerenciar a manipulação síncrona de objetos cooperativos do PROSOFT, permitindo a construção de aplicações *groupware* para o ambiente. Trata-se de uma *framework middleware* que também provê:

- Um modelo de versões de objetos ortogonal (baseado em [GOL96]);
- Facilidades de *Undo/Redo* de operações sobre objetos cooperativos;
- Gerência de usuários, estações de trabalho e ferramentas do ambiente;
- Direitos de acesso aos objetos de baixa granulosidade;
- Uma técnica para evolução de ATOs convencionais para suportar a manipulação cooperativa síncrona de objetos.

O PROSOFT Cooperativo gerencia a comunicação entre os usuários e os ATOs-*Groupware*[2] disponíveis no ambiente. A figura 4 mostra um diagrama que explica a estratégia adotada. Um usuário solicita uma operação através da ICS (chamada "inclui_atividade") para modificar um objeto (chamado "nome-obj-coop"). Toda essa informação é enviada aos ATOs do PROSOFT Cooperativo aonde é verificado se o usuário possui direito de acesso ao objeto e é feita uma nova chamada ICS para o ATO associado ao objeto (neste exemplo, SADT[3]). Além disso, o último argumento enviado ao ATO destino é o identificador do usuário que solicitou a operação. Esta informação adicional pode ser utilizada para identificar visualmente o usuário que realizou uma operação sobre o objeto compartilhado e, especificamente no ATO SADT, foi utilizada para identificar com cores diferenciadas os objetos adicionados por usuários diferentes em um diagrama SADT compartilhado (interação WYSIWIS - *what you see is what I see*). É importante observar que vários usuários distribuídos podem estar solicitando operações sobre o mesmo objeto ou outros objetos.

[2] Terminologia adotada em [REI98] para identificar ATOs que permitem a manipulação cooperativa síncrona de objeto

[3] O ATO SADT é responsável pela criação e execução de diagramas SADT [ROS85].

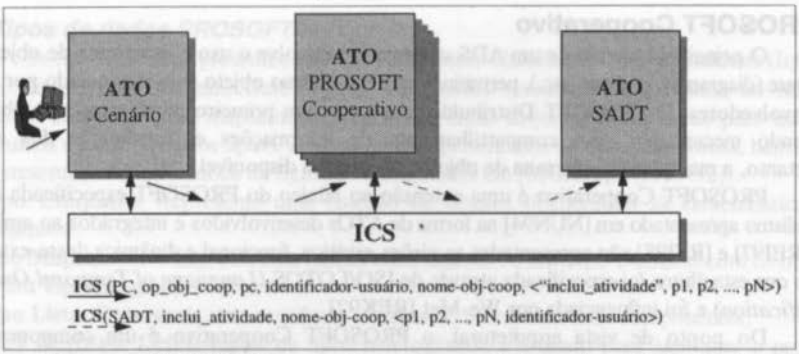

Figura 4: PROSOFT Cooperativo como mediador entre solicitações de usuários e o ATO destino [REI98]

3.2. Gerência de Usuários e Estados dos Objetos

A abordagem do PROSOFT Cooperativo classifica os usuários de acordo com as tarefas genéricas que podem desempenhar durante a evolução do ciclo de vida de software. Um **super-usuário** é definido quando da criação de um ambiente cooperativo, e tem acesso irrestrito a todas as informações do ambiente. Existem também os usuários **gerentes**, que atuam nos grupos e **usuários comuns,** que podem pertencer a grupos. Todas as atividades realizadas sobre os usuários são armazenadas, permitindo que sejam consultadas posteriormente, identificando qual usuário realizou uma atividade e em qual instante.

Todos os objetos do ambiente são associados com um conjunto de usuários e grupos que possuem diferentes direitos de acesso. A união dos espaços de trabalho público e dos usuários constitui o **Espaço de Trabalho Compartilhado**. Uma classificação de usuários foi definida baseada no papel desempenhado por estes no modelo de cooperação:

- Usuário **Criador** é aquele que criou o objeto;
- Usuário **Editor** é aquele que pode modificar o objeto, ou ainda desfazer operações feitas por ele próprio sobre o objeto;
- Usuário **Visualizador**, que pode acompanhar uma sessão de uso cooperativo (mas não pode atuar sobre o objeto);
- Usuário **Gerente**, que pode agir como moderador na sessão cooperativa do objeto, e, por exemplo, pode modificar o estado do objeto, especificar um direito de acesso para um usuário ou desfazer operações realizadas por algum usuário no objeto.

Estados foram definidos para os objetos (ou versões) refletindo o grau de maturidade atingido pelos mesmos. Um objeto é criado inicialmente no estado **instável**, onde apenas o seu criador tem direito de modificá-lo. A partir de uma solicitação do criador, o objeto pode ser promovido ao estado **estável**, permitindo que vários usuários trabalhem de forma cooperativa. Através de uma solicitação de um gerente, o objeto torna-se **consolidado**, onde nenhuma modificação posterior é permitida (objetos consolidados podem possuir versões derivadas).

Esta abordagem permite que o gerenciador do processo de software atue como super-usuário do ambiente cooperativo, gerenciando automaticamente os direitos de acesso dos desenvolvedores de acordo com as atividades do processo e permitindo que vários desenvolvedores atuem sobre o mesmo objeto cooperativo para concluir uma atividade.

3.3. ATOs do PROSOFT Cooperativo

Alguns ATOs foram construídos para especificar o PROSOFT Cooperativo. O ATO PROSOFT Cooperativo, principal ATO desta especificação, integra os diversos componentes e fornece as operações básicas do ambiente coperativo para os outros ATOs do PROSOFT. Por este motivo, a maioria das suas operações são chamadas via ICS aos seus ATOs componentes. A figura 5 mostra a instanciação do ATO PROSOFT Cooperativo.

As folhas da instanciação de um ATO são tipos primitivos ou definidos pelo desenvolvedor. No caso do ATO PROSOFT Cooperativo, as folhas são tipos definidos, ou seja, outros ATOs participam da especificação e são chamados através da ICS.

O ATO Objeto Cooperativo (OC) gerencia o armazenamento dos objetos que compõem o Espaço de Trabalho Compartilhado (A classe do ATO Espaço de Trabalho Compartilhado não é mostrada neste artigo mas possui o ATO OC como componente). Para cada objeto PROSOFT são armazenadas as seguintes informações: o registro da **última alteração** que o objeto foi submetido; o **nome** do objeto; o **estado** em que o objeto se encontra. No caso de objetos no estado Estável, há o controle das permissões de acesso para os usuários e grupos; o **nome do objeto ancestral** (se houver); uma lista ordenada das **versões derivadas**; um **histórico** das últimas operações realizadas sobre o objeto (para permitir o *undo*); informações acerca do **ATO** ao qual o objeto pertence (para saber se é um ATO-*Groupware)*; e uma **descrição textual** para o objeto. A figura 6 apresenta a instanciação da classe deste ATO.

Figura 5: Classe do ATO PROSOFT Cooperativo [REI98].

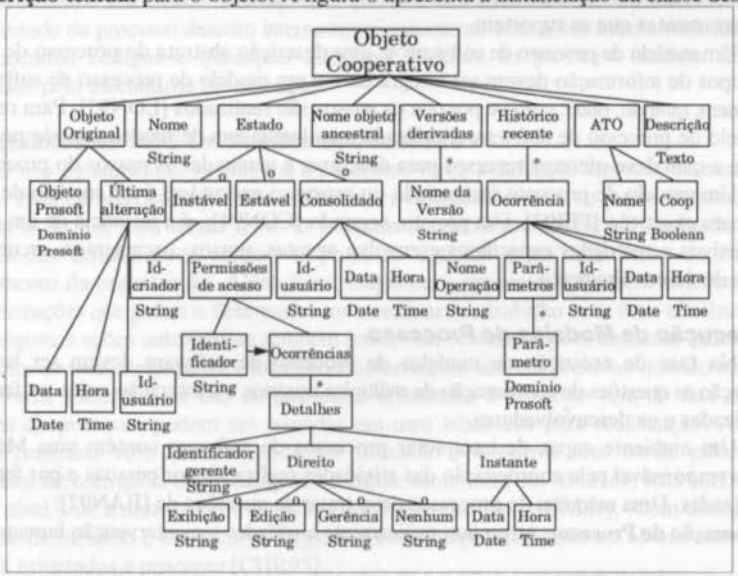

Figura 6: Instanciação do ATO ObjetoCooperativo [REI98].

As operações algébricas sobre os objetos desses e dos outros ATOs do PROSOFT Cooperativo não serão mostradas por questão de espaço. Na figura 7 são mostradas algumas primitivas para manipulação de objetos no ATO Espaço de Trabalho Compartilhado. A semântica de todas as operações do ambiente está em [REI98].

cria_obj	nomes_obj_modificados	acessa_obj
nome_ancestral_obj	data_último_acesso_obj	libera_obj
estado_obj	hora_último_acesso_obj	cria_versão_obj
existe_obj	data_última_alteração_obj	undo_obj
nome_versão_mais_recente_obj	hora_última_alteração_obj	torna_estável_obj
nomes_versões_obj	usuário_tem_permissão_acesso	torna_consolidado_obj
nomes_obj_coop_criador	direito_acesso_obj	inclui_permissão_acesso_obj
id_usuários_ativos_obj	é_gerente_obj	altera_permissão_acesso_obj
id_usuários_ativos_obj_instante	nomes_obj_sendo_utilizados	exclui_permissão_acesso_obj
id_criador_obj	quantidade_usuários_ativos_obj	altera_descrição_obj
id_gerentes_obj	usuário_está_ativo_obj	aplica_operações_obj
descrição_obj	op_obj_coop	

Figura 7: Primitivas de manipulação de objetos no ATO Espaço de Trabalho Compartilhado.

4. Gerenciador de Processos de Software para o PROSOFT

Esta seção apresenta a terminologia utilizada e o gerenciador de processos de software construído para o PROSOFT.

4.1. Processo de Software

O processo de produção de software envolve atividades complexas desempenhadas por pessoas (agentes) com as mais diversas capacidades. Neste sentido, a modelagem e a execução de processos de software são de fundamental importância para o aumento da qualidade do produto de software e têm sido estudadas pela comunidade de Engenharia de Software com o intuito de prover ferramentas que as suportem.

Um modelo de processo de software[4] é uma descrição abstrata do processo de software. Vários tipos de informação devem ser integradas em um modelo de processo de software para indicar quem, quando, onde, como e por que os passos são realizados [LON93]. Para representar um modelo de processo de software é utilizada uma linguagem de modelagem do processo de software, a qual deve oferecer recursos para descrever e manipular os passos do processo.

Um modelo de processo instanciado ou processo executável é um modelo de processo pronto para execução [FEI93]. Um projeto, segundo [CON94], é a instância de um processo, com objetivos e restrições específicos e envolve agentes, prazos, orçamentos, recursos e um processo de desenvolvimento.

4.2. Execução de Modelos de Processo

Na fase de execução de modelos de processo de software devem ser levadas em consideração as questões de coordenação de múltiplos usuários e a interação entre as ferramentas automatizadas e os desenvolvedores.

Um ambiente capaz de interpretar processos de software contém uma Máquina de Processo responsável pela coordenação das atividades realizadas por pessoas e por ferramentas automatizadas. Uma máquina de processos deve tratar as questões de [BAN92]:

- **Automação de Processo**: Ativar automaticamente atividades sem intervenção humana, através

[4] Alguns autores (citados em [CON94]) usam este termo para denotar a estrutura que representa processos de software, ou seja, o esquema do processo (tipos, regras, etc.) Aqui, esta estrutura está nos ATOs construídos para o gerenciador.

de uma integração com as ferramentas do ambiente;

- **Trabalho Cooperativo**: Suporte à coordenação e cooperação de pessoas trabalhando em um projeto de software;
- **Monitoração**: Prover diferentes visões da execução do processo, permitindo que o gerente de projetos obtenha informações sobre o andamento correto das atividades;
- **Registro da história do processo**: Coletar dados da evolução do processo para permitir que o processo melhore onde houver necessidade, e seja corrigido para atender novos requisitos.

Para que as questões acima sejam tratadas, alguns requisitos de execução devem ser satisfeitos pelo mecanismo de execução e pelo ambiente de desenvolvimento que suporta o processo [FRO94]:

- **Fluxo de Controle**: a sequência de atividades no processo deve ser representada e obedecida;
- **Iteração**: algumas atividades podem necessitar de repetição e a ativação automática dessas atividades é tarefa do mecanismo de execução;
- **Interação humana**: a interface com os desenvolvedores deve ser adequada e permitir que o *feedback* necessário seja informado;
- **Gerência da Informação**: o armazenamento persistente de todos os dados envolvidos no processo, com controle de acesso e gerência de versões;
- **Métricas**: a coleção automática de dados sobre o processo e o produto;
- **Adaptação dinâmica**: a mudança do processo durante sua execução;
- **Execução de ferramentas**: suporte à interação com vários tipos de ferramentas do ambiente;
- **Restrições e alocação de recursos**: quem está trabalhando com o que e que recursos são necessários para quais atividades.

Durante a execução de processos de software são manipuladas informações sobre o próprio processo e seu estado. Isto significa que uma das tarefas do mecanismo de execução é manter o estado do processo descrito internamente consistente com o seu estado real. Além disso, os documentos, códigos e quaisquer informações sobre os produtos também devem ser manipulados pelo mecanismo de execução.

4.3. Requisitos de ADS Orientados ao Processo

O uso de ambientes orientados a processos força a definição rigorosa da execução do processo. Esta característica permite melhorar a comunicação entre as pessoas envolvidas e a consistência do que está sendo feito. O treinamento de novos usuários é facilitado pois um desenvolvedor novato pode estudar o processo definido e ter uma visão de como é o funcionamento da organização. Além disso, enquanto o processo executa, o ambiente sempre provê informações que guiam o desenvolvedor a realizar seu trabalho com mais eficiência. Neste sentido, algumas ações automáticas também podem ser realizadas pelo ambiente "liberando" os seus usuários de tarefas repetitivas e fornecendo informações sobre o processo quando necessário.

Outra vantagem do uso de ambientes orientados a processo vem do fato de que as definições de processo podem ser reunidas em uma biblioteca para reutilização. Assim, um processo realizado com sucesso pode ser acessado por outros sem muito esforço. Esta característica faz com que a organização não somente economize em recursos, mas também possa atingir o nível 3 de maturidade do modelo CMM, descrito em [HUM89]. Além disso, a coleta automática de métricas é um recurso de grande contribuição que pode ser obtido com o uso de ambientes orientados a processo [CHR95].

Alguns requisitos são considerados indispensáveis para o apoio ao desenvolvimento de software com as vantagens citadas acima e estão presentes na maioria dos ADS orientados a

processo reais. São eles:

- **Suporte a múltiplos usuários**: mecanismos para atender vários usuários ao mesmo tempo requisitando serviços do ambiente;
- **Gerência de objetos**: controle do acesso e da evolução de objetos compartilhados. As versões dos documentos e produtos de software devem ser gerenciadas para permitir cooperação e consistência;
- **Gerência de comunicação entre pessoas**: as pessoas que estarão envolvidas no desenvolvimento de software devem ter acesso a mecanismos de comunicação, tais como mensagens eletrônicas e conferência eletrônica;
- **Gerência de cooperação**: a edição cooperativa de documentos e itens de software deve ser gerenciada pelo ambiente de forma que os usuários envolvidos obtenham comunicação síncrona sobre os produtos que estão manipulando;
- **Gerência de Processo**: o suporte à modelagem e execução do processo de software. Dentro deste requisito, encontra-se a necessidade de um formalismo de modelagem de processo e uma máquina de execução das definições de processo;
- **Extensibilidade**: permitir extensão do ambiente através da inclusão de novas ferramentas, sejam elas de apoio ao desenvolvimento de software ou com outras funções;
- **Integração entre todos os módulos**: todos os níveis de integração devem estar disponíveis para viabilizar os outros requisitos.

Os requisitos de extensibilidade e integração entre os módulos são claramente identificados na arquitetura do PROSOFT convencional. O requisito de suporte a múltiplos usuários é provido pelo PROSOFT Distribuído. As gerências de objetos, comunicação e cooperação são providas pelo PROSOFT Cooperativo. E finalmente a gerência de processo é provida no gerenciador apresentado a seguir.

4.4. Componentes do Gerenciador de Processos de Software do PROSOFT

Os componentes de mais alto nível que formam o Gerenciador de Processos (GP) são:

- **Cargos:** O GP possui um conjunto de cargos. Cada cargo contém, dentre outras informações, um conjunto de habilidades do agente para exercer o cargo (papel);
- **Agentes:** São usuários do ambiente cooperativo que estão envolvidos em vários projetos e ocupam cargos na organização. O GP contém um conjunto de agentes que participam de projetos. Cada agente possui uma agenda manipulada somente pelo mecanismo de execução que funciona como uma lista de tarefas a fazer;
- **Recursos:** Os recursos devem ser gerenciados de forma que não haja conflitos no uso dos mesmos. O GP dispõe de um conjunto de recursos que são manipulados somente pelo gerente de desenvolvimento ou pelo mecanismo de execução;
- **Ambiente Cooperativo:** É uma instância do PROSOFT Cooperativo. O GP interage com esse componente como seu super-usuário, ou seja, com acesso irrestrito a todas as informações do mesmo. A interação ocorre através de chamadas ICS e permite que durante a execução do processo de software sejam fornecidos direitos de acesso aos objetos de entrada de atividades, por exemplo;
- **Projetos:** A gerência de projetos está fortemente relacionada com a gerência de processos [GIM94]. O GP contém um conjunto de projetos, onde cada projeto possui, dentre outras informações, um processo de desenvolvimento, um conjunto de métricas (algumas são coletadas automaticamente) e um conjunto de estimativas (útil para detectar falhas no andamento do processo através da comparação entre estimativas e métricas);
- **Processos Definidos:** O GP contém um conjunto de processos definidos, que são descrições

de processos prontas para reutilização. Quando for necessário, um gerente de desenvolvimento ou outro agente habilitado pode solicitar reutilização de um processo definido para um projeto que deverá entrar em execução.

A principal tarefa do GP é executar um processo de desenvolvimento descrito em um projeto. Portanto as características do processo de desenvolvimento e suas atividades são importantes para explicitar as características do gerenciador.

Um processo de desenvolvimento é um conjunto de atividades. As operações sobre um processo são refletidas diretamente nas suas atividades. As atividades de um processo podem ter sub-atividades. Entretanto, todas as atividades, em qualquer nível de granulosidade possuem as mesmas características básicas. A seguir são apresentados os principais componentes de uma atividade de um processo de desenvolvimento:

- **Agentes:** os responsáveis pela execução da atividade que serão notificados pelo GP quando a atividade estiver pronta para começar. Pode ser um agente específico, um grupo de agentes (por exemplo grupo de revisão) ou a combinação de agentes que não estão no mesmo grupo. Quando a atividade for automática, o agente não é especificado;
- **Produtos envolvidos:** um conjunto de produtos de entrada (que serão consumidos pela atividade) e um conjunto de produtos de saída (produzidos pela mesma). Os produtos são objetos cooperativos e o GP manipula seus direitos de acesso para os agentes da atividade;
- **Condições:** A execução de uma atividade pode estar condicionada à existência de um produto ou ao estado de um objeto cooperativo. Assim, uma atividade pode ter uma pré-condição, avaliada antes da execução, e uma pós-condição, avaliada depois da sua execução;
- **Estados:** cada atividade pode estar em um dos seguintes estados: **Esperando**: quando a atividade está aguardando que suas antecessoras terminem; **Pronta**: quando todas as atividades anteriores já concluíram; **Ativa**: quando a atividade está em execução; **Parada**: quando todos os agentes envolvidos na mesma solicitaram pausa na atividade; **Completa**: quando todos os agentes envolvidos na mesma solicitaram seu término, ou quando a atividade é automática e completou sua função. O controle da transição de estados das atividades é automatizado, não sendo permitida manipulação por qualquer usuário do ambiente;
- **Recursos:** o conjunto de recursos alocados quando a atividade está ativa;
- **Cronograma:** com datas de início e fim planejadas e reais para a atividade;
- *Script:* pode ser representado por: **Descrição Informal**: a atividade deve ser executada por agentes humanos pois consiste de um texto produzido pelo projetista do processo; **Operação automática**: a atividade é executada por uma ferramenta do PROSOFT. Portanto, o *script* consiste de uma mensagem ICS para um ATO do ambiente no momento da execução da atividade; **Processo**: quando a atividade é decomposta em outras atividades. A figura 8 mostra um exemplo de decomposição de atividades de um processo.

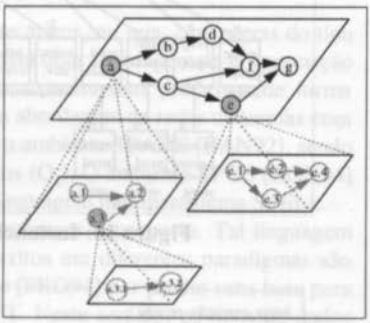

Figura 8: Decomposição de atividades [LIM98].

4.5. ATOs do Gerenciador de Processos

Para a construção do ambiente gerenciador de processos foram desenvolvidos 10 ATOs

PROSOFT e utilizados outros ATOs disponíveis no ambiente. O ATO Gerenciador de Processos (GP) é a principal ferramenta do ambiente, integrada a todas as outras ferramentas e através da qual é possível interagir com o ambiente. A instanciação do ATO GP é apresentada na figura 9.

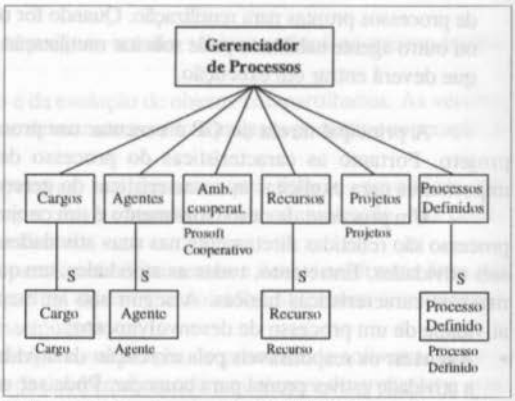

Figura 9: Instanciação do ATO Gerenciador de Processos [LIM98].

Conforme apresentado na seção anterior, o GP possui vários componentes definidos como ATOs PROSOFT e manipulados através de mensagens ICS. O componente Projetos representa os vários projetos de desenvolvimento ocorridos ou em execução. Para cada projeto existe um processo de desenvolvimento que, por sua vez, consiste de um conjunto de atividades. Cada atividade pode ter sub-atividades, caracterizando os vários níveis hierárquicos de um processo de desenvolvimento. A figura 10 apresenta a instanciação do ATO Atividade.

As operações algébricas para esses ATOs também não serão mostradas por questão de espaço. Entretanto estão listadas na figura 11 algumas operações importantes do ATO GP.

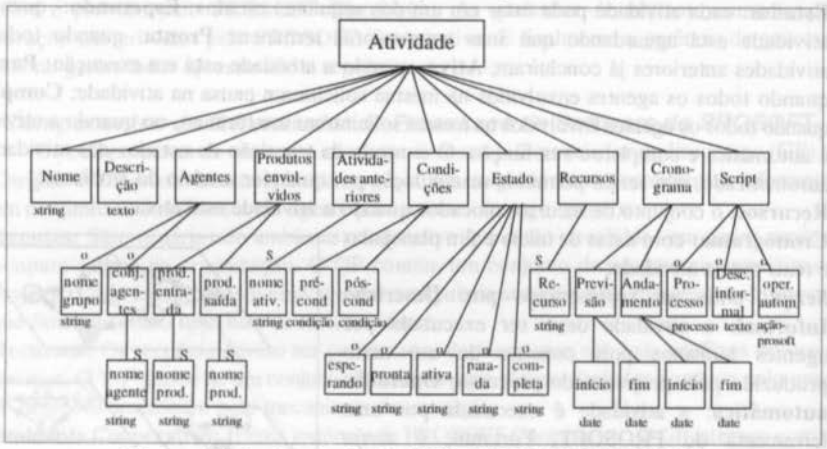

Figura 10: Instanciação do ATO ATIVIDADE [LIM98].

inicia_execução_projeto	op_objeto_cooperativo	custo_real_agentes_projetos
reutiliza_procdef_em_projeto	percentual_ativs_completas	custo_real_recursos_projetos
refazer_atividade	percentual_ativs_atrasadas	define_novo_projeto
executar_atividade_agente	percentual_ativs_paradas	altera_processo_projeto
pausar_atividade_agente	percentual_ativs_ativas	atualiza_métricas_projeto
reiniciar_atividade_agente	custo_previsto_agentes_projetos	altera_atividade_projeto
terminar_atividade_agente	custo_previsto_recursos_projetos	

Figura 11: Exemplos de operações do ATO GP.

5. Trabalhos relacionados

Nesta seção são realizados alguns comentários acerca de trabalhos relacionados ao desenvolvimento cooperativo de software e execução de modelos de processo de software.

5.1. Desenvolvimento Cooperativo de Software

Acerca do apoio ao envolvimento cooperativo de profissionais no contexto de desenvolvimento de software diversas abordagens são encontradas na literatura, como destacado por [ARA97] e [REI98]. Em síntese, pode-se identificar duas divisões principais relacionadas ao sincronismo de percepção proporcionado pelas ferramentas de edição cooperativa: o suporte a interações assíncronas (como proposto por [DUA92] e [SOU97]) e o apoio à edição síncrona de artefatos de software ([DOU97] e [ROS92]). O PROSOFT Cooperativo, por sua vez, define uma abordagem para edição síncrona de objetos que integra um modelo de versões ortogonal ao modelo de objetos. Além disso, sobre esse modelo foi definido um conjunto de estados que refletem marcos históricos na evolução e manipulação dos objetos.

Acredita-se que a principal contribuição do PROSOFT Cooperativo à área de CSCW consiste em estabelecer um tratamento formal para o problema. Apesar dos inúmeros benefícios da aplicação de técnicas de especificação formal no contexto do desenvolvimento de *groupware*, o seu uso ainda é muito limitado. Aspectos como testes de validação de *groupware* são difíceis de se aplicar devido, principalmente, ao fato de serem constituídos por grande observação e, consequentemente, muito tempo é gasto na avaliação dos protótipos. Observa-se que a integração de técnicas de especificação formal ao ciclo de desenvolvimento de software cooperativo pode ser útil para incrementar a qualidade do *groupware* desenvolvido, assim como é útil para qualquer sistema complexo [COH86].

5.2. Gerência de processos de software

Modelos de processo de software podem ser representados através de vários paradigmas, sendo que o paradigma escolhido influencia a forma de execução do processo. Em [KAI88], as ações do processo são representadas através de regras associadas a pré e pós condições e avaliadas por uma máquina de inferência. Na abordagem apresentada aqui, cada atividade possui condições associadas, mas estas são avaliadas pelo mecanismo de execução e não por uma máquina de inferência.

O ambiente Adele [BEL94] utiliza bancos de dados ativos, ou seja, com regras do tipo Evento-Condição-Ação. O ambiente Arcadia [TAY88] possui uma abordagem de programação de processos que derivou da linguagem Ada e, portanto, executa seus processos de forma procedimental. O ambiente MELMAC [DEI90] utiliza uma abordagem de redes de tarefas com redes de Petri de alto nível (redes FUNSOFT), assim como o ambiente SPADE [BAN92], sendo que este usa um sistema de banco de dados orientado a objetos (O_2). O ambiente EPOS [CON94] [NGU96] utiliza um paradigma híbrido e está baseado na linguagem multiparadigma SPELL.

Em [FRO94] uma linguagem intermediária de processos é proposta. Tal linguagem funciona como uma semântica para a qual modelos descritos em diferentes paradigmas são traduzidos. A abordagem apresentada aqui se assemelha a de [FRO94] por prover uma base para a construção de linguagens de processo para o PROSOFT. Neste sentido, os tipos de dados construídos procuram atender todos os paradigmas encontrados na literatura, ou seja, procurou-se criar uma semântica para que qualquer paradigma possa gerar modelos executáveis no ambiente.

Do ponto de vista arquitetural, o ambiente PROSOFT baseia-se no modelo de referência ECMA [BRO92], sendo que o PROSOFT Cooperativo contempla os serviços de repositório e de integração de dados, enquanto que o GP fornece os serviços de gerência de processos.

A evolução de processos no PROSOFT permite que modelos sejam modificados durante

a execução (por agentes autorizados) desde que sejam observadas algumas regras de consistência. Do ponto de vista da interação com usuários, a abordagem apresentada assemelha-se com o ambiente ProcessWeaver [CHR95], onde agendas são o principal mecanismo de comunicação entre o ambiente e o desenvolvedor.

Alguns pontos desenvolvidos nessa abordagem não são tratados em sua maioria pelos ADS citados, como por exemplo: reutilização de processos, gerência de métricas (com coleta automática) e estimativas, gerência automática dos recursos e dos direitos de acesso a objetos cooperativos, execução automática de atividades, dentre outros pontos.

Neste sentido, procurou-se reunir na abordagem apresentada várias propostas a fim de aumentar a qualidade do desenvolvimento de software, assim como mostrar a integração dessas propostas no PROSOFT. A partir do que foi construído é possível vislumbrar a evolução do ambiente com um paradigma de auxílio à reutilização de processos de software, assistência inteligente ao usuário através de bases de conhecimento, geração automática de estimativas baseada no banco de métricas de projetos similares, dentre outras possibilidades.

6. Conclusões e trabalhos futuros

Este artigo apresentou a evolução do ADS PROSOFT para gerenciar o processo de desenvolvimento cooperativo de software. Para tanto, os principais objetivos deste ADS foram mostrados, enfatizando o apoio à utilização de métodos formais no ciclo de vida de software.

Os requisitos que influenciaram o Gerenciador de Processos de Software do PROSOFT foram mostrados, bem como o seu relacionamento com o PROSOFT Cooperativo. O Gerenciador apresentado neste artigo provê apoio à execução, reutilização e evolução de modelos de processo e foi especificado na forma de ferramentas (ATOs) que foram integrados ao PROSOFT. Esta evolução não compromete nenhuma característica do ambiente e sua integração ao PROSOFT Cooperativo proporciona serviços para controlar a interação cooperativa dos engenheiros de software na utilização dos ATOs dispostos no ambiente.

Uma consequência deste trabalho é a integração das vantagens encontradas nas técnicas de especificação formal à tecnologia de gerenciamento de processo de software. Esta combinação pode proporcionar um aumento significativo na qualidade dos produtos desenvolvidos [COH86], assim como permite que propriedades de processo de software sejam pesquisadas e avaliadas. Conforme esperado, o processo de especificação formal proporcionou importantes percepções acerca das tecnologias sendo estudadas, o que ajudou a aumentar a qualidade dos componentes propostos. Espera-se com a disponibilização de novos ATOs-*Groupware* e a completa adaptação dos ATOs existentes para suportar a edição cooperativa de objetos a realização de uma avaliação quantitativa do aumento da produtividade atingido e qualitativa dos produtos gerados.

Este Gerenciador de Processo de Software requer e fornece uma base genérica para que linguagens de modelagem de processos sejam propostas para o ambiente e integradas ao modelo de execução proposto. Portanto, não impõe um paradigma de modelagem específico. Atualmente uma linguagem de modelagem de processos de software está sendo desenvolvida [DAL97] para o ambiente.

Para validar o mecanismo de execução proposto foi construída em [LIM98] uma instância do ambiente contendo um exemplo de processo de software (conhecido como ISPW-6 *example*) extraído de [KEL90], com algumas características a mais como gerência de recursos e execução automática de atividades. Este exemplo é geralmente usado para validar ADS orientados a processos, em função de suas características. A execução deste processo exemplo (um termo do gerenciador de processos) foi convertida para um modelo CCS, permitindo a percepção das características de interação entre as atividades e o paralelismo das mesmas.

PROSOFT é um ambiente em constante evolução. O PROSOFT Cooperativo está

evoluindo para apoiar a interação assíncrona, em especial os aspectos voltados à documentação acerca de justificativas de projeto, argumentação de grupo e apoio à tomada de decisões de grupo, a partir da extensão da ferramenta HYPERPRO [BAL94], disposta no ambiente para geração de hiper-documentos. Atualmente, os esforços são direcionados para construir uma interface entre o ADS PROSOFT e a linguagem de programação Java (como apresentado em [SCH97]). Uma nova geração de ATOs distribuídos é esperada.

7. Bibliografia

[ARA97] Araújo, R.; Dias, M.; Borges, M. Suporte Por Computador Ao Desenvolvimento Cooperativo de Software: Classificação e Propostas. Simpósio Brasileiro de Engenharia de Software, 11., Fortaleza. **Anais...** SBC, 1997, p.299-314.

[BAL94] Balsemão, L.; Nunes, D. **Hyperpro**: Uma ferramenta para a construção de hiperdocumentos no PROSOFT: Trab. de Diplomação. Porto Alegre: Cic/ UFRGS, 1994.

[BAN92] Bandinelli, S. et al. Process Enactment in SPADE. In: European Workshop On Software Process Technology, 2., 1992, Trondheim, Norway. **Proceedings...**

[BEL94] Belkhatir, N.; Estublier, J.; Melo, W. ADELE-TEMPO: An Environment to Support Process Modelling and Enaction. In: FINKELSTEIN, A. et al. (Ed.). **Software Process Modelling and Technology**. Research Studies Press, 1994.

[BJØ78] Bjørner, D.; Jones, C. **The Vienna Development Method:** the meta-language. Berlin: Springer-Verlag, 1978. (Lecture Notes in Computer Science)

[BOR95] Borges, M.; Cavalcanti, M.; Campos, M. **Suporte por Computador ao Trabalho Cooperativo.** Canela-RS: II/UFRGS, 1995. Trabalho apresentado na Jornada de Atualização em Informática, 14., 1994, Canela.

[BRO92] Brown, A.; Earl, A.; McDermid, J. **Software Engineering Environments**: Automated Support for Software Engineering. London: McGraw-Hill, 1992.

[CHR95] Christie, A. **Software Process Automation**: The technology and its adoption. Springer Verlag, 1995.

[COH86] Cohen, B. et al. **The Specification of Complex Systems**. Addison Wesley, 1986.

[CON94] Conradi, R. et al. EPOS: Object-Oriented Cooperative Process Modelling. In: Finkelstein, A. et al. (Ed.). **Software Process Modelling and Technology**. Research Studies Press, 1994. p. 33-70.

[DAL97] Dal Pizzol, A. Sintaxe e semântica abstrata de uma linguagem de programação de processos de software no PROSOFT. In: Semana Acadêmica do CPGCC, 2., 1997, Porto Alegre. **Anais...** Porto Alegre: CPGCC da UFRGS, 1997. p. 195-198.

[DEI90] Deiters, W.; Gruhn, V. Managing Software Processes in the Environment MELMAC. **Software Engineering Notes**, v. 15, n. 6, Dec. 1990.

[DOU97] Dourish, P. **Extending Awareness Beyond Synchronous Collaboration**. Disponível por www em http://www.best.com/~jdp/chi97-awareness.html (1997)

[DUA92] Duarte, R.; Fuks, H.; Lucena, C. Software Design Cooperativo: Um Estudo de Caso. In: Simpósio Brasileiro de Engenharia de Software, 6., Gramado. **Anais...** UFRGS, 1992.

[FEI93] Feiler, P.; Humphrey, W. Software Process Development and Enactment: Concepts and Definitions. In: Int. Conference on the Software Process, 2., 1993, Berlin. **Proceedings...** IEEE Computer Society Press, Mar. 1993.

[FRO94] Froehlich, G. **Process Modelling Support in Metaview**. Canada: University of Sakatchewan, 1994. Master of Science Thesis.

[GIM94] Gimenes, I. **Uma introdução ao Processo de Engenharia de Software**: Ambientes e Formalismos. Caxambu-MG: SBC, 1994. 42f. Trabalho apresentado na Jornada de

Atualização em Informática, 13., 1994, Caxambu.
[GOL96] Golendziner, L.; Santos, C. **Versions and configurations in object-oriented database systems: a uniform treatment.** Prêmio Compaq de estímulo à pesquisa e desenvolvimento em informática, 1., São Paulo: Instituto Uniemp, 1996.
[HUM89] Humphrey, W., **Managing the Software Process**. Addison-Wesley, 1989.
[KAI88] Kaiser, G.; Feiler, P.; Popovich, S. Intelligent Assistance for Software Development and Maintenance. **IEEE Software**, v. 5, n. 3, May 1988.
[KEL90] Kellner, M. et al. ISPW-6 Software Process Example. In: International Software Process Workshop, 6., 1990, Kioto, Japan. **Proceedings...** IEEE Press, Oct. 1990.
[LON93] Lonchamp, J. A Structured Conceptual and Terminological Framework for Software Process Engineering. In: International Conference on The Software Process, 2., 1993, Berlin. **Proceedings...** IEEE Press, Mar. 1993.
[LIM98] Lima, C. **Um Gerenciador de Processos de Software para o Ambiente PROSOFT**. Porto Alegre: CPGCC-UFRGS, 1998. Dissertação de Mestrado.
[MOR97] Moraes, S. **Um Ambiente Expert para Apoio ao Desenvolvimento de Software.** Porto Alegre: CPGCC-UFRGS, 1997. Dissertação de Mestrado.
[NGU96] Nguyen, M.; Wang, A. **Total Software Process Model in EPOS**. Disponível por www em: http://www.idt.unit.no/~epos/Papers. (dez. 1996).
[NUN92] Nunes, D. Estratégia Data-Driven no Desenvolvimento de Software. In: Simpósio Brasileiro de Engenharia de Software, 6., Gramado. **Anais...** II/UFRGS, 1992.
[NUN94] Nunes, D. **PROSOFT:** Rel. de Pesquisa. Porto Alegre: CPGCC-UFRGS, 1994.
[REI97] Reis, R.; Lima, C.; Nunes, D. Cooperative Software Development and the PROSOFT Environment. In: International Symposium On Computer And Information Sciences, 12., 1997, Antalya, Turkey. **Proceedings....** Bogaziçi University, Oct. 1997.
[REI98] Reis, R. **Uma Proposta de Suporte ao Desenvolvimento Cooperativo de Software no Ambiente PROSOFT.** Porto Alegre: CPGCC-UFRGS, 1998. Dissertação de Mestrado.
[REK92] Rekers, J.; Sprinkhuiken-kuyper, I., Leiden Univeristy, 1992. **A Lotos specification of a CSCW tool.** Ftp:// ftp.wi.LeidenUniv.nl/pub/CS/TechnicalReports/1992/tr92-28.ps.gz
[ROS85] Ross, D.Applications and Extensions of SADT.**IEEE Computer**, v.18, n.4,Apr.1985.
[ROS92] Roseman, M.; Greenberg, S. GroupKit: A Groupware Toolkit for Building Real-Time Conferencing Applications. In: ACM Conference on Computer-supported Cooperative Work (CSCW 1992), 1992, Toronto. **Proceedings...** ACM PRESS, Nov. 1992. p. 43-50.
[SCH95] Schlebbe, H. **Distributed PROSOFT.** Department of Computer Science, Tecnical Report, University of Stuttgart, Germany, 1995.
[SCH97] Schlebbe, H.; Schimpf, S. **Reengineering of PROSOFT in Java.** Technical Report, University of Stuttgart, Germany, 1997.
[SOU97] Souza, C.; Wainer, J.; Rubira, C. **Um Modelo de Anotações para o Desenvolvimento Cooperativo de Software.** Disponível por www em http://www.icmsc.sc.usp.br/womh97/artigos/crbsouza.ps, 1997.
[TAY88] Taylor, R. N. et al. Foundations for the Arcadia Environment Architecture. **Software Engineering Notes**, New York, v. 13, n. 5, Feb. 1989.
[TOP95] Topper, A. Tools for group development. **Object Magazine**, v. 4, n.8, Jan. 1995.
[VES95] Vessey, I.; Sravanapudi, A.. CASE Tools as Collaborative Support Technologies. **Communications of the ACM**. v. 38, n. 1, p.83-95 Jan. 1995.
[WAT91] Watt, D. **Programming Language Syntax and Semantics.** Prentice-Hall, 1991.