
# Relatório Técnico

Núcleo de Computação Eletrônica

## Suporte Automatizado ao Projeto Cooperativo de Software

Renata Mendes de Araujo
Marcos Roberto da Silva Borges

NCE - 06/97

Universidade Federal do Rio de Janeiro

# - Suporte Automatizado ao Projeto Cooperativo de Software -

## Uma Coletânea das Abordagens para Suporte ao Entendimento, Comunicação, Coordenação e Memória de Grupo em Projetos de Software

Renata Mendes de Araujo[1] Marcos Roberto da Silva Borges[2]

**Resumo**

*O objetivo deste trabalho é o de apresentar as propostas de suporte à cooperação no contexto de desenvolvimento de software sugeridas até o momento atual na figura de ferramentas ou ambientes voltados especificamente para o suporte à colaboração neste contexto. Para isto, procuramos classificá-las descrevendo os recursos que oferecem para o suporte aos requisitos que consideramos como fundamentais para o completo apoio ao projeto cooperativo de software: o entendimento entre os participantes, a comunicação, o registro do conhecimento e a coordenação e controle do processo.*

**Palavras-Chave:** CSCW, *groupware*, suporte ao projeto cooperativo de software

## 1. INTRODUÇÃO

Já não se questiona mais a característica cooperativa de processos de desenvolvimento de software. Uma gama de atividades de interação entre indivíduos é realizada para promover a construção de um sistema e cada vez mais torna-se reconhecida a necessidade de melhorar a qualidade destas interações como ponto fundamental para o conseqüente aumento da qualidade dos produtos gerados e para promover o agenciamento desta produção. O próprio modelo de capacitação de software - CMM (*Capability Maturity Model*) - define que, conforme as organizações sobem para níveis mais altos de maturidade, as tarefas de projeto passam de tarefas envolvendo o desenvolvimento individual e isolado para o trabalho em equipes [PATN95].

A construção de um produto de software é um processo que envolve grupos de atores com interesses e visões distintas entre si mas em direção a um só objetivo. Usuários se preocupam com a funcionalidade, possibilidades de interação e influências do novo software em suas atividades; engenheiros de software se concentram em capturar os requisitos de forma acurada e completa; programadores procuram por desempenho, confiabilidade e robustez; e, por fim, os gerentes tentam controlar o planejamento e o fluxo de trabalho.

Coordenar estes interesses e possibilitar que estes atores trabalhem harmonicamente pode parecer apenas uma questão gerencial. Mas, nota-se que toda uma filosofia de trabalho cooperativo é preciso ser aplicada para que todos os fatores que envolvem este processo possam ter garantias quanto a alcançarem seus objetivos individuais, gerando um produto útil comum.

Aplicar técnicas cooperativas na produção de software não parece ser o mesmo que utilizar ferramentas gerenciais para planejamento, controle e acompanhamento do projeto. Representa uma perspectiva para a construção de ambientes de desenvolvimento que considerem o relacionamento humano como uma atividade chave durante o processo de construção [TROT92].

---

[1] COPPE - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil

[2] Núcleo de Computação Eletrônica/Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil

Buscando soluções para esta questão, um contingente da comunidade de pesquisa em engenharia de software tem procurado aplicar os conceitos e resultados obtidos com as pesquisas que envolvem a área de Trabalho Cooperativo Suportado Por Computador (CSCW) no contexto de desenvolvimento de software, na tentativa de suportá-lo através de técnicas, ambientes e ferramentas cooperativas.

Uma grande variedade de propostas têm sido apresentadas, apostando na inegável necessidade de suporte automatizado ao processo de colaboração no contexto de desenvolvimento de software e seguindo as tendências atuais das novas tecnologias de comunicação e interação. Muitas destas propostas definem abordagens sociais e metodologias para suporte à interação de grupos de desenvolvimento. Outras propostas tratam especificamente sobre a especificação, projeto e uso de groupwares específicos para este contexto.

Nosso objetivo, com este trabalho, é o de apresentar as propostas de suporte à cooperação no contexto de desenvolvimento de software sugeridas até o momento atual na figura de ferramentas ou ambientes voltados especificamente para o suporte à colaboração neste contexto. Para isto, procuramos classificá-las descrevendo os recursos que oferecem para o suporte aos requisitos que consideramos como fundamentais para o completo apoio ao projeto cooperativo de software: o entendimento entre os participantes, a comunicação, o registro do conhecimento e a coordenação e controle do processo.

A partir de uma análise das características de suporte oferecidas por cada uma das propostas, procuramos delinear o estado da arte neste nicho de pesquisa, levantando as questões que já foram resolvidas, os problemas que já se reconhecem a existência mas estão abertos para serem resolvidos e questões ao nosso ver importantes que ainda não foram consideradas.

Nas seções seguintes, discutimos as questões que envolvem o suporte ao projeto cooperativo de software e apresentamos as abordagens sugeridas pelas pesquisas na área para atender a cada um destas questões. Na seção 7 analisamos as propostas das ferramentas ou ambientes de suporte ao projeto cooperativo de software em relação a questões quanto a arquitetura, plataforma e suporte a fases do processo de desenvolvimento. A seção 8 oferece um resumo das abordagens descritas nas seções anteriores e uma análise das informações coletadas.

## 2. QUESTÕES ENVOLVENDO O PROJETO COOPERATIVO DE SOFTWARE

Segundo nosso ponto de vista, quatro questões se evidenciam como primordiais para que o processo de colaboração entre os membros de uma equipe de desenvolvimento se realize com sucesso. Estas questões encontram-se intimamente dependentes e interrelacionadas uma às outras e correspondem à **comunicação** efetiva entre as partes, à possibilidade de **entendimento** através desta comunicação, ao **registro do conhecimento** gerado durante o trabalho sendo desenvolvido e à **coordenação** ou controle da realização das tarefas [ARAU96].

Estas questões são dependentes umas das outras na medida em que, para suportar corretamente o processo de colaboração, é preciso investir em entendimento e, consequentemente, viabilizá-lo através de recursos de comunicação. Para possibilitar o entendimento é preciso, ainda, tornar o conhecimento explícito, ou seja, organizar o conhecimento comum do grupo, de forma que qualquer um possa acessá-lo. Mas, toda esta rede de recursos precisa caminhar para um objetivo único e comum e, no caso de desenvolvimento de software, é preciso que seja mantido um ritmo, uma coordenação.

## 3. SUPORTE AO ENTENDIMENTO

*"The Holy Grail of teamwork is shared understanding."*

J. Conklin [CONK96]

Usuários, desenvolvedores e gerentes de projetos, como especialistas, vivem em ambientes de trabalho complexos definidos por suas áreas de especialização. Quando se reúnem em equipes, a profundidade de experiências e conhecimento de cada um aparece como parte da base de conhecimento da equipe. Mas, estas diferenças em especialização podem emperrar o processo de entendimento como se estivessem falando línguas diferentes. Para superar este obstáculo, a equipe deve construir toda uma estrutura para compartilhar seus conhecimentos, principalmente no que se refere a conceitos e termos, para que a colaboração entre eles não seja continuamente prejudicada por problemas de mal entendimento.

O entendimento ou a formação de um conhecimento comum corresponde à construção ou estabelecimento de uma interface de compreensão e à construção de uma estrutura para compartilhamento de conceitos, termos e nomenclaturas. Padrões de conversação, linguagens de trabalho, definição de conceitos, glossários, dicionários de dados etc são recursos que irão compor esta interface que é o conhecimento comum.

A palavra-chave para a escolha de recursos de suporte ao entendimento é **visualização comum**. Ou seja, se o conhecimento coletivo pudesse ser encarado como um objeto concreto, poderíamos dizer que este seria eficiente somente se usuários, desenvolvedores e software pudessem vê-lo como um mesmo objeto, mesmo tamanho, forma e cores. Estas interfaces deveriam descartar ambigüidades e objetivar sempre a normalização de conceitos que influenciarão a comunicação da equipe, as características da interface (homem/máquina) do futuro sistema e a cultura da coletividade que o receberá.

### 3.1 ABORDAGENS DE SUPORTE AO ENTENDIMENTO ENTRE PARTICIPANTES

A partir da análise das características das ferramentas ou ambientes que se propõem a suportar o entendimento ao longo do processo de desenvolvimento, percebemos que algumas propostas se concentram em oferecer recursos para que os membros da equipe de projeto possam, cooperativamente, gerar, organizar e refinar conceitos relativos ao contexto de desenvolvimento. Outras propostas dão ênfase ao levantamento e refinamento cooperativo de requisitos do sistema em construção.

#### 3.1.1 Suporte à Formalização de Conceitos

Mais do que meros dicionários de dados, algumas ferramentas auxiliam a coleta/geração e organização de conceitos. Podem ir além, reconhecendo a necessidade de suporte à discussões como mecanismo para promover o refinamento destes conceitos.

A ferramenta *Conflict-Resolution Supporting Tool* [IOCH95][MARA96], por exemplo, tem o objetivo de auxiliar equipes de desenvolvimento a identificar conflitos de linguagem entre os membros do grupo, para evitar que erros de conceituação possam comprometer a qualidade do processo de desenvolvimento.

Ao invés de se preocupar em suportar a resolução dos conflitos de linguagem, a ferramenta enfatiza o suporte à identificação destes conflitos através da análise dos diálogos e documentos que descrevem a aplicação ou o projeto do sistema. A ferramenta é capaz de estabelecer um conjunto de termos mais utilizados no contexto de desenvolvimento em questão e, baseado neste conjunto, os criadores do texto devem prover suas próprias definições para cada um. Os usuários ou consumidores destes textos podem questionar a definição dos termos e uma negociação é desencadeada. Esta negociação é suportada pela ferramenta.

O objetivo de entendimento entre participantes do projeto e suporte ao trabalho criativo em conjunto motiva a ferramenta DNP (*Designer's Notepad)* [TWID93]. Esta ferramenta procura oferecer mecanismos para suportar a liberdade de expressão em relação a conceitos, necessária nos estágios iniciais do projeto de software, e uma maior formalização destes conceitos conforme o projeto evolui.

Trata-se, portanto de um processador de idéias, envolvendo a geração, organização e refinamento de conceitos e o suporte à argumentação em relação às definições. O núcleo do sistema é uma ferramenta gráfica com recursos de hipertexto que possibilita a rápida geração, organização e refinamento de diagramas de conceitos.

### 3.1.2 Suporte ao Levantamento e Refinamento de Requisitos

Nestas propostas, paradigmas mais complexos para representação de requisitos como pontos de vista e cenários são utilizados como interface de entendimento entre as partes envolvidas com a atividade de levantamento de requisitos. Estas propostas também reconhecem a necessidade da argumentação para refinamento e formalização do conhecimento em construção.

A ferramenta VORD (*Viewpoint-Oriented Requirements Definition*) [VORD96] se propõe a suportar a atividade de levantamento e engenharia de requisitos de software baseando-se no paradigma de pontos de vista.

O conceito de ponto de vista corresponde à representação do sistema sob a visão de um indivíduo, uma organização ou um sub-sistema. Através da coleta dos pontos de vistas de todas as partes interessadas na construção do software, pode ser possível a criação de um panorama preciso das expectativas e necessidades do ambiente onde será inserido.

VORD se preocupa em coletar os múltiplos pontos de vista, especificar requisitos para cada um e possui extensões para realizar análises para levantamento de conflitos e anomalias entre pontos de vista para serem discutidos. A princípio, esta discussão sobre conflitos e anomalias de pontos de vista não é suportada pela ferramenta, ou seja, não é oferecido um ambiente automatizado para discussão entre os participantes. Após todas as anomalias terem sido discutidas e resolvidas, pode-se partir para a elaboração da especificação de requisitos do sistema.

Não nos foi possível obter informações sobre como os pontos de vista são coletados e se esta atividade é uma atividade explicitamente cooperativa. Ao que nos parece, a ferramenta não oferece recursos para interação entre os envolvidos com a coleta dos pontos de vista, se restringindo apenas a analisá-los.

Cavalcanti e Borges [CAVA96] propõem um ambiente cooperativo para suporte ao projeto participativo de software. O principal objetivo do ambiente é o de reduzir os problemas de equívoco e incertezas que influenciam o processo de entendimento entre os elementos participantes de uma equipe de software. Este ambiente utiliza as abordagens de análise de cenários e um modelo de discussão baseado no *Inquiry Cycle Model* como técnicas para levantamento e refinamento de requisitos.

O ambiente de projeto de software *Evolving Artifact* (EVA) [OSTW95] se propõe a auxiliar a construção e compreensão iterativa de problemas de projeto e suas soluções através da construção e refinamento de representações de projeto.

Uma das premissas do projeto é a de que o significado não é uma propriedade intrínseca de artefatos e símbolos. Ao contrário, os significado são prescritos por cada cultura. Representações de projeto devem, portanto, ter significados iguais entre as fronteiras culturais que se estabelecem entre usuários e desenvolvedores.

O modelo "ação-reflexão-crítica" é proposto como modelo para projeto colaborativo, que sintetiza a teoria de projeto com teorias de como pessoas solucionam problemas. Este modelo permite a usuários e desenvolvedores desenvolver um entendimento do problemas de projeto e de suas soluções através da construção de representações externas que dão base à comunicação

sobre as intenções de projeto e críticas em relação ao mesmo. O entendimento mútuo é estabelecido através da criação, discussão e modificação das representações.

A abordagem de EVA é a de combinar documentação de projeto a protótipos, onde protótipos aumentam o significado da documentação para os usuários e a documentação provê o contexto para a interpretação de protótipos.

## 4. SUPORTE À COMUNICAÇÃO

> *"A large-scale project has a lot of moving parts, which makes it that much easier to break down. Communication is the oil that keeps everything working properly."*
>
> Ewin Martinez (CSC Consulting & Systems Integration) [HILD96]

Quando pensamos em possibilidades de comunicação, a palavra-chave torna-se **ligação**. Não se pode esperar colaboração entre partes sem que haja uma ligação entre elas. A comunicação entre os membros de uma equipe de desenvolvimento reside na existência e potencialidade das ligações entre eles. Estas ligações podem ser definidas como técnicas e/ou canais de comunicação que permitem aos usuários e desenvolvedores trocar informações [KEIL95].

Uma série de fatores contribuem, hoje, para dispor uma variedade de ligações, tais como os avanços tecnológicos e a redução de custos de serviços de telecomunicações. Estas ligações podem se classificar em **diretas** e **indiretas**. Ligações diretas são importantes quando existe um alto índice de ambigüidade na comunicação, situação que ocorre comumente, por exemplo, durante a elicitação de requisitos. Em ligações indiretas, os participantes se comunicam através de intermediários (humanos ou não).

Keil e Carmel [KEIL95] analisam as ligações mais comumente utilizadas em projetos de software, e suas conclusões indicam que: um maior número de ligações implica em melhor comunicação; deve-se reduzir a confiança em ligações indiretas; e deve-se considerar o uso de ligações não utilizadas tradicionalmente no contexto de trabalho. Contudo, o uso indiscriminado de ligações pode levar ao insucesso, uma vez que a troca de informações se distribui por muitos canais, não sendo possível para os personagens captar todas elas. Daí ser necessário o estabelecimento de um número máximo de ligações. Além disso, um número excessivo de ligações pode implicar em custos maiores.

A eficiência das ligações varia, ainda, de acordo com o ambiente de desenvolvimento. Segundo as análises dos autores, cada ambiente parece "preferir" uma determinada ligação que talvez nunca tenha sido utilizada em outros ambientes.

### 4.1 ABORDAGENS PARA SUPORTE À COMUNICAÇÃO

Procuramos observar como os ambientes de suporte ao projeto cooperativo de software disponibilizam as ligações de comunicação entre seus usuários em relação: ao modo de interação imposto ao grupo de desenvolvimento, aos recursos de percepção (*awareness*) oferecidos aos participantes e às metáforas de interação utilizadas para comunicação.

#### 4.1.1 Modos de Interação

Quanto aos modos de interação, várias das ferramentas e ambientes estudados não discutem este aspecto ou mesmo mencionam como pretendem promover a interação entre seus participantes. Pelo o que nos foi possível analisar, privilegia-se a interação remota [CSD96][ARAU94] e assíncrona [ARAU94][CPCE94][CAVA94][BISC94], talvez por ser o modo de interação que leve a maiores novidades de pesquisa e flexibilize mais as oportunidades de interação entre os membros da equipe de projeto.

Por outro lado, muitas propostas de ambientes de suporte ao projeto cooperativo de software se baseiam na definição de uma estrutura de integração de ferramentas (vide seção 7.2) e, assim, o

modo de interação entre seus usuários estaria vinculado ao modo de interação imposto por cada ferramenta.

Nota-se claramente, ainda, uma tendência a flexibilizar o modo de interação de acordo com a natureza da atividade sendo realizada, conforme apontado por Cavalcanti e Borges em sua proposta de ambiente para projeto participativo de software [CAVA96], no ambiente Flecse (*Flexible Environment for Collaborative Software*) [DEWA93] e no ambiente OZWeb [OZWE96]. Isto significa que os ambientes procuram oferecer uma estrutura de interação onde passar de interações síncronas para interações assíncronas, bem como ter acesso ao contexto de trabalho de forma remota ou localmente, seja uma transição possível e facilitada.

### 4.1.2 Espaços de Trabalho Compartilhados

Segundo Twidale, Rodden e Sommerville [TWID93], espaços de trabalho compartilhados são os recursos mais utilizados para suporte ao projeto cooperativo de software. Nesta abordagem, aos desenvolvedores é oferecido uma superfície compartilhada, onde podem expressar suas idéias e/ou construir seus produtos.

A ferramenta DNP (*Designers' Notepad*) [TWID93] oferece um espaço gráfico compartilhado como meio para representação de idéias durante conversas e discussões realizadas durante as atividades de projeto. Da mesma forma, o ambiente para suporte ao projeto participativo de software de Cavalcanti e Borges [CAVA96] oferece ferramentas para co-edição de cenários e modelos de domínio da aplicação em construção. O ambiente Flecse [DEWA93] oferece uma ferramenta para edição cooperativa de código denominada MEdit. O ambiente CSD (*Cooperative Software Development*) [CSD96] segue a mesma filosofia, oferecendo um espaço para edição de componentes de software e realização de conferências, baseado em várias mídias (texto, gráfico e áudio). Este tipo de espaço compartilhado, com uso de recursos gráficos, é comumente utilizado para interações síncronas.

Mas, espaços compartilhados podem compreender bases de informações com o registro das interações realizadas por cada participante. Esta forma de compartilhamento de trabalho é comumente utilizada em interações assíncronas conforme proposto nos ambientes Quorum [ARAU94], EPOS (*Expert System for Program and System Development*) [EPOS96], ARCoPAS [CAVA94] e *Beyond-Sniff* [BISC94].

### 4.1.3 Uso de Sistemas de Mensagens

Sistemas de mensagens são recursos amplamente utilizados para o suporte à comunicação em ambientes cooperativos, principalmente para estabelecer canais de comunicação informal e paralela à atividade em andamento.

Este é o caso dos ambientes ICARO [LICE96] e RASP (*Regatta Automated Software* Process) [SWEN93] que utilizam o sistema de mensagens para a delegação e negociação de realização de tarefas entre os gerentes de projeto e os outros membros da equipe.

O ambiente EPOS [EPOS96] possui uma ferramenta - CHAT (*Conflict Handling Toolkit*) - baseada num sistema de mensagens para apoio à resolução dos conflitos que ocorrem durante a execução das atividades do processo. As mensagens podem ser enviadas pelos próprios usuários ou podem ter sido geradas pelo sistema na ocorrência de conflitos.

Já a ferramenta Quorum [ARAU94] utiliza um sistema de mensagens para promover interações marginais aos processos de tomada de decisão, como: solicitação de informações, lançamento de comentários, solicitação de participação de membros do grupo, comunicação de eventos, ocorrências e detalhes, envio de lembretes etc.

### 4.1.4 Percepção

Para a contextualização de cada participante em relação aos outros membros da equipe, algumas ferramentas oferecem paradigmas de percepção como cursores personalizados,

WYSIWIS, delegação de papéis e teleapontamento, conforme proposto no ambiente CSD [CSD96].

Outras ferramentas se baseiam em consultas ou navegação pelo espaço de trabalho a procura de informações sobre a interação e as atividades realizadas por cada membro. Ambientes de suporte ao trabalho em grupo devem oferecer facilidades para auxiliar o usuário a encontrar-se no sistema e o que pode ser feito ali. Dentre estas facilidades, podemos destacar: auxiliar o usuário a determinar como entrar num contexto específico; indicar as "obrigações" de um determinado usuário; auxiliar o usuário a compreender as relações que existem entre os dados presentes no sistema; auxiliar o usuário a compreender as relações existentes entre os contextos nos quais está envolvido etc.

A ferramenta CHAT, provida pelo EPOS [EPOS96], permite que os usuários obtenham informações sobre espaços de trabalho, componentes e relações. Utilizando o navegador do espaço de trabalho, os usuários podem checar se alguém está conectado, quais componentes foram recuperados da base (*check out*), quando o espaço de trabalho foi criado, quem o criou e com que objetivos.

O ambiente *Conversation Builder* [KAPL92] tem como objetivo principal ser um ambiente "ativo". Isto significa que o ambiente tem conhecimento sobre as ações realizadas pelos usuários do sistema e pode usar este conhecimento para prover ajuda sensível ao contexto, guiar usuários pelo processo de desenvolvimento e manter os usuários a par das ações de outros. *Conversation Builder* auxilia o usuário para que tenha noção das tarefas em que está engajado, as relações entre estas tarefas, as ações que pode realizar em cada uma, a estrutura da base de componentes (hipertexto) e as ações relevantes de outros membros.

O ambiente *Beyond-Sniff* [BISC94] possibilita o acesso a anotações registradas pelos membros da equipe através de consultas. As consultas podem ser feitas a partir dos artefatos de software armazenados ou podem ser formuladas consultas baseadas nos atributos das anotações (tipo, data, artefato relacionado etc). Usuários podem ser notificados quanto à ocorrência de determinadas anotações ou mesmo serem notificados de quando uma determinada anotação foi lida. Anotações podem, ainda, ser endereçadas a usuários específicos, podem ser inseridas manualmente ou podem ser geradas automaticamente por ferramentas.

## 5. SUPORTE À MEMÓRIA DE GRUPO

> *"Where is the wisdom we have lost in knowledge? Where is the knowledge we have lost in information?"*
>
> T.S. Eliot

A forma de captura do conhecimento, conforme realizada na maioria dos processos de desenvolvimento, se concentra na preservação de documentos nas mais variadas formas que representam os produtos gerados. Este conhecimento pode ser encarado como o conhecimento formal e é nele que o grupo se baseia como memória de trabalho.

Entretanto, o conhecimento dito informal, que é, a grosso modo, a descrição do processo pelo qual os produtos foram criados, compreendendo o registro das idéias, fatos, questões, pontos de vista, conversas, discussões, decisões etc, que aconteceram no decorrer do processo e acabaram por definí-lo, é difícil de ser capturado.

Os resultados finais de um processo de desenvolvimento, indiferentemente do ciclo de vida utilizado, formam a base de importantes decisões. Estas decisões compreendem a determinação do nível de sofisticação do sistema, a avaliação de sua utilidade, o congelamento de requisitos e o projeto da estrutura interna do sistema.

Mas, é importante observar que a racionalização (*rationale*) do processo e outras formas de conhecimento informal devem estar intimamente relacionados aos artefatos sendo produzidos pois um se baseia no outro para darem sentido ao processo.

O armazenamento do conhecimento requer ferramentas e processos que preservem o contexto do trabalho enquanto este evolui. Este contexto apresenta a forma de uma rede de informações que incluem fatos, hipóteses, restrições, decisões e suas razões, o significado de conceitos e, é claro, os documentos formais.

Sendo a palavra-chave relacionada à memória de grupo o **contexto**, é este que interfaceia comunicações e determina o entendimento. É sobre o contexto, os elementos armazenados na memória de grupo, que usuários, desenvolvedores e gerentes irão travar suas comunicações e ter acesso aos produtos sendo gerados. É sobre este mesmo contexto que o conhecimento comum, a interface de entendimento, se estabelece e que todos devem visualizar sob a mesma ótica. Mas, este contexto de trabalho é amorfo até que outras interfaces que promovam a sua captura, sua organização sua disseminação e seu reuso possam ser oferecidas para sua utilização.

## 5.1 Abordagens para Organização, Disseminação e Reuso da Memória Coletiva

### 5.1.1 Uso de Hipertexto

No que concerne ao estabelecimento da memória coletiva, podemos destacar o papel definitivo da metáfora de hipertexto como tecnologia para sua organização e disseminação. É através de interfaces hipertextuais que usuários e desenvolvedores irão construir uma interface de conhecimento comum, irão registrar suas idéias, estabelecer comunicação e registrar a evolução do produto gerado [ARAU96].

A maioria dos ambientes e ferramentas estudadas armazenam em uma base hipertextual o material produzido durante o processo de construção (os componentes de software). Dentre estas, algumas se preocupam em expandir esta base com as contribuições informais, ampliando a capacidade da memória de grupo. Outras, se preocupam em armazenar apenas a interação informal, e os produtos de software gerados são armazenados em outras bases, podendo, contudo, serem acessados através de ligações externas.

O ambiente OZWeb [OZWE96], por exemplo, armazena todo o material produzido durante o processo de construção de software. Mas, não é claro quanto ao armazenamento do conhecimento informal.

O ambiente *Conversation Builder* [KAPL92], por sua vez, provê um sistema de hipertexto para armazenamento e ligação de nós de informação que correspondem aos produtos de software gerados (código fonte, relatórios de erros, documentos de projeto etc). Estes nós podem ser arbitrariamente associados entre si e navegados, desde que usuários tenham permissão para tal.

O ambiente ICARO [LICE96] faculta a geração automática de documentos HTML a partir da documentação de projeto gerada com o uso de uma ferramenta CASE. A partir desta documentação em HTML, o ambiente permite que discussões sejam desencadeadas e registradas.

A ferramenta Quorum [ARAU94] possui uma base de discussão baseada numa extensão do modelo de discussão IBIS com recursos de hipertexto para livre associação entre as contribuições apresentadas durante interações de discussão e o material externo à discussão.

O ambiente ARCoPas [CAVA94] apóia o processo de engenharia reversa de sistemas, no que se refere a recuperação do projeto arquitetônico dos mesmos. O projeto arquitetônico do sistema é representado, em ARCoPAS, através da notação *HyperText Design Notation* (HTDN). A documentação do projeto arquitetônico em HTDN é gerada automaticamente por um dos módulos do ambiente e, associado à esta documentação, o grupo de manutenção do sistema pode realizar discussões para decidir quanto ao seu re-projeto.

O foco da pesquisa em EVA [OSTW95] está em oferecer ferramental de suporte para apoiar a colaboração através de representações de projeto, a fim de manter um processo de construção evolutivo baseado na comunicação que se realiza ao redor destas representações e para capturar esta comunicação relacionada ao artefato em evolução juntamente com suas representações.

EVA suporta a criação de um novo tipo de artefato de software que integra documentação baseada em hipertextos com protótipos. EVA provê um tipo de ligação executável que, quando selecionada, ocasiona a execução de um protótipo do sistema. Usando EVA, usuários e desenvolvedores podem criar uma documentação baseada em hipertexto descrevendo a aplicação e seus requisitos: desenvolvedores constroem protótipos que serão embutidos no contexto apropriado da documentação; e usuários podem ter acesso e interagir com os protótipos e registrar seus comentários e críticas ao hipertexto. Documentação e protótipos são modificados de acordo com as necessidades e o processo iterativo tem continuidade. O processo de construção do sistema é, portanto, direcionado pela comunicação entre desenvolvedores e usuários e suportado pelo artefato de software em evolução.

O ambiente *Beyond-Sniff* [BISC94] permite que as anotações geradas por seus usuários, relacionadas aos artefatos de software, sejam organizadas numa base hipertextual. Desta forma, a memória de grupo preserva os artefatos sendo produzidos e a comunicação informal através de anotações.

### 5.1.2 Modelos de Argumentação

Aliado aos recursos de organização e navegação oferecidos pela tecnologia de hipertextos, é imprescindível prover recursos para captura da memória coletiva. Esta captura têm sido comumente realizada através do registro de justificativas de projeto baseado em modelos de argumentação.

Este é o caso do ambiente CPCE (*Collaborative Process-Centered Environment*) [CPCE94], da ferramenta Quorum [ARAU94], do ambiente ARCoPAS [CAVA94] e do ambiente para projeto participativo de Cavalcanti e Borges [CAVA96]. No CPCE, que é um ambiente centrado em controle de processos, a encenação do modelo de processo se baseia num modelo de discussão que é uma extensão do modelo sugerido por Potts. O ambiente proposto por Cavalcanti e Borges utiliza a técnica do *Inquiry Cycle Model* que se baseia em questões, respostas a questões e razões para as respostas. Questões propõem dúvidas ou problemas quanto ao conteúdo de documentos. As respostas provêem respostas e soluções às questões propostas. E as razões compreendem argumentos ou justificativas para as respostas. A ferramenta Quorum e o ambiente ARCoPAS propõem o uso do modelo de argumentação IBIS.

Segundo Twidale, Rodden e Sommerville [TWID93], o registro de discussões e justificativas de projeto (*design rationale*) através de modelos de argumentação têm sido uma das principais abordagens para o suporte ao projeto de software. Mas, uma série de problemas é levantada com o uso desta abordagem de suporte, no que diz respeito a dificuldades de aceitação por parte dos usuários, por os obrigarem a decidir previamente a natureza de cada contribuição que lançarem.

Os estudos realizados no decorrer do desenvolvimento da ferramenta DNP (*Designers' Notepad*) [TWID93], confirmam que há uma grande variação das atividades de projeto tanto entre usuários e pelos mesmos usuários ao longo do tempo e de acordo com as circunstâncias. Esta variação mostra a necessidade de dotar a ferramenta com flexibilidade para suportar esta variação. No caso do DNP, esta flexibilidade está em evitar a delegação de semântica a entidades e ligações entre elas, permitindo que sejam utilizadas livremente pelos usuários nos estágios iniciais do projeto, quando ainda não se pode exigir muita formalização de conceitos.

Os atributos das entidades e ligações são utilizados como mecanismos para suportar a evolução desta estrutura de conceitos. Esta liberdade permite que a estrutura de conceitos apropriada surja durante o projeto após as entidades terem sido levantadas e criadas. Isto contrasta com

outros sistemas que possuem um conjunto de tipos de entidades pré-definidas e que requerem que o usuário selecione o tipo correspondente à contribuição que pretendem criar.

### 5.1.3 Gerência de Configuração

A evolução dos componentes de software gerados, de acordo com as decisões tomadas pelos membros da equipe de desenvolvimento é preocupação constante dos ambientes que se propõem a suportar o processo de construção de software. A quase totalidade das abordagens para controle da configuração do software ou controle de versões, tendem para uma estrutura onde são armazenadas as alterações consecutivas a um determinado componente de software, facultando aos participantes visualizar a versão de um determinado componente no tempo com maior flexibilidade.

Algumas abordagens, seguindo a filosofia da necessidade de manter uma memória de grupo completa, registram, além das evoluções, os motivos que levaram às alterações.

A linguagem VTML (*Versioned Text Markup Language*) [VITA96] prentende ser uma linguagem de marcação de textos para controle de versões de documentos. Contudo, não registra os motivos das alterações, apenas a evolução dos produtos.

O ambiente EPOS [EPOS96] possui a memória de grupo baseada em versões de componentes de software. O mecanismo para controle de versões em EPOS é denominado COV (*Change Oriented Versioning*). Em COV, uma versão não é considerada como um objeto explícito mas como o resultado de determinadas mudanças que foram aplicadas à base de dados. Nem todas as alterações são incluídas numa versão escolhida, apenas a combinação daquelas especificadas em opções definidas pelo usuário. Opções seriam valores lógicos que determinam a visão da base que o usuário deseja recuperar.

A mesma estratégia segue o ambiente proposto por Magnusson e Guerraoui para desenvolvimento colaborativo de software orientado a objetos [MAGN96]. O histórico de alterações de um programa é armazenado através de estruturas hierárquicas denominadas grafos de evolução, que descrevem as versões do sistema e os deltas relativos às alterações realizadas entre versões. Um grafo de evolução contém resultados de várias ações: criação de revisões, criação de alternativas a versões e fusão de alternativas. Um grafo de evolução indica, portanto, o estado de um programa. Usuários podem visualizar as edições realizadas por outros usuários e quando foram realizadas, através do grafo de evolução.

O ambiente PROSOFT [REIS96] baseia a gerência de configuração no modelo de ramificação *check in/check out* onde uma versão é um objeto composto pelo objeto original e as modificações ocorridas.

O *Beyond-Sniff* [BISC94] também baseia o registro da evolução de seus artefatos de software em mecanismos tradicionais de gerência de configuração. Oferece uma ferramenta *TurboMixer* que provê suporte à comparação e fusão de projetos e cópias de projetos.

## 6. Suporte à Coordenação

> *"Coordination is the integration or linking together of different work units within an organization to accomplish a collective set of tasks."*
>
> Hayward P. Andres [ANDR95]

Construir software cooperativamente exige um gerenciamento constante das atividades sendo realizadas pelo grupo como um todo e daquelas realizadas individualmente por cada participante. É preciso, portanto, prover suporte para a execução das tarefas do grupo e das tarefas individuais de cada membro através de mecanismos de definição, visualização e controle do encaminhamento do processo.

É preciso prover recursos para que todos os membros tenham a noção do contexto de suas atividades dentro do contexto geral do processo, para que consigam perceber o andamento das atividades sendo realizadas por outros membros e para compreender como os resultados gerados pelas atividades alheias podem ser conjugados aos seus para chegarem mais rapidamente ao resultado final.

A palavra-chave relacionada à coordenação, portanto, refere-se a **acompanhamento**. Especificar como a interação se dará, definir regras e limites, estipular responsabilidades e controlar a execução de tarefas são questões que precisam ser suportadas durante o processo.

## 6.1 Abordagens para Suporte à Coordenação

O suporte à coordenação de atividades parece ser a questão de maior ênfase no que se refere à pesquisa em projeto cooperativo de software. Definição de responsabilidades, modelagem e controle de processos e definições de mecanismos de controle de concorrência são os ítens que os ambientes cooperativos de desenvolvimento mais focalizam como recursos para coordenação.

A razão para a ênfase em controle de execução de atividades pode estar na crescente importância que as técnicas de modelagem e controle de processos têm apresentado no contexto de suporte ao desenvolvimento de software, com o surgimento de ambientes centrados em processos.

### 6.1.1 Políticas de Cooperação baseadas em Responsabilidades

Em todos os ambientes e ferramentas pesquisados, há a preocupação na modelagem de usuários e delegação de responsabilidades aos diferentes tipos de usuários modelados.

O ambiente OZWeb [OZWE96] se preocupa em modelar explicitamente os usuários como grupos de usuários através de suas características, responsabilidades e permissões em relação ao processo. Assim também a ferramenta Quorum [ARAU94] e o ambiente ARCoPAS procuram classificar usuários em papéis com responsabilidades distintas.

O ambiente EPOS (*Expert System for Program and System Development*) [EPOS96] e o ambiente *Conversation* Builder [KAPL92] também associam usuários à execução de atividades do processo mas não parecem se preocupar com a modelagem de usuários em papéis.

### 6.1.2 Modelagem e Encenação de Processos de Software

Esta é uma questão de grande discussão em relação a ambientes de suporte ao projeto cooperativo de software. Muitas são as propostas para prover a modelagem e encenação de processos de software levando em consideração o aspecto cooperativo do desenvolvimento.

Modelos de processo descrevem o conhecimento utilizado para governar a execução de processos e checar sua consistência e correção. Em desenvolvimento de software, o uso de modelos de processo tradicionais torna-se inadequado, haja visto as distinções entre os processos de software e os processos industriais tradicionais. Os processos que suportam a construção de software representam uma atividade altamente intelectual, criativa, que envolve vários personagens humanos e pode ser apenas parcialmente automatizada [ARME92]. Além disso, compreendem um processo evolutivo onde não só os produtos gerados sofrem alterações constantes ao longo do processo como o próprio processo é passível de alterações em sua definição e execução.

Consequentemente, não se pode garantir a existência de um modelo de processo universal para suporte ao desenvolvimento de software mas sim, devemos nos inclinar a tendência em prover processos customizados para cada contexto de projeto. As linguagens para modelagem e representação de processos têm um papel importante para prover esta customização.

Para atender adequadamente as necessidades de suporte de processos de desenvolvimento, as linguagens de representação de processos deveriam apresentar características que as

permitissem descrever elementos incompletos, ambíguos, informais ou formais e deveriam ser capazes de modelar a intervenção humana no processo e os eventos de ativação de ferramentas. Além disso, deve ser suficientemente flexível para permitir a modificação dinâmica do processo enquanto este estiver sendo encenado. O ideal, ainda, seria se as linguagens de representação de processos pudessem representar toda a produção do software, incluindo tanto as atividades técnicas como as gerenciais.

O ambiente SPADE (*Software Process Analysis, Design and Enactment*) [BAND93] define a linguagem SLANG (*Spade Language*), baseada em Redes de Petri de alto nível. Em SPADE, as atividades podem ser instanciadas dinamicamente durante a encenação do processo e um modelo de processo definido em SLANG pode incluir a descrição de interações entre usuários e ferramentas. Esta modelagem explícita da interação permite guiar e controlar a cooperação durante a encenação do processo para assegurar sua consistência. A semântica da interação entre ferramentas e indivíduos está contida na descrição do processo.

O ambiente OZWeb [OZWE96] permite que qualquer paradigma de modelagem de processos possa ser utilizado. Os modelos de processo são encenados em uma máquina de *workflow* oferecida pelo ambiente. O ambiente permite que usuários sejam associados dinamicamente a atividades, permitindo, inclusive, a delegação de atividades. Há ainda a preocupação em tolerar inconsistências, operar com informações incompletas e permitir fragmentações no processo.

O projeto do ambiente OZWeb apresenta, ainda, um modelo para ambientes descentralizados centrados em processos. Este modelo procuraria suportar times dispersos geograficamente, onde cada time é gerenciado por seu próprio processo [BEN-94]. A tentativa é a de explorar a modelagem e a encenação de colaboração entre grupos por processos independentes, autônomos e possivelmente pré-existentes, diferentemente do suporte intra-grupos comumente oferecido por ambientes.

Esta proposta faz distinções entre distribuição e descentralização, enfocando o último aspecto, que diz respeito a subsistemas relativamente independentes com algum grau de correlação entre eles, geralmente dispersos por vários sítios. A transparência de uso dos recursos não situados localmente é intencionalmente não suportada por violar a autonomia entre os subsistemas.

Dos vários requisitos necessários para o suporte a tal estrutura de colaboração, dois merecem destaque na proposta: a autonomia na execução de processos e a colaboração e interoperabilidade entre processos, que passam a ser atendidos através de dois protocolos definidos para o modelo (*Summit Protocol* e *Treaty Protocol*).

O ambiente EPOS [EPOS96] se preocupa com o suporte à modelagem, simulação, encenação de processos de desenvolvimento de software e modelagem de meta-processos.

EPOS provê um ambiente de suporte ao processo contendo uma metodologia de modelagem, bibliotecas de modelos de processo reutilizáveis e ferramentas para aquisição, modelagem, análise, evolução, execução e controle do processo.

Uma linguagem de modelagem de processos foi definida especificamente para o ambiente EPOS - SPELL (*Software Process Evolutionary Language*). SPELL é baseada num modelo de objetos contendo as classes que descrevem as entidades envolvidas na modelagem do processo. Este modelo está integrado ao modelo de classes da base de dados do ambiente.

O controle do processo é realizado por ferramentas de encenação do processo definido em SPELL. O controle se baseia nas definições da cadeia de tarefas do processo como: pré e pós condições para realização de atividades, decomposição de atividades, definições quanto a execução automática ou manual das atividades, execução periódica, oportunística ou tardia etc.

O CPCE (*Collaborative Process-Centered Environment*) [CPCE96] tem o objetivo de aplicar a abordagem e tecnologia de modelagem de processos a uma classe de aplicações cooperativas representadas pelas aplicações assíncronas focalizadas em tarefas com um certo nível de estruturação. O desafio está em lidar com entidades e interações de granularidade mais fina do

que as comumente consideradas em ambientes centrados em processos e promover maior adaptabilidade e flexibilidade.

A granularidade fina é atendida pelo fato do processo se basear num modelo de discussão onde questões, posições e argumentos são apresentados em relação a artefatos sendo manipulados pelos participantes ou em relação aos passos definidos para a execução da atividade em andamento. Portanto, processo e meta-processso se confundem numa mesma estrutura de argumentação. Modelos de processos, histórias e justificativas incluem vários objetos em diversas granularidades.

Quanto à adaptabilidade, processos são especializados (instanciados) a partir do modelo geral de processos e podem ser customizados para atender à requisitos específicos de determinados contextos de colaboração.

Por fim, a flexibilidade prevê alterações no processo sendo realizadas pelo levantamento de questões quanto ao meta-processo. O ambiente insinua a possibilidade de suportar a alteração de processos através de evoluções de esquema da base de dados.

O ambiente Coo (*Cooperation & Coordination)* [CANA94] tem o objetivo de construir uma estrutura para suporte a processos de desenvolvimento de software, focalizando, particularmente, na coordenação e na cooperação consistente entre diferentes tarefas e atividades envolvidas no processo.

O modelo de processos de Coo define que um processo de software pode ser modelado como uma árvore onde os nós internos representam tarefas a serem realizadas que podem ser decompostas em sub-tarefas e onde as folhas representam atividades que não podem ser decompostas.

Cada tarefa ou atividade possui seu próprio espaço de trabalho que contém o conjunto de objetos passíveis de serem manipulados pela tarefa. Este espaço de trabalho pode ser classificado em público, privativo ou semi-público [GODA94].

O conceito de cooperação entre processos em Coo é visto como a possibilidade de processos interagirem enquanto são executados. Neste contexto, um espaço de trabalho é associado a cada transação, resultando em uma hierarquia de espaços de trabalho similar à hierarquia de processos, e cada processo pode cooperar com outros pela transferência de objetos de/para seu espaço de trabalho para/de outros espaços de trabalho.

No ambiente Coo, cada processo é executado como uma transação aninhada. A execução de um processo leva a uma árvore de transações onde cada nó interno corresponde a tarefas (processos que podem ser decompostos em subprocessos) e folhas correspondem a atividades (processos que não podem ser subdivididos). Atividades são os únicos processos permitidos a modificar objetos. Tarefas são utilizadas somente para sincronizar seus sub-processos.

Em Coo, a serialização da execução do processo só é requerida para atividades. Tarefas não precisam ser necessariamente executadas em série. Isto significa que uma tarefa pode ver resultados intermediários de outras tarefas, sem que esta tenha chegado ao fim.

As restrições relativas a um processo são checadas quando resultados são retornados de processos inferiores. Um resultado que viola uma das restrições não pode ser retornado. Contudo, um processo pode retornar resultados intermediários que não estariam sujeitos a restrições. Resultados intermediários permitem que sub-processos diferentes interajam e negociem objetos antes de ter sua execução terminada. Com este mecanismo, os criadores do Coo pretendem permitir a cooperação entre processos (e consequentemente entre desenvolvedores que executam o processo).

Uma característica central do ambiente Coo é uma consequência de experiências de projeto anteriores que demonstram que existem limites para uso de abordagem puramente baseadas em conhecimento para controlar a interação entre desenvolvedores: é impossível prever, e portanto modelar, todas as interações possíveis entre desenvolvedores. Como consequência, é necessário

basear a encenação do processo num modelo de transação seguro que, no momento em que o conhecimento sobre o processo seja insuficiente, possa definir um modo de sincronização padrão para sua encenação [GODA94].

Os modelos de transação tradicionais, por se basearem na serialização de atividades, implicam em interações hostis ao invés de interações cooperativas, além de não suportar de forma satisfatória processos longos, conforme requerido por processos de desenvolvimento de software.

Quando há conflitos entre processos (conflitos de acesso a objetos), são iniciadas discussões entre os agentes humanos associados ao processo em conflito. Todos os agentes direta ou indiretamente envolvidos com esta tarefas são potencialmente indicados para a negociação. Então, a organização hierárquica de processos pode ser utilizada para organizar as negociações.

O ambiente *Conversation Builder* [KAPL92] possibilita a especificação de múltiplos protocolos para coordenação de atividades, permitindo que os usuários trabalhem com várias instâncias de protocolos de acordo com a atividade que realizem.

Atividades realizadas no ambiente são consideradas como "conversas", seguindo a filosofia de Winograd e Flores de "conversa para ação". Um protocolo de conversação provê um conjunto de regras que governam o comportamento das conversas instanciadas sob aquele protocolo, tais como as atividades que podem ser realizadas, por quem serão realizadas, sob quais circunstâncias e quais seus efeitos. Um protocolo pode ser encarado como uma forma de programa de processo.

O ambiente RASP (*Regatta Automated Software Process*) [SWEN93] foi desenvolvido para suportar a automação de processos de software e sua reengenharia permitindo que membros de equipes tomem parte do planejamento e reengenharia do processo.

RASP está baseado num modelo de negociação para promover a interação entre os agentes humanos do processo, coordenando-o. Usuários podem realizar pedidos a qualquer outro membro da equipe. Este membro pode aceitar ou declinar da responsabilidade de realizar a atividade requisitada. Se aceita, então, é esperado que esta tarefa seja realizada por este membro em um momento posterior. A aceitação da realização de uma atividade pode iniciar um sub-processo que será definido pelo "dono" da atividade/processo.

Usuários criam políticas (modelos de processos) utilizando uma ferramenta gráfica para construção de diagramas de processos. Cada usuário cria o esquema do serviço/processo que executa. Estes diagramas são compostos de estágios, que representam passos dentro da política de trabalho que devem ser realizados pelo usuário ou por outras pessoas. A coordenação do grupo é realizado através do controle de execução das políticas de cada usuário.

RASP é ciente da estrutura da organização. Isto significa que as políticas utilizadas por um usuário dependem das políticas contruídas por ele, ou das políticas definidas para o grupo ao qual pertence e dependem, ainda, das políticas da organização. As políticas definidas para a organização podem ser adaptadas conforme necessário para atingir a forma de trabalho de cada usuário ou de determinados grupos.

RASP não possui o objetivo de automatizar o trabalho de indivíduos. Pessoas diferentes irão utilizar ferramentas diferentes para seus trabalhos. Por ser difícil e até impossível modelar as atividades num nível mais baixo (atividades realizadas por cada indivíduo), RASP se preocupa em modelar o comportamento interpessoal que, apesar de ser também complexo, é mais fácil de ser observado e levantado. Desta forma, RASP se preocupa em modelar a comunicação entre usuários e flexibiliza a exeucação de suas tarefas permitindo que utilizem as ferramentas de trabalho que mais lhes aprouverem.

RASP evita o problema da complexidade na modelagem de processos por não exigir que os modelos estejam completos. Quando há a ocorrência de uma exceção, o usuário pode sobrescrever o processo.

Para Sommerville e Rodden [SOMM93], uma abordagem efetiva para a modelagem de processos de software não surgirá até que se tenha uma compreensão clara das influência sociais e organizacionais neste processo. Sugerem ainda que é provavelmente prematuro incorporar facilidades explícitas de modelagem de processo em ambientes de suporte.

### 6.1.3 Controle de Concorrência

O modelo tradicional de controle de concorrência em ambientes para múltiplos usuários baseia-se nos conceitos de transação e chaveamento. O problema destas abordagens, quando utilizada em ambientes para suporte a colaboração, está em colocar uma visão particular da cooperação. Esta visão de controle rígido é muitas vezes inaceitável dada a riqueza de padrões de interação em desenvolvimento cooperativo de software. Em muitos momentos do processo de desenvolvimento, por exemplo, um usuário precisa ter conhecimento de que um dado componente está sendo utilizado por outro membro da equipe e pode ainda querer ver o que este membro executa sobre o componente, como se estivesse "olhando por sobre seus ombros" [SOMM93].

Um requisito importante para ferramentas cooperativas é o de que o uso por múltiplos usuários deve ser explícito ao invés de prevenido ou "disfarçado". Isto requer um modelo de controle de acesso a informações e ferramentas diferente dos modelos tradicionais.

O ambiente EPOS [EPOS96] busca oferecer suporte para planejamento, levantamento e tratamento de conflitos de concorrência, ao invés de evitar que ocorram, colocando restrições à forma de trabalho em grupo. Funcionalidades foram adicionadas à base de dados para permitir um acesso mais flexível e uma série de ferramentas para introduzir e recuperar informações foram implementadas como, por exemplo, uma ferramenta de apoio a resolução de conflitos, fusão de versões, sincronização de versões, revisão de componentes etc.

No ambiente Coo [CANA94], cada tarefa/atividade possui a definição de seu espaço de trabalho, contendo o conjunto de objetos que pode ter acesso.

O PROSOFT [REIS96] é um ambiente de desenvolvimento de software que tem como objetivo suportar o desenvolvimento formal de programas, fornecendo integração de dados, controle e apresentação entre ferramentas construídas no ambiente. Como extensão ao ambiente PROSOFT, está sendo estudado a especificação de recursos para o suporte ao desenvolvimento cooperativo, dotando o ambiente de uma arquitetura cliente/servidor onde ferramentas compartilham e gerenciam objetos. Estas ferramentas são denominadas Ambientes de Tratamento de Objetos (ATOs) e estão interligadas através da Interface de Comunicação do Sistema (ICS). Cada ATO é descrito por sua classe e um conjunto de operações para criar, manipular e visualizar os objetos que pertençam a esta classe.

O ambiente ARCoPAS [CAVA94] possibilita a existência de cópias individuais da base de informações compartilhadas, onde os membros da equipe de projeto podem realizar alterações e incluir seus comentários pessoais sem alterar a base comum.

O *Beyond-Sniff* [BISC94] oferece controle otimista e pessimista para acesso aos artefatos armazenados. O controle pessimista significa que o acesso é controlado através de mecanismos de chaveamento. O controle otimista permite que os desenvolvedores trabalhem autonomamente em cópias do projeto que serão fundidas posteriormente, gerando uma nova versão do mesmo.

O ambiente proposto por Magnusson e Guerraoui [MAGN96] buscam se libertar do mecanismo de chaveamento de objetos propondo um protocolo denominado *wait-free* onde um usuário pode editar uma nova versão de um programa sem ter que aguardar pelo estabelecimento de uma comunicação remota seja qual for o estado do sistema. Mais tarde, as versões editadas por usuários distintos podem ser fundidas. Mas, inconsistências e conflitos devem ser tratados pelos desenvolvedores. Em alguns casos, podem ser definidas linhas "protegidas", ou seja, versões do software que só podem ser editadas por um grupo de usuários pré-definido.

## 7. OUTRAS QUESTÕES

Outros recursos surgem como importantes para categorizar as tendências em suporte automatizado ao projeto cooperativo de software. Embora algumas destas questões estejam relacionadas às questões mencionadas na seção anterior, merecem um destaque especial por se tratarem de elementos relevantes para a construção de ferramentas cooperativas em geral. Gostaríamos, portanto, de observar questões quanto: à arquitetura dos ambientes, suas plataformas de implementação e às fases ou atividades do processo de desenvolvimento que as ferramentas se propõem a suportar.

### 7.1 SUPORTE A ATIVIDADES DO PROCESSO DE DESENVOLVIMENTO

As atividades básicas para construção de software - especificação de requisitos, projeto, implementação, verificação/validação e evolução - apresentam variações significativas no que diz respeito à natureza das interações que ocorrem entre os responsáveis por cada uma delas, exigindo suportes com características diferenciadas [GIBB89][SOMM93].

A atividade de especificação, por exemplo, envolve necessidades marcantes de interação, comunicação e entendimento por ser o momento onde inconsistências deveriam ser resolvidas e o desenvolvimento de conhecimento inicial comum é realizado. As atividades de projeto e implementação envolvem trabalho criativo em conjunto, onde é necessário o suporte para geração de idéias, críticas e organização das informações criadas. Nestas atividades, abordagens menos rígidas e estruturadas para o suporte à interação parecem ser mais promissoras. Já as atividades de verificação, testes e validação requerem coordenação e controle em primeiro lugar. Por fim, a manutenção ou evolução contínua de um produto de software exige a existências de estruturas para captura, organização, gerência e reuso do conhecimento informal criado a cada nova alteração no produto, compondo a memória organizacional daquele contexto de desenvolvimento.

Na literatura pesquisada, encontramos descrições de ambientes que se preocupam em oferecer suporte ao processo de colaboração ao longo de todo o processo, englobando todas as atividades básicas, a saber: CPCE [CPCE94], Coo [CANA94], OZWeb [OZWE96], SPADE [BAND93], EPOS [EPOS96], EVA [OTSW95] e *Conversation Builder* [KAPL92].

Outras ferramentas foram desenvolvidas para o suporte a atividades específicas, como:

- Levantamento e análise de requisitos: VORD [VORD96], a proposta ambiente para projeto participativo de Cavalcanti e Borges [CAVA96] e *Conflict-Resolution Supporting Tool* [IOCH95].
- Inspeção de Software: CSI (*Collaborative Software Inspection*) [MASH93]
- Projeto: CSD [CSD96] e a proposta de ambiente para projeto participativo de Cavalcanti e Borges [CAVA96]
- Codificação: CSD [CSD96], *Beyond-Sniff* [BISC94], Flecse [DEWA93], ambiente proposto por Magnusson e Guerraoui [MAGN96]
- Recuperação de Projeto (Engenharia Reversa): ARCoPAS [CAVA94]

Por fim, algumas ferramentas se propõem a suportar tipos particulares de interações que ocorrem ao longo deste processo, como:

- Suporte a discussões ao longo do desenvolvimento: ICARO [LICE96] e Quorum [ARAU94]
- Suporte ao processamento de idéias: DNP (*Designers' Notepad*) [DNP96]
- Suporte à coordenação de atividades: RASP (Regatta Automated Software Process) [SWEN93]

## 7.2 ARQUITETURAS E PLATAFORMAS

A maioria dos ambientes para suporte ao desenvolvimento de software baseiam o seu suporte à cooperação no compartilhamento de repositórios de informações e em sistemas de gerência de configuração das informações aí armazenadas. Desta forma, as informações sobre o projeto tornam-se acessíveis a todos os membros da equipe (com algumas restrições devido ao controle de acesso) e é assegurado o trabalho independente de cada participante sobre um conteúdo comum.

Portanto, por um bom tempo, ambientes de desenvolvimento de software foram vistos como um repositório centralizado de informações associado a uma estrututra de integração de ferramentas que utilizam este repositório e seus serviços. A estruturação de ambientes de desenvolvimento nesta filosofia de integrador de ferramentas permite o compartilhamento de entidades de projeto entre múltiplas ferramentas (e por trás delas, usuários), conforme adotado pelos ambientes: SPADE [BAND93], OZWeb [OZWE96], Conversation Builder [KAPL92] e PROSOFT [REIS96].

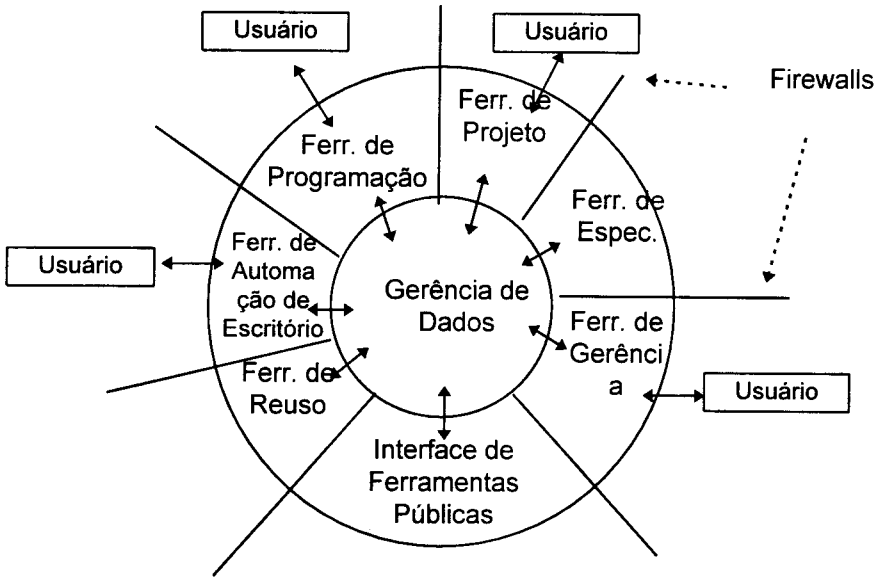


Figura 1 - Cooperação num ADS com repositório de dados centralizado [SOMM93]

Contudo, esta arquitetura baseada em repositórios de software e ferramentas integradas através deste repositório auxiliam a controlar o processo mas, segundo Sommerville e Rodden [SOMM93], não são tão úteis para promover suporte apropriado ao processo de cooperação no contexto de desenvolvimento de software. De forma a proteger as informações concentradas no repositório, o gerenciador da base de software limita a interação entre as ferramentas. O gerente do repositório de software encoraja a comunicação entre usuários através da base de informações e tanto as ferramentas como seus usuários parecem estar protegidos por *firewalls* que inibem a interação direta entre eles (Figura 1).

A infraestrutura de ambientes de desenvolvimento se focalizou em prover este repositório centralizado e em delegar ao ambiente o controle a este repositório através de políticas de uso. Estas políticas determinariam o protocolo para interação das ferramentas com o repositório de dados. Desta forma, era inevitável que ambientes de desenvolvimento se orientassem em direção ao controle do processo.

A filosofia de ambientes centrados num núcleo provedor de informações envolve um conjunto de suposições de como uma equipe de projeto deve trabalhar. Estas suposições falham em

considerar os aspectos cooperativos do desenvolvimento, no que se refere à modelagem da informação e aos mecanismos de suporte ao seu compartilhamento.

Uma visão alternativa de arquiteturas de ambientes de desenvolvimento de software, apresentada pelo ambiente COSIE (*Cooperative Systems Integration Environment*) [SOMM93] e *Beyond-Sniff* [BISC94], está em considerar o ambiente como um conjunto de componentes (ou gerentes) fracamente conectados. Estes componentes se comunicam através de um protocolo comum (gerente de comunicações), permitindo uma maior interconexão entre gerentes. Componentes fazem pedidos de serviços e o gerente de comunicações é responsável por descobrir quais componentes podem responder ao serviço (Figura 2).

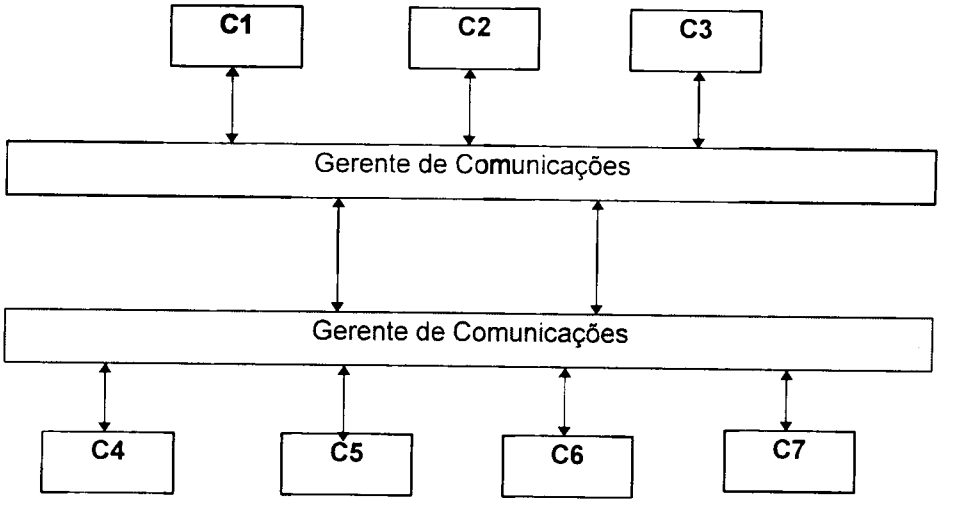


Figura 2 - Arquitetura baseada em gerentes de comunicação [SOMM93]

Esta arquitetura oferece vantagens ao suporte à cooperação pois vários gerentes autônomos podem ser projetados para prover recursos específicos de cooperação. Dado que os mecanismos que os gerentes utilizam para comunicação estão definidos claramente em termos de um protocolo, o conjunto de componentes pode ser facilmente estendido. Os serviços de armazenamento de informações são integrados como componentes e seguem o mesmo princípio de integração, deixando de serem centralizados.

O ambiente proposto por Magnusson e Guerraoui [MAGN96], por sua vez, está organizado sobre uma arquitetura de múltiplos clientes e múltiplos servidores, na intenção de flexibilizar o mecanismo de controle de concorrência aos artefatos de software.

### 7.2.1 Plataformas

Quanto a plataformas de implementação de ambientes de projeto cooperativo de software, vemos (como em toda a área de pesquisa em *groupware*) uma grande tendência para o uso da plataforma Internet/WWW usando diretamente o protocolo HTTP, como em ICARO [LICE96] ou lançando mão de *proxy servers* ao HTTP como em OZWeb [OZWE96].

Outras plataformas são utilizadas como: plataforma DEC-FUSE para integração de ferramentas [BAND93], Unix [SWEN93] [MAGN96], Internet/Unix [CSD96], FolioViews [CAVA94] e Lotus Notes [ARAU94].

## 7.3 ARMAZENAMENTO

Procuramos observar, na propostas de ambientes cooperativos de software, as soluções utilizadas para o armazenamento das informações em bases de dados.

Surpreendentemente (ou não), a maioria das propostas estudadas sugerem o uso de bases de objetos como repositório de informações.

O ambiente SPADE [BAND93] e o ambiente CPCE [CPCE94] apostam em bancos de dados orientados a objetos comercias como: $O_2$ e GemStone, respectivamente.

Outros ambientes construíram seus próprios gerenciadores de objetos como o OZWeb [OZWE96], EPOS [EPOS96] e Coo [CANA94].

O ambiente *Conversation Builder* [KAPL92] usa um mecanismo para persistência de objetos mas, planejavam a evolução para o uso de bancos de dados orientados a objetos.

O ambiente proposto por Magnusson e Guerraoui [MAGN96] armazena todas as informações sobre a configuração do sistema em construção em arquivos Unix utilizando um formato especial (Treefile).

### 7.3.1 A Questão da Definição de Esquemas de Bases de Software

Esquemas de bases de software têm sido tradicionalmente vistos como descrições estáticas de dados. Mas, estes mesmos esquemas necessitam se tornar mais dinâmicos se pretendem suportar apropriadamente a natureza das interações dos grupos que suportam.

A existência de um esquema pressupõe a existência de uma elemento humano privilegiado e centralizador responsável pela definição das informações contidas no esquema. Esta figura privilegiada torna-se imprópria numa abordagem cooperativa onde, em alguns contextos de interação, não se espera a existência de tais privilégios.

Mas, esta flexibilidade de definição de esquemas está intimamente relacionada com a possibilidade de prover sua evolução com o tempo, o que ainda não pode ser realizado a baixos custos.

## 8. Conclusão

Neste trabalho, procuramos descrever as abordagens utilizadas pelas ferramentas e ambientes que se propõem a apoiar os aspectos cooperativos que envolvem o projeto de software. Para isto, procuramos classificar as abordagens segundo quatro questões que consideramos como básicas para o apoio ao projeto cooperativo de software: o suporte ao entendimento entre participantes, a oferta de canais de comunicação, a abordagem de registro da memória de grupo e os protocolos e padrões de coordenação. Um resumo das abordagens pode ser visualizado na

Tabela 1.

Podemos observar, em primeiro lugar, a existência de dois ramos de pesquisa na área de projeto cooperativo de software. Um deles diz respeito à construção de ferramentas isoladas, voltadas para o suporte a aspectos específicos ou a fases isoladas do processo de construção de software. O outro ramo focaliza a construção de ambientes cooperativos, com ênfase na definição de um núcleo principal de controle de atividades através de abordagens de modelagem de processos, onde ferramentas podem ser acopladas e seu uso gerenciado.

Esta divisão de esforços é proveitosa, na medida em que há a necessidade sempre crescente de customização e flexibilização de ambientes de desenvolvimento de software, e que pode ser atingida quando se dispõe de uma máquina de controle de processos passiva ao uso de um conjunto considerável de ferramentas.

O primeiro ramo de pesquisa tende a ter uma maior preocupação com aspectos de interface, comunicação e coleta da memória de grupo; enquanto que o segundo se concentra em aspectos de comunicação, registro da memória de grupo e coordenação de tarefas.

Mas, apesar desta divisão proveitosa de esforços, pois se imagina que, posteriormente, os resultados e experiências dos ramos de pesquisa possam ser integrados num ambiente "ideal", muito pouco é apresentado quanto a resultados de uso das abordagens sugeridas.

Seguindo esta linha, a pesquisa se inclina para proposta de ambientes cujos suportes abranjam mais amplamente o processo de desenvolvimento. Novamente, poucos resultados são apresentados de que um ambiente integrado para o suporte a todo o processo melhore a sua qualidade. Contudo, algumas abordagens sugerem a possibilidade de alterações dinâmicas ao gerenciamento das atividades realizadas no ambiente e outras se preocupam com questões relativas a modelagem de meta-processos, o que, se por um lado flexibiliza bastante a idéia de ambiente integrados, ainda parece ser uma utopia a níveis operacionais.

Outra observação importante está na no reconhecimento da necessidade de abordagens de suporte ao entendimento entre participantes no que se refere a definição e formalização de conceitos. A quase totalidade das ferramentas analisadas, voltadas para suporte a fases específicas do processo de desenvolvimento, se concentram neste objetivo. Mas, os resultados, como de praxe, ainda não são conclusivos quanto ao seu real uso.

Quanto ao aspecto de canais de comunicação, percebemos as conferências como principal recurso utilizado pelas ferramentas/ambientes. Consideramos como conferência as interações que envolvem a comunicação remota ou local, síncrona ou assíncrona entre indivíduos acerca de um foco comum, seja este um espaço compartilhado de trabalho ou uma base de discussões.

As metáforas de percepção e interação nestas conferências seguem as propostas sugeridas na área de CSCW em escopo geral, mas estudos sobre a necessidade ou não de definição de metáforas específicas para interação entre grupos de desenvolvimento ainda precisam ser mais elaborados.

As ferramentas/ambientes seguem abordagens bem próximas para o gerenciamento da memória de grupo, realizando o registro em bases hipertextuais e focalizando a gerência de configuração. A maioria também reconhece a necessidade de registro da memória informal. Mas, ainda há o que se discutir em relação aos modelos de argumentação utilizados até então como forma de captura do conhecimento informal.

O suporte à cooperação está massivamente concentrado na modelagem e encenação de processos e modelagem de usuários. Evidencia-se contudo, a necessidade de atender a questões específicas à modelagem e encenação de processos de software, principalmente no que se refere à dinamicidade do processo. As abordagens para modelagem e encenação parecem seguir as propostas tradicionais de linguagens de modelagem mas, com recursos específicos voltados para o suporte à interação de grupos e para as atividades de desenvolvimento de software.

Concluíndo, propostas para o apoio ao projeto cooperativo de software estão surgindo, dada a sua necessidade, são potencialmente promissoras, mas ainda há muito o que percorrer em relação a resultados concretos de sua utilização em projetos reais e de longa duração.

| Ferramenta/ Ambiente | Escopo de Suporte ao Processo | Recursos de Suporte Oferecidos | | | |
|---|---|---|---|---|---|
| | | Entendimento | Comunicação | Memória de Grupo | Coordenação |
| *Magnusson e Guerraoui* | Projeto e Codificação | | | • Gerência de Configuração | •Controle de Concorrência |
| *Flecse* | Codificação, Inspeção e Testes | | •Conferências | | •Controle de Concorrência |
| *ARCoPAS* | Recuperação de Projeto Arquitetônico (Engenharia Reversa) | | •Conferências | •Registro de argumentação<br>•Hipertexto com estrutura modular de sistemas | •Modelagem de Usuários |
| *EVA* | Suporte global às atividades do processo | •Análise de protótipos e documentação de projeto | | •Hipertexto de documentos de software e protótipos | |
| *Beyond-Sniff* | Codificação | | •Mecanismo de anotações | •Registro de anotações<br>•Hipertexto de anotações e artefatos de software<br>•Gerência de Configuração | •Controle de Concorrência |
| *RASP* | Suporte à coordenação de atividades | | •Sistema de Mensagens | •Registro da execução de processos | •Modelagem e encenação de processos (workflow) |
| *Cavalcanti & Borges* | Levantamento/Refinamento de Requisitos | •Definição de Cenários | •Conferências | •Registro de argumentação<br>•Gerência de Configuração | |
| *CRST* | Levantamento/Refinamento de Requisitos | •Identificação e Resolução de Conflitos de linguagem | •Conferências | •Base de Conceitos | |
| *DNP* | Levantamento/Refinamento de Requisitos | •Processamento de Idéias | •Conferências | •Registro de diagramas de conceitos<br>•Registro de argumentação | |
| *VORD* | Levantamento/Refinamento de Requisitos | • Coleta de pontos de vista | | | |
| *ICARO* | Suporte à discussões de projeto | | •Conferências | •Hipertexto de documentos de software<br>•Registro de argumentação | • Delegação de Responsabilidades |
| *Quorum* | Suporte à discussões de projeto | | •Conferências | •Registro de argumentação | •Delegação de responsabilidades |
| *CSI* | Suporte à tarefa de inspeção de software | | •Conferências | •Registro dos resultados da inspeção | •Delegação de responsabilidades |
| *OZWeb* | Suporte global às atividades do processo | | | •Hipertexto de componentes de software | •Modelagem e encenação de processos<br>•Modelagem de usuários<br>•Controle de processos descentralizado (inter-grupos) |
| *CSD* | Suporte global às atividades do processo | | • Conferências | | |
| *EPOS* | Suporte global às atividades do processo | | •Conferências | •Gerência de Configuração | •Modelagem de usuários<br>•Modelagem e encenação de processos |
| *Conversation Builder* | Suporte global às atividades do processo | | | •Hipertexto de componentes de software | •Protocolos de conversação |
| *CPCE* | Suporte global às atividades do processo | | | •Registro de discussões | •Modelagem de processos |
| *PROSOFT* | Suporte global às atividades do processo | | | •Gerência de Configuração | •Controle de concorrência |
| *SPADE* | Suporte global às atividades do processo | | | | •Modelagem e encenação de processos |
| *Coo* | Suporte global às atividades do processo | | | | •Modelagem e encenação de processos |

Tabela 1 - Resumo das características de suporte dos ambientes/ferramentas de apoio ao projeto cooperativo de software

## 9. REFERÊNCIAS


[ANDR95] - Andres, H.P. (1995) Coordination of software development workgroups Proceedings of the First Americas Conference on Information Systems Pittsburgh Pennsylvania USA Agosto págs. 1-3.

[ARAU94] - Araujo, R.M. (1994) Quorum - um SSDG para o desenvolvimento de software Tese de Mestrado COPPE/UFRJ - Programa de Engenharia de Sistemas e Computação Agosto.

[ARAU96] - Araujo, R.M. Xexéo, G.B. (1996) Projeto participativo de software - uma visão sob a perspectiva da ecologia cognitiva Exame de Qualificação ao Doutorado COPPE/UFRJ Programa de Engenharia de Sistemas e Computação.

[ARME92] - Armenise, P. Bandinelli, S. Ghezzi, C. Morzenti, A. (1992) Software processes representation languages: survey and assessment 4th International Conference on Software Engineering and Knowledge Engineering Capri Itália Junho págs. 455-462.

[BAND93] - Bandinelli, S. Braga, M. Fuggetta, A. Lavazza, L. (1993) Cooperation in the SPADE environment: a case study Proceedings of Workshop on Computer Supported Cooperative Work, Petri Nets and Related Formalisms Chicago Junho.

[BEN-94] - Ben-Shaul, I.Z. Kaiser, G.E. (1994) A paradigm for decentralized process modeling and its realization in the Oz environment 16th International Conference on Software Engineering Sorrento Italy Maio.

[BISC94] - Bischofberger, W.R. Kofler, T. Mätzel, K. Schäffer, B. Computer-supported cooperative software engineering with beyond-sniff UBILAB Technical Report 94.9.1 Union Bank of Switzerland Zurich.

[CAVA94] - Cavalcanti, M.C.R. Borges, M.R.S. (1994) ARCoPAS: um ambiente para recuperação cooperativa do projeto arquitetônico de sistemas VIII Simpósio Brasileiro de Engenharia de Software Curitiba Outubro.

[CAVA96] - Cavalcanti, M.C.R. Borges, M.R.S. (1996) Participatory software design - specifying a methodology and a support environment unpublished paper Object Technology Laboratory School of Engineering Santa Clara University Fevereiro.

[DEWA93] - Dewan, P. Riedl, J. (1993) Toward computer-supported concurrent software engineering Computer January págs. 17-27.

[GIBB89] - Gibbs, S. (1989) CSCW and software engineering Object-Oriented Development Tsichritzs, D. (ed) Centre Universitaire d'Informatique Université de Genève Julho págs. 31-40.

[IOCH95] - Iochpe, C. (1995) Improving requirements analysis through cscw Primeiro Seminário Internacional CYTED-RITOS sobre Sistemas Cooperativos Lisboa Portugal Setembro pp.29-37.

[KAPL92] - Kaplan, S. M. Tolone, W. J. Carroll, A. M. Bogia, D. P. Bignoli, C. (1992) Supporting collaborative software development with conversation builder V Symposium on Software Development Environments.

[KEIL95] - Keil, M. Carmel, E. (1995) Customer-developer links in software development CACM vol. 38 n. 5 págs.33-44.

[LICE96] - Licea, G. Favela, J. (1996) ICARO: a web-based environment for collaborative software development Segundo Seminário Internacional CYTED-RITOS sobre Sistemas Cooperativos Setembro Puerto Varas Chile págs.23-30.

[MAGN96] - Magnusson, B. Guerraoui, R. (1995) Support for collaborative object-oriented development International Symposium on Parallel and Distributed Computing Systems (PDCS'96), Dijon, Setembro.

[MASH93] - Mashayekhi, V. Drake, J. M. Tsai, W. Riedl, J. (1993) Distributed, collaborative software inspection IEEE Software September págs. 66-75.

[PATN95] - Patnayakuni, R. Patnayakuni, N. (1995) A socio-technical approach to CASE and the software development process Proceedings of the 1st Americas Conference on Information Systems Pittsburgh Pennsylvania USA Agosto págs. 6-8.

[SOMM93] - Sommerville, I. Rodden, T. (1993) Environments for cooperative systems development, Proceedings of the Software Engineering Environments Conference, Reading, UK Julho págs. 144-155.

[SWEN93] - Swenson, K. D. (1993) Regatta project: a tool for business process reengineering Proceedings of the First International Conference in Technologies and Theories for Human Cooperation, Collaboration and Coordination - Applica'93 Março.

[TROT92] - Trotta, C. N. F. (1992) Software development environments and computer supported cooperative work Exame de Qualificação ao Doutorado COPPE/UFRJ - Programa de Engenharia de Sistemas e Computação.

## 10. WEBLIOGRAFIA

[CANA94] - Canals, G. Charoy, F. Godart, C. Molli, P. (1994) P-Root & COO: building a cooperative software development environment http://gille.loria.fr:7000/see95/see95.html atualização em 31/08/94 acesso em 12/12/96.

[CONK96] - Conklin, J. (1996) Designing organizational memory: preserving intellectual assets in a knowledge economy Corporate Memory Systems, Inc http://www.zilker.net/business/info/pubs/desom/ Setembro.

[CoTech96] - (1996) CoTech Project P4 - CSCW & Software Engineering http://cscw.risoe.dk/www/CoTech/Projects.html acesso em 05/03/97 última atualização em 23/11/96.

[CPCE94] - (1994) A collaborative process-centered environment kernel http://www.loria.fr/~jloncham/caise.ps.gz acesso em 13/12/96 publicado em CAISE'94.

[CSD96] - (1996) CSD Home Page http://www.igd.fhg.de/~marcos/csd.html acesso em 16/1196.

[DNP96] - (1996) The Designers' NotePad http://www.comp.lancs.ac.uk/computing/users/mbt/dnp/dnp.html acesso em 22/12/96

[EPOS96] - (1996) EPOS group home page http://www.idt.unit.no/~epos/ acesso em 10/12/96.

[GODA94] - Godart, C. Canals, G. Charoy, F. Molli, P. (1994) An introduction to cooperative software development in COO International Conference on Systems Integration ICIS'94 também disponível em http://gille.loria.fr:7000/icsi94/ICSI94/ICSI94.html acesso em 13/12/96.

[GODA96] - Godart, C. Canals, G. Charoy, F. Molli, P. Skaf, H. (1996) Designing and implementing COO: design process, architectural style, lessons learned http://gille.loria.fr:7000/icse96.ps.gz acesso em 13/12/1996.

[HILD96] - Hildebrand, C. (1996) Loud and clear (interview with Erwin Martinez) CIO Magazine http://www.cio.com/CIO/041596_qa.html acesso em 20/12/96.

[MARA96] - Maranhão, A. S. Iochpe, C. (1996) Proposta de identificador de palavras-chave para ferramenta de suporte à resolução de conflitos de linguagem I Semana Acadêmica do CPGCC/UFRGS - Instituto de Informática Setembro http://www.inf.ufrgs.br/~deamar/seminar.htm acesso em 24/12/96.

[OSTW95] - Ostwald, J. (1995) Supporting collaborative design with representations for mutual understanding CHI'95 Proceedings Doctoral Consortium disponível em http://www.acm.org/sigchi/chi95/Electronic/documnts/doctoral/jod_bdy.html acesso em setembro/1996.

[OZWE96] - (1996) OZWeb project summary http://www.psl.cs.columbia.edu/edcs/ acesso em 19/11/96.

[REIS96] - Reis, R. Q. Nunes, D. J. (1996) Uma proposta de suporte a cooperação no ambiente PROSOFT I Semana Acadêmica do CPGCC/UFRGS - Instituto de Informática Setembro http://www.inf.ufrgs.br/~quites/artigos/cprosoft.html acesso em 24/12/96.

[TWID93] - Twidale, M. Rodden, T. Sommerville, I. (1993) The designers' notepad: supporting and understanding cooperative design http://www.comp.lancs.ac.uk/computing/users/mbt/dnp/dnp_ecscw93.ps acesso em 22/12/96.

[VITA96] - Vitali, F. Durand, D.G. (1996) Using versioning to support collaboration on the WWW http://cs-pub.bu.edu/students/grads/dgd/version.html acesso em 06/01/97.

[VORD96] - (1996) VORD Home Page http://www.comp.lancs.ac.uk/computing/users/rcm/vord.html acesso em 12/12/96.