Apêndice A
Componentes e Frameworks

Para atingir a qualidade e cumprir o orçamento e os prazos previstos, não é mais possível começar a construir software a partir do zero e estruturá-lo na forma de blocos monolíticos onde uma modificação propaga efeitos colaterais por todo o código, dificultando a manutenção e a substituição de suas partes [Werner & Braga, 2005, p.66]. O Desenvolvimento Baseado em Componentes (DBC) surgiu como uma abordagem para o desenvolvimento de software, cujo objetivo é a quebra dos blocos monolíticos em componentes interoperáveis, instrumentando o desenvolvimento e reduzindo seus custos, por meio do reuso de componentes [Sametinger, 1997]. O software passa a ser composto de partes relativamente independentes, que foram concebidas para serem substituíveis, reusáveis e interoperáveis em uma infra-estrutura de execução específica.

A.1.
Componentes de Software

Componentes têm um papel de destaque em outras engenharias, uma vez que permite a adoção do conceito de “caixa-preta”, que possibilita ao desenvolvedor um maior nível de abstração e independência. Na indústria automobilística, é possível utilizar o mesmo pneu, parafuso, gasolina e motor para diferentes marcas e modelos de automóveis. Não é necessário, e seria demasiadamente caro, reprojetar e construir sob demanda cada peça para cada carro.

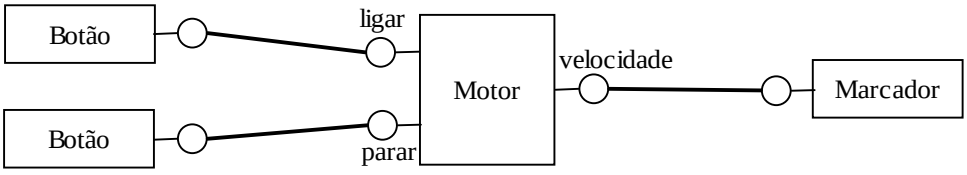

Figura A.1. Exemplo de uso de componentes [D’Souza & Wills, 1998, p.405]

O exemplo da Figura A.1, tratado por D'Souza & Wills [1998], ilustra a componentização na ligação de um motor a dois botões e a um marcador. Os mesmos botões e o marcador são reusáveis em outras situações (por exemplo, utilizando o marcador para marcar a temperatura) e estes componentes são substituíveis por outros equivalentes, porém com implementações diferentes (por exemplo, substituindo os dois botões por uma chave liga-desliga ou o marcador analógico por um digital). A interoperabilidade é propiciada pela compatibilidade entre os conectores. O conjunto todo pode ser encarado como um componente de um sistema maior.

A componentização de software visa trazer estes princípios para o desenvolvimento: reuso, substituição e montagem [D'Souza & Wills, 1998]. O desenvolvedor passa a desenvolver pedaços de software encapsulados na forma de componentes para que outros desenvolvedores possam utilizá-los, substituí-los ou modificá-los, com efeitos colaterais reduzidos.

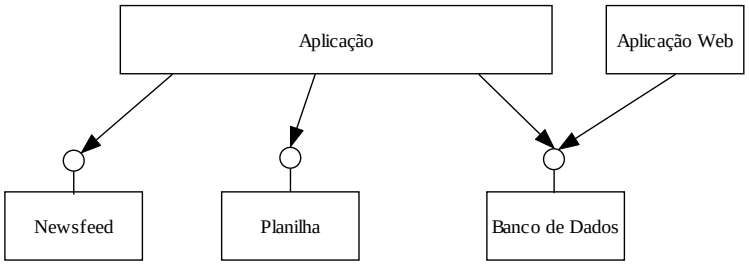

Figura A.2. Exemplo de uso de componentes de software [D'Souza & Wills, 1998, p.384]

D'Souza & Wills [1998] exemplificam a composição de software através da situação ilustrada na Figura A.2. Uma aplicação lê dados de uma fonte sobre ações, registra em uma planilha e passa os resultados a um banco de dados. O banco de dados é compartilhado por uma aplicação web, que extrai informações sob demanda. Cada componente é relativamente independente, compartilhado e utilizado para compor um sistema maior. Os componentes disponibilizam interfaces para interconexão e são substituíveis por outros equivalentes. Eventualmente, um componente não precisa de interação com o usuário ou de mecanismos de persistência.

A.2. Definição de Componente de Software

Nesta seção são apresentadas algumas definições para componente de software encontradas na literatura. O termo componente também é comparado com outros termos e conceitos utilizados no desenvolvimento de software. Para alguns autores um componente de software é qualquer elemento reusável: código binário, código fonte, estruturas de projeto, especificações e documentações [Krueger, 1992]. Outros autores focam na tecnologia, e consideram como componente qualquer pedaço de código que segue uma especificação. A Tabela A.1 apresenta algumas definições encontradas na literatura, ordenadas cronologicamente.

Booch [1987]:	Módulo logicamente coerente e fracamente acoplado que denota uma abstração única.
Component Int. Labs [Orfali et al., 1995]:	Pedaço de software pequeno o suficiente para implementar e manter, grande o suficiente para distribuir e dar suporte, e com interfaces padronizadas para oferecer interoperabilidade.
Brown [1996]:	Parte não-trivial de um sistema, praticamente independente e substituível, com uma função clara no contexto de uma arquitetura bem definida.
Sametinger [1997]:	Pedaço de software autocontido, claramente identificável, que descreve ou executa funções específicas, tem interfaces claras, documentação apropriada e um status de reuso.
Szyperski [1997, p.34]:	Unidade binária com interfaces contratualmente especificadas e dependências de contexto explícitas, instalável de forma independente e usado por terceiros sem modificação para compor aplicações finais.
D'Souza & Wills [1998, p.387]	Um pacote coerente de software que (a) pode ser desenvolvido e instalado independentemente como uma unidade, (b) tem interfaces explícitas e bem definidas para os serviços que provê, (c) tem interfaces explícitas e bem definidas para os serviços que espera de outros, e (d) pode ser utilizado para composição com outros componentes, sem alterações em sua implementação, podendo eventualmente ser customizado em algumas de suas propriedades.
Councill & Heineman [2001, p.7]	Um *componente de software* é um elemento que está em conformidade com um modelo de componentes e pode ser instalado independentemente e composto sem modificações.
Barroca et al. [2005]:	Unidade de software independente, que encapsula, dentro de si, seu projeto e implementação, e oferece serviços, por meio de interfaces bem definidas, para o meio externo.
OMG [2005]:	Um componente representa uma parte modular de um sistema que encapsula seu conteúdo e cuja manifestação é substituível. Um componente define seu comportamento através de interfaces fornecidas e requeridas.

Tabela A.1. Definições de componente de software

Neste trabalho, é adotada a definição proposta por D'Souza & Wills [1998], que enfatiza a possibilidade de customização e a necessidade de explicitar as interfaces fornecidas e requeridas. Também são consideradas as definições de

Szyperski [1997], que diz que um componente é uma unidade executável[32], e a de Councill & Heineman [2001, p.7], que diz que o componente segue um modelo de componentes. Apesar destas duas características não estarem explícitas na definição proposta por D'Souza & Wills [1998], os autores as adotam em seu livro. Das outras definições apresentadas na tabela, algumas são razoavelmente equivalentes à definição adotada [Sametinger, 1997; Orfali et al., 1995; Barroca et al., 2005], e outras são mais genéricas [Booch, 1987; Brown & Wallnau, 1996; OMG, 2005].

Componentes de software são utilizados nas mais diversas áreas e com os mais diversos propósitos. Um exemplo bastante conhecido é o uso de plugins nos navegadores web, utilizados para estender o suporte à visualização de conteúdos, como animações flash, documentos Word e vídeos diversos. Os plugins se comportam como componentes, pois são autocontidos, reusáveis, substituíveis, provêem serviços específicos de uma maneira coesa e bem definida, seguem uma padronização e são disponibilizados como uma unidade executável em uma plataforma pré-definida. Várias aplicações utilizam o suporte a plugins para estender as funcionalidades nativas, como o Adobe Photoshop, o Eclipse IDE e o Apache Web Server.

Componentes visuais de interface com o usuário (widgets) é outro exemplo bastante difundido do uso de componentes. Alguns ambientes de desenvolvimento integrado (IDEs) disponibilizam ao desenvolvedor um conjunto de componentes visuais que são utilizados para compor a interface com o usuário. O programador instancia e posiciona os componentes, configura suas propriedades e associa métodos a seus eventos, sem ter acesso a seu código fonte.

Dada a elasticidade e o desgaste do termo componente, é apropriado diferenciá-lo de outros conceitos e tecnologias para melhor caracterizá-lo. A seguir, o conceito de componente de software é comparado com os conceitos de módulo, objeto, classe, biblioteca e API.

[32] Clemens Szyperski, na segunda edição de seu livro "Component Software – Beyond Object-Oriented Programming", lançada em 2002, passou a caracterizar um componente como executável, ao invés de binário, visto que a capacidade de execução em uma plataforma é mais relevante do que a forma de empacotamento, afirmando que eventualmente um componente é implementado na forma de um script de código.

O conceito de componente é similar ao conceito tradicional de módulo, presente há bastante tempo na Engenharia de Software. A modularidade é de fato um pré-requisito para a tecnologia de componentes [Szyperski, 1997, p.33], que é diferenciada pela existência de uma padronização (*component model*) e de uma infra-estrutura específica para seu gerenciamento e execução (*component framework* e *container*) [D'Souza & Wills, 1998, p.386].

Também há uma certa confusão entre componentes e objetos, principalmente porque a maioria dos componentes é implementada utilizando linguagens orientadas a objetos [D'Souza & Wills, 1998, p.390]. O componente cria, envia e recebe objetos, e muitas vezes é representado e acessado por meio deles.

Componente e classe também são conceitos relacionados. Ambos utilizam polimorfismo e ligação dinâmica, realizam interfaces, podem participar de um relacionamento de dependência e composição, admitem instâncias e são participantes de interações [Booch et al., 2000]. Entretanto, classe e componente estão em níveis de abstração diferentes. Classe é uma abstração lógica do domínio (classe conceitual) ou uma estrutura de uma linguagem de programação utilizada para instanciar objetos (classe de software). Um componente é uma unidade executável, que pode ser a implementação "física" de uma ou mais classes[33]. Uma classe só pode pertencer a um único componente [Szyperski, 1997, p.32].

Um componente não necessariamente precisa estar na mesma máquina que a aplicação que o utiliza e nem escrito na mesma linguagem de programação. Diferentemente de classes, um componente pode estar disponível somente na forma binária e o nome do componente não pode ser utilizado para definir o tipo de uma variável ou parâmetro [Weinreich & Sametinger, 2001, p.36]. As padronizações para componentes também são mais abrangentes do que as padronizações para classes, definindo, entre outros fatores, empacotamento, ciclo de vida, conectores, interfaces providas e requeridas, etc. [D'Souza & Wills, 1998, p.390]. Componentes também têm uma gama maior de mecanismos de intercomunicação, como eventos e workflow, além das mensagens da orientação a

[33] Um componente não obrigatoriamente precisa conter classes [Szyperski, 1997, p.31]. Um componente pode conter código escrito em outras tecnologias.

objetos. Por fim, instâncias de componentes tendem a ser mais estáticas do que instâncias de classes, não tendo sua configuração alterada ao longo do ciclo de vida [D'Souza & Wills, 1998, p.391]. Mesmo estando em níveis de abstração diferentes, por simplificação do discurso, costuma-se chamar de componente a(s) classe(s) utilizadas na construção do componente.

Biblioteca de funções é outro termo similar a componente. Uma biblioteca provê serviços coesos para um sistema e é autocontida, reusável e substituível. Uma biblioteca pode ser disponibilizada na forma binária e orientada a objetos. Entretanto, uma biblioteca normalmente não pressupõe uma padronização, um modelo específico, um ciclo de vida, instalação e configuração (*deploy*), e nem a existência de uma plataforma específica de execução.

Um componente, assim como uma biblioteca, provê uma API (*Application Program Interface*), que representa parte do "contrato" de utilização daquele pedaço de código. Uma API expõe o conjunto de operações, procedimentos, funções, tipos e constantes que um pedaço de código oferece para ser utilizado externamente. Respeitando a API e os requisitos não-funcionais, é possível trocar o componente ou a infra-estrutura, sem impactar o código cliente. Por exemplo, o sistema operacional Windows oferece uma API padrão que é compatível entre suas diversas versões, de modo que um software desenvolvido para uma versão anterior do sistema operacional normalmente continua a funcionar em uma versão mais recente. O ambiente de trabalho é composto instalando e configurando os componentes necessários.

Muitas vezes, o termo componente é usado com pouco rigor em diversos níveis de abstração, o que pode ser a origem da confusão sobre o termo [D'Souza & Wills, 1998, p.388; Apperly, 2001, p.29]. Costuma-se chamar de componente: a especificação do componente, a implementação (código fonte), o executável que implementa a especificação (*deployable component*), a instalação em particular daquele executável, a execução específica daquele executável e a instância do componente. Com relação à granularidade, um componente de software pode ser construído a partir de uma única classe ou ser tão complexo como um subsistema [Gimenes & Huzita, 2005]. As definições apresentadas na Tabela A.1, ajudam a dar indícios sobre os critérios para considerar um pedaço de código como sendo um componente: ele deve ser passível de implementar e manter

independentemente, coeso, não-trivial e com uma função clara no contexto da arquitetura, bem como passível de separação, distribuição e reuso.

Resumindo, no contexto do DBC, componente de software é um elemento arquitetural mapeado em um arquivo executável, que segue uma especificação, e foi concebido para ser autocontido, reusável, substituível, além de prover serviços específicos de uma maneira coesa e bem definida. Portanto, classificar algo como componente depende também de como o código foi concebido e construído e como ele se relaciona com o restante do sistema, e não somente da aderência a uma padronização. Vale ressaltar que na literatura costuma-se encontrar alguns sinônimos para componentes, utilizados em contextos específicos, como plugin, add-on, módulo, serviço e widget.

A.3. Utilização de Componentes

Nesta seção são definidos alguns termos referentes à utilização de componentes (porta, conector, interface, adaptador, instância e *deployment*). Também são discutidas as maneiras de customizar componentes.

Uma **porta** é um meio identificável de conexão, por onde um componente oferece seus serviços ou acessa os serviços dos outros [OMG, 2005]. Cada porta tem uma identificação (nome ou número pelo qual é acessada). Cada porta define os tipos de valores que são transmitidos ou recebidos e agrupa interfaces [D'Souza & Wills, 1998, p.415].

Ao meio por onde é feita a conexão entre duas ou mais portas é dado o nome de **conector** [D'Souza & Wills, 1998, p.414; OMG, 2005]. O conector é implementado por invocação explícita de função, envio de mensagens síncronas ou assíncronas, propagação de eventos, stream de dados, workflow, código móvel, diálogo através de uma API, transferência de arquivo, pipe, propagação de eventos, buffer, sinalização, compartilhamento de arquivo via ftp, etc. [D'Souza & Wills, 1998, p.389, p.395; Wills, 2001, p.312]. Os tipos de conectores variam para cada tecnologia e ferramenta adotada. Por exemplo, JavaBeans oferece um conjunto de conectores diferentes dos oferecidos pelo COM+ [D'Souza & Wills, 1998, p.414]. Tipicamente, os componentes interagem entre si através de

chamadas de métodos, em um estilo cliente/servidor ou publicador/ouvinte [Weinreich & Sametinger, 2001, p.43]. A conexão entre os componentes é feita em tempo de codificação, compilação, inicialização ou execução. O conector além do fluxo de informações define também o fluxo de controle [Wills, 2001, p.313].

A **interface** de um componente define seus pontos de acesso, por meio dos quais outros componentes utilizam os serviços oferecidos [Szyperski, 1997, p.40]. A interface representa o contrato de utilização do componente. Respeitando os contratos, pode-se alterar a implementação interna do componente ou substituí-lo[34] por outro, sem modificar quem o utiliza. O contrato cobre aspectos funcionais e não funcionais [Szyperski, 1997, p.370]. Aspectos funcionais incluem a sintaxe e a semântica da interface. Os não funcionais incluem os aspectos referentes à qualidade de serviço.

A interface de um componente de software funciona como a especificação dos pinos de um circuito integrado. Para usar o circuito, basta conhecer o funcionamento de sua interface externa e prever na arquitetura do circuito, o encaixe para ele. Não é necessário o conhecimento dos detalhes internos de funcionamento do componente (abordagem caixa preta). Ao estabelecer o contrato de utilização entre o componente e seus clientes e separar a especificação e a implementação do componente, os componentes são desenvolvidos e substituídos transparentemente para seus clientes [Szyperski, 1997, p.34].

A interface define os serviços que os clientes podem requisitar de um componente e as restrições que devem ser observadas pelos componentes e clientes [Councill & Heineman, 2001, p.8; Weinreich & Sametinger, 2001, p.39]. Um componente apresenta múltiplas interfaces correspondendo aos conjuntos de serviços que visam diferentes necessidades dos clientes, de modo a simplificar o uso e reduzir o acoplamento entre os componentes [D'Souza & Wills, 1998, p.397]. As interdependências entre os componentes ficam restritas às interfaces específicas em vez de abranger o componente como um todo [OMG, 2005]. A substituição é realizada levando em conta somente as interfaces utilizadas.

[34] Para um componente substituir outro, o substituto deve prover ao menos os mesmos serviços solicitados ao original e não deve solicitar serviços distintos dos que o anterior utilizava [D'Souza & Wills, 1998].

Normalmente, um componente possui pelo menos duas interfaces. Uma interface é relativa à funcionalidade que o componente oferece a outros componentes (interface de negócio) e outra à conexão com a infra-estrutura de execução (interface de sistema). Na conexão com a infra-estrutura de execução são tratados serviços técnicos, como os relacionados ao ciclo de vida, à instalação e à persistência. Apperly [2001, p.30] compara componentes com janelas utilizadas na construção civil. As janelas são entregues como um pacote fechado, são plugadas em uma infra-estrutura, com um determinado propósito, e suas funcionalidades são utilizadas por seus clientes. Há uma interface que define a utilização pelos clientes (interface de negócio) e uma outra interface que determina a junção da janela com a infra-estrutura, especificando quantos e quais buracos de fixação são necessários (interface de sistema). Tanto o cliente quanto a infra-estrutura são independentes dos detalhes internos de implementação (revestimento dos pregos, tipo de cola empregada, etc.).

As interfaces são fornecidas (*provided interfaces*) ou requeridas (*required interfaces*) [Councill & Heineman, 2001, p.9]. Um componente realiza uma determinada interface fornecida se contém uma implementação de todas as operações definidas por aquela interface. Um componente possui uma interface requerida se utiliza uma operação definida naquela interface. Componentes se conectam por meio da interface requerida de um com a interface fornecida de outro [Barroca et al., 2005, p.6], conforme ilustrado na Figura A.3. Se um componente for totalmente autocontido, ele não possui interface requerida. Um componente pode requerer mais de uma interface e estar em conformidade com mais de uma interface fornecida. As dependências de contexto são definidas pelas interfaces requeridas [Szyperski, 1997, p.369].

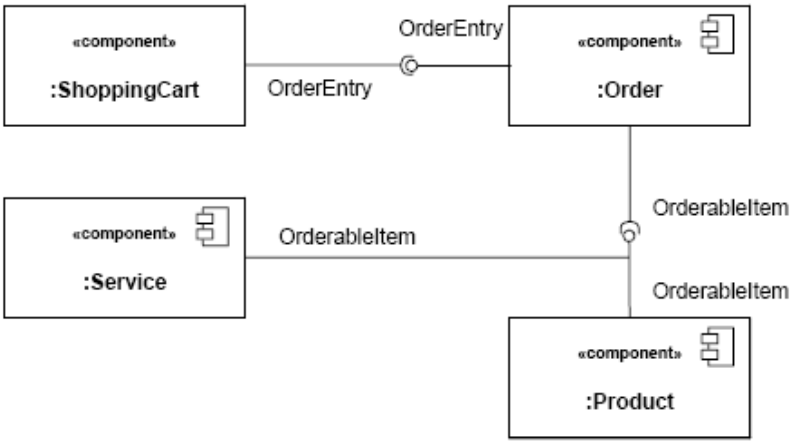

Figura A.3. Conexão entre os componentes [OMG, 2005]

A implementação de interface da orientação a objetos pode ser utilizada para implementar o conceito de interface de componente. Propriedade e eventos são mapeados na forma de métodos, porém, alguns aspectos da interface, como os aspectos não-funcionais e as interfaces requeridas, não são passíveis de representação [Gimenes & Huzita, 2005; D'Souza & Wills, 1998, p.388]. Alguns modelos de componentes definem maneiras próprias para representar a interface, como por exemplo, a IDL (*Interface Definition Language*) do CORBA [Siegel, 2000].

Um componente implementa diretamente uma interface ou implementa objetos que provêem interfaces [Szyperski, 1997, p.41]. Estes objetos são passados de componente em componente, de modo que um componente não tem ciência da origem do objeto e do código sendo executado [Szyperski, 1997, p.42]. Chama-se de **instância de componente**, o conjunto de objetos pelos quais se manipula o componente [Szyperski, 1997, p.370]. Estes objetos são a manifestação do componente em tempo de execução [D'Souza & Wills, 1998, p.390]. Código adicional, conhecido como **adaptador**, pode ser inserido entre componentes para realizar conversões simples de modo a compatibilizar interfaces nos casos em que a substituição de tipos não é suficiente.

Para utilizar um componente é necessário implantá-lo (**deployment**) em uma infra-estrutura de execução. A implantação não pressupõe a modificação do componente, entretanto, oferece a possibilidade de customização externa [Heineman, 2000; D'Souza & Wills, 1998, p.395]. A customização consiste na adaptação de um componente antes de sua instalação ou uso, normalmente com o

objetivo de especializar seu comportamento [Weinreich & Sametinger, 2001, p.42; D'Souza & Wills, 1998, p.xvii]. Como normalmente componentes são desenvolvidos no estilo *blackbox*, revelando o mínimo possível de sua implementação, as maneiras mais comuns de customizar um componente é através da modificação de propriedades ou da composição com outros componentes que especializam determinados comportamentos [Weinreich & Sametinger, 2001, p.42].

Na customização por composição, um componente contém uma referência a outro e repassa a ele as chamadas das operações. Um componente só depende da interface do outro, o que caracteriza um reuso *blackbox* [Szyperski, 1997, p.137]. No reuso *blackbox*, nenhum detalhe, além dos providos pelas interfaces do componente e por sua especificação, é disponibilizado aos clientes. Outra maneira de customizar o comportamento de um componente é modificando suas propriedades. Técnicas para isto incluem passagem de parâmetros para seus métodos, tabelas lidas pelo componente, configuração e opções na implantação. Uma abordagem comum é associar um arquivo descritor a um componente.

O arquivo descritor possibilita que o componente seja configurado sem que seja necessário recompilá-lo [Szyperski, 1997, p.33]. O arquivo descritor provê informações sobre o conteúdo do pacote, sobre as dependências externas e sobre as configurações do componente. Esta descrição é analisada pela infra-estrutura de execução e é utilizada para instalar e configurar o componente [Weinreich & Sametinger, 2001, p.44].

Componentes são *blackbox* ou *whitebox*, com as classificações intermediárias de *graybox* ou *glassbox* [Szyperski, 1997, p.29]. No reuso *whitebox*, detalhes do código interno são liberados, possibilitando, por exemplo, o uso da herança para customizar o comportamento do componente [Szyperski, 1997, p.33]. Apesar de não ser recomendada, a herança que ultrapassa as fronteiras do componente às vezes é utilizada [Szyperski, 1997, p.32]. Entretanto, esta herança cria uma dependência e acoplamento direto entre as implementações das classes, o que vai de encontro às características de componentes. D'Souza & Wills [1998, p.475] propõe uma maneira de transformar heranças em composições de objetos.

A.4. Representação de Componentes

A documentação favorece o reuso [Sametinger, 1997]. A documentação deve ser suficiente para recuperar um componente, avaliar sua adequabilidade ao contexto do reuso, fazer adaptações e integrá-lo ao seu novo ambiente [Gimenes & Huzita, 2005]. É necessária uma notação comum entre os envolvidos para documentar e debater sobre o projeto do sistema, reduzindo a chance de má interpretação. A UML (*Unified Modeling Language*) é a linguagem padrão para representar o projeto de sistemas orientados a objetos.

O conceito de componente adotado inicialmente na UML, até a sua versão 1.5, é excessivamente abrangente se comparado com a concepção da comunidade de desenvolvimento baseado em componentes [Houston & Norris, 2001, p.257]. De acordo com os autores originais da UML: “Os componentes são empregados para a modelagem de coisas físicas que podem residir em um nó, como executáveis, bibliotecas, tabelas, arquivos e documentos” [Booch, Rumbaugh & Jacobson, 2000, p.341]. A representação de componente na UML 1.x é apresentada na Figura A.4 à esquerda.

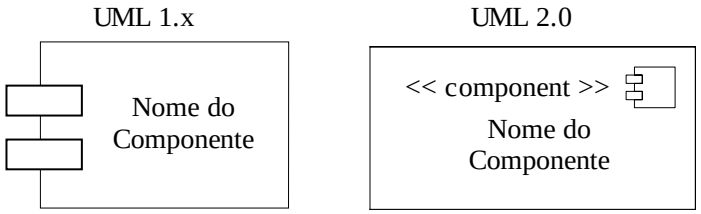

Figura A.4. Representação de componente na UML 1.x e 2.0

A versão 2.0 da UML [OMG, 2005], cuja versão final do documento de especificação de superestrutura foi lançada em agosto de 2005, altera profundamente a concepção de componente de software em relação à versão anterior. Nesta nova concepção, um componente é visto como uma unidade modular com interfaces bem definidas e substituível no contexto do sistema. A Figura A.4 à direita apresenta o novo símbolo de componente. A UML 2.0 prevê a representação de portas, conectores, interfaces providas e requeridas. A Figura A.5 à esquerda apresenta um componente com duas interfaces providas e três requeridas, representadas na forma icônica. A Figura A.5 à direita representa as interfaces na forma expandida e a ligação entre o componente e a interface

provida através de um relacionamento de realização e a ligação com uma interface requerida através de um relacionamento de dependência.

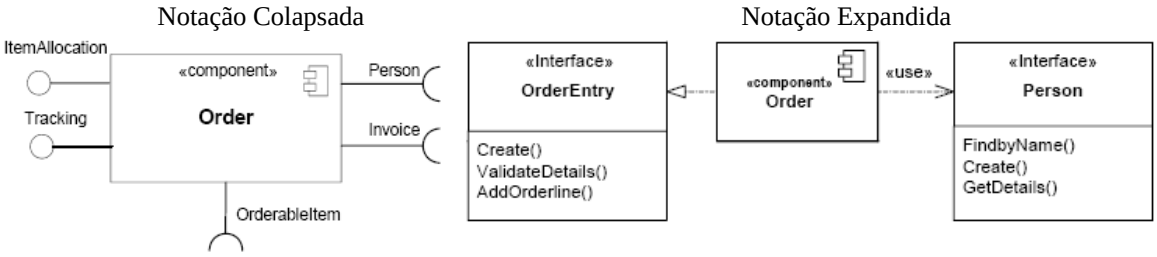

Figura A.5. Representação de interfaces na forma colapsada e expandida [OMG, 2005]

Para representar as classes que o componente implementa, é utilizado o relacionamento de dependência ou a estrutura de classes é aninhada dentro símbolo de componentes, conforme ilustrado na Figura A.6.

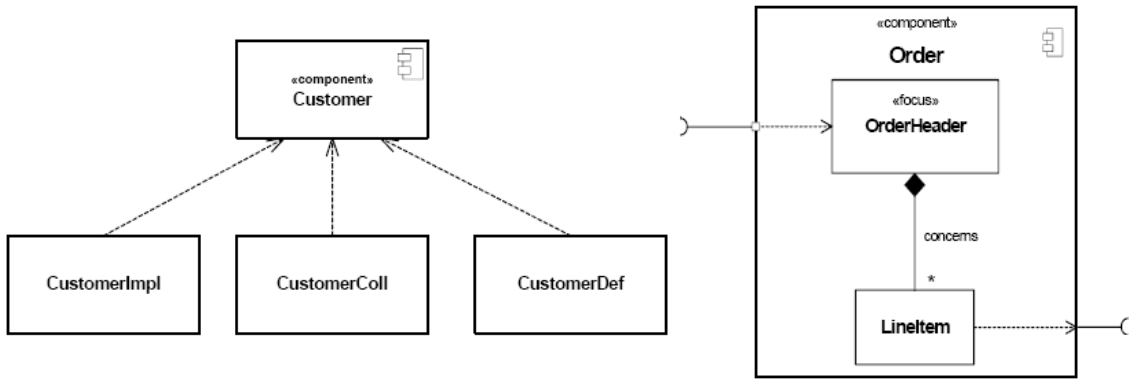

Figura A.6. Representação das classes que o componente implementa [OMG, 2005]

D'Souza & Wills [1998, p.84] propõe uma extensão à UML para representar a interface de um componente e seu modelo de objetos. Conforme ilustrado na Figura A.7, o compartimento do meio da representação da interface é utilizado para explicitar o modelo de objetos manipulado pelas operações da interface, de modo a facilitar o entendimento e o uso do componente.

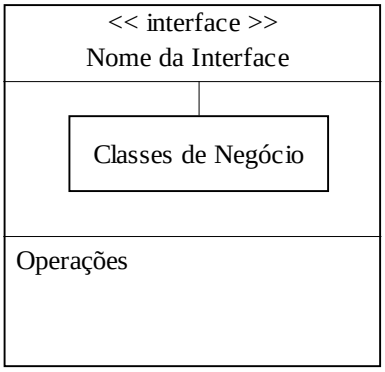

Figura A.7. Representação de um componente e seu modelo de objetos

Além da documentação gráfica da estrutura e dos inter-relacionamentos entre os componentes, é necessário descrevê-los textualmente. Nesta descrição é necessário englobar aspectos referentes aos requisitos funcionais e não-funcionais. Memória utilizada e tempo de resposta são exemplos de aspectos que precisam ser documentados, pois podem criar dependências não previstas, em um fenômeno chamado de vazamento de propriedade [Gimenes & Huzita, 2005]. Não há uma notação padrão para a descrição dos componentes. Paludo & Burnett [2005] propõem a utilização da estrutura de documentação de padrões de projeto para documentar componentes. São documentados, por exemplo, nome, intenção, aplicabilidade, variabilidade, estrutura, código de exemplo, usos conhecidos e componentes relacionados.

A.5. Implementação de Componentes

Há componentes de software implementados em linguagens procedurais, como Pascal e C, embora a maior parte dos componentes seja desenvolvida utilizando metodologias e linguagens orientadas a objetos [Vicenzi et al., 2005, p. 235; Apperly, 2001, p.29]. Para um componente o relevante é sua interface externa e não a maneira como foi implementado.

As primeiras APIs para componentes disponibilizavam apenas um conjunto de funções que componentes externos poderiam invocar, enxergando o componente com um bloco único. Atualmente, as APIs possibilitam acessar os objetos que compõe o modelo do componente [D'Souza & Wills, 1998, p.394]. Desta maneira, as interfaces que o componente disponibiliza ou requer definem o **modelo de objetos** que é compatível com aquele componente. Referências a objetos criados em um componente deixam as fronteiras e se tornam visíveis aos clientes do componente [Szyperski, 1997, p.31]. Por exemplo, é possível acessar o objeto célula do componente Planilha Eletrônica, o objeto ponto do Gerenciador Gráfico, entre outros. [D'Souza & Wills, 1998, p.394].

Outro fator importante a ser considerado na construção de componentes é o empacotamento, que possibilita que o componente seja instalado como uma unidade [Szyperski, 1997, p.276]. O empacotamento pode conter arquivos,

módulos, código executável, código fonte, código de validação, especificações, testes, documentações, etc. [D'Souza & Wills, 1998, p.386]. Os mecanismos para o empacotamento diferem de tecnologia para tecnologia [D'Souza & Wills, 1998, p.387]. Por exemplo, componentes Java são empacotados em arquivos JAR, que incluem as classes que implementam os serviços dos componentes. [D'Souza & Wills, 1998, p.400]. Os diversos fatores que definem as possibilidades de implementação de um componente são definidos no *component model* correspondente.

A.5.1. Modelo de Componentes

Ao comprar um eletrodoméstico, ele é conectável à tomada de uma casa porque a tomada, o plug e a energia disponíveis são aderentes a uma padronização. Para obter o reuso e substitutibilidade de componentes de software, eles também devem ser aderentes a um padrão, que na literatura é chamado de *modelo de componentes* (*component model*[35]). Assim como há mais de um padrão para tomada, há mais de um modelo para componentes de software. Um programador pode criar seu próprio modelo, eventualmente sendo uma especialização de um modelo disponível. Para compatibilizar componentes de um modelo para outro, adaptadores podem ser desenvolvidos.

Um modelo de componentes define vários aspectos da construção e da interação dos componentes, abordando, entre outros fatores: como especificar o componente, como instanciar o componente, quais os tipos de conectores e portas disponíveis, qual o modelo de dados padrão, como as ligações entre os objetos pertencentes a componentes diferentes são realizadas, como são feitas transações distribuídas, quais são os tipos de interface, as interfaces obrigatórias, como são tratados o catálogo e a localização dos componentes, o despacho das requisições e respostas, a segurança, o repositório, o formato, a nomeação, os meta-dados, a interoperabilidade, a documentação, os mecanismos de customização, a composição, a evolução, o controle de versões, a instalação e a desinstalação, os

[35] O termo *component model* é equivalente ao termo *component architecture* definido por D'Souza & Wills [1998].

serviços de execução, o empacotamento, etc. [Councill & Heineman, 2001, p.7, p.11; Weinreich & Sametinger, 2001, p.37; Szyperski, 1997, p.32; D'Souza & Wills, 1998, p.396, p.411].

O modelo de componentes define o padrão de descrição de interface [Weinreich & Sametinger, 2001, p.39]. Alguns modelos de componentes utilizam uma *interface description language* (IDL) independente da linguagem de programação, enquanto outros utilizam conceitos da própria linguagem ou da tecnologia para definir a interface, como é o caso das interfaces OO. Dependendo da IDL, interfaces podem incluir as exceções que podem ser lançadas, pré-condições e pós-condições para cada operação [Weinreich & Sametinger, 2001, p.39].

Um componente deve ser unicamente nomeado [Weinreich & Sametinger, 2001, p.40]. Alguns modelos de componentes definem a existências de identificadores únicos, como é o caso do COM Component Model. Outros modelos utilizam um espaço de nomes hierárquico, como é o caso das tecnologias da Sun Microsystems, que utilizam o nome do domínio invertido.

Um modelo de componentes deve definir quais são os padrões de composição. Os tipos de acoplamento mais comuns entre dois componentes são cliente/servidor e publicador/ouvinte [Weinreich & Sametinger, 2001, p.43]. No primeiro, o cliente explicitamente chama operações do servidor e, no segundo, o ouvinte se registra como tratador de eventos e informações disponibilizadas pelo publicador. O modelo de componentes define também os tipos de conectores disponíveis, que podem ser herdados da linguagem de programação correspondente ou serem próprios da tecnologia, como no caso do SOAP e IIOP.

Eventualmente, versões antigas e atuais de um componente co-existem no mesmo sistema. As regras e padrões para a evolução dos componentes e seu versionamento também fazem parte do modelo de componentes [Weinreich & Sametinger, 2001, p.44]. Um modelo de componentes também descreve como os componentes são empacotados e de que forma que eles podem ser individualmente instalados no sistema.

O modelo de componentes também define o padrão de *deployment*, que especifica a estrutura e a semântica para os arquivos descritores. O modelo de

componentes define a maneira como é feito o deployment, incluindo o eventual registro do componente e da interface [Weinreich & Sametinger, 2001, p.45]. Tipicamente o *deployment* envolve três passos: instalação do componente; configuração do componente e do ambiente de execução; e instanciação do componente para uso [Councill & Heineman, 2001, p.9].

Os modelos de componentes genéricos não provêem um modelo de negócio específico para um domínio [Weinreich & Sametinger, 2001, p.37]. Um projeto define seu próprio modelo de componentes independentemente ou como uma especialização de um existente [D'Souza & Wills, 1998, p.411]. Por exemplo, a OMG (*Object Management Group*), prevê a utilização do CORBA como base para a construção de modelos de componentes especializados. CORBAservices define a padronização voltada para serviços gerais de sistemas distribuídos, enquanto CORBAfacilities define a padronização dos serviços horizontais (comuns a vários domínios) [Weinreich & Sametinger, 2001, p.46]. Além da padronização de serviços de infra-estrutura, alguns modelos de componentes definem também uma padronização vertical, estabelecendo o modelo de objetos que os componentes interoperantes compartilham e manipulam [D'Souza & Wills, 1998, p.396].

Corba Component Model (CCM), Microsoft OLE, (D)COM/COM+, Sun EJB, JavaBeans e Web Service são alguns exemplos de modelos de componentes. Alguns modelos, como CORBA e Web Services, possibilitam o reuso de componentes que não foram necessariamente desenvolvidos na mesma linguagem de programação [Gimenes & Huzita, 2005]. Na subseção seguinte, são apresentados alguns modelos de componentes a título de exemplo.

A.5.2. Exemplos de Modelos de Componentes

O **OLE** (*Object Linking and Embedding*) é um modelo de componentes proposto pela Microsoft cujo propósito inicial era integrar em um documento objetos gerados por diversas aplicações. Ao instalar no sistema operacional aplicações compatíveis com o modelo, elas tornam-se disponíveis para receber e oferecer conteúdos. O editor de documento passa a ser um container que carrega

conteúdos disponibilizados pelas demais aplicações, como gráficos, sons, vídeo, figuras, conforme ilustrado na Figura A.8. Ao instalar uma nova aplicação, não é necessário modificar as demais para que elas troquem conteúdos.

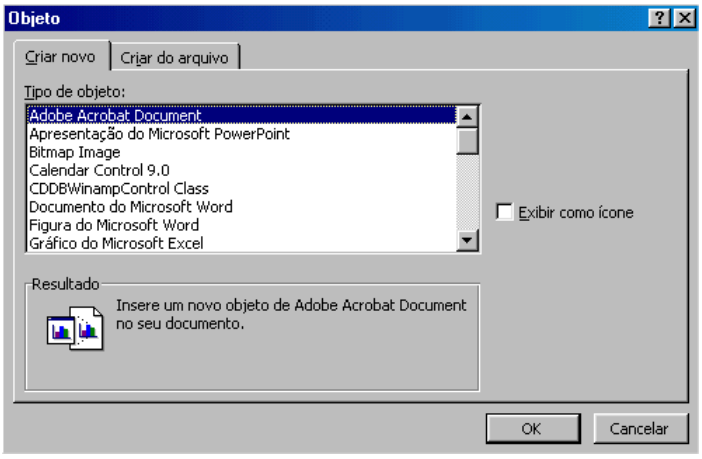

Figura A.8. Inserção de um objeto OLE em um documento do Microsoft Word

COM, **DCOM** e **ActiveX** são outros modelos de componentes utilizados pela Microsoft que vieram da evolução das idéias do modelo original OLE [Brockschmidt, 1996]. Os modelos oferecem suporte a diversos níveis de complexidade de componentes e possibilitam a integração de aplicações desenvolvidas por diferentes fabricantes. A solução enfoca componentes binários interoperáveis.

A Microsoft definiu um padrão para seus componentes de interface com o usuário (widgets), disponibilizados através da linguagem Visual Basic. O modelo introduzido foi o **VBX**, que posteriormente foi substituído pelo **OCX**, pelo ActiveX e depois pelo COM. Atualmente, o modelo de componentes do **.NET** unifica toda a tecnologia de componentes da Microsoft. A Borland desenvolveu seu próprio modelo de componentes de interface, o **CLX** (*Component Library for Cross Platform*), que possibilita o desenvolvimento de aplicações para o Microsoft Windows e para o Linux através dos ambientes Kylix, Delphi e C++ Builder.

O **CORBA** (*Common Object Request Broker Architecture*) fornece um modelo de componentes que possibilita a comunicação entre componentes distribuídos. O CORBA utiliza a IDL (*Interface Definition Language*) para descrever a interface pública de seus serviços. O cliente não é dependente da localização do objeto que está sendo invocado, da linguagem que o mesmo foi programado ou do sistema operacional que está sendo utilizado.

Web Services é um padrão para comunicação entre componentes através da web. A aplicação utiliza os serviços dos componentes através do protocolo SOAP, que encapsula as chamadas dos serviços e os retornos em pacotes XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
http://java.sun.com/xml/ns/portlet" xmlns="http://java.sun.com/xml/ns/portlet">
        <portlet>
              <portlet-name>helloWorld</portlet-name>
                <portlet-class>examples.helloworld.HelloWorld</portlet-class>
              <portlet-info>
                        <title>Hello World</title>
                </portlet-info>
        </portlet>
</portlet-app>
```

Figura A.9. Arquivo descritor de um portlet

JavaBeans é o modelo básico de componentes da Sun, a partir do qual são feitas diversas extensões. O JavaBeans define como tratar no componente eventos, propriedades, introspecção, reflexão, customização e empacotamento [Szyperski, 1997]. **Enterprise Java Beans** é o modelo para a interconexão de componentes remotos em aplicações corporativas. Recentemente a Sun lançou o modelo de componentes para interface com o usuário **Java Server Faces** (JSF) e o modelo **Portlets** (JSR 168) para o reuso de componentes em portais na web. Utilizando este modelo é possível reusar um componente desenvolvido para um portal em outro, de modo que uma instituição monta seu portal buscando ou adquirindo componentes de terceiros. Cada componente possui um arquivo descritor (Figura A.9) que define suas configurações e implementa uma interface que define os métodos ativados durante seu ciclo de vida (Figura A.10).

```
package examples.helloworld;

 import java.io.*;
 import javax.portlet.*;

 public class HelloWorld extends GenericPortlet
 {
 public void render(RenderRequest request, RenderResponse response)
 throws PortletException, IOException
     {
             response.getWriter().write("<p>Hello World</p>");
     }
 }
```

Figura A.10. Implementação de um método referente ao ciclo de vida de um portlet

Para executar estes componentes, é necessária uma infra-estrutura de execução específica, que cuida do ciclo de vida, agregação, segurança, configuração, persistência, etc. Por exemplo, no caso do Portlets, o container aciona os componentes que geram dinamicamente o conteúdo exibido para o usuário, agregando a saída de cada um em uma única apresentação.

A.6.
Infra-estrutura de Execução

Um container é uma plataforma, normalmente desenvolvida por terceiros, com o objetivo de hospedar e gerenciar componentes de um determinado modelo, provendo a eles serviços de infra-estrutura de execução. O acesso remoto, o gerenciamento de transações distribuídas, o pooling de recursos, o acesso concorrente, o clustering, a tolerância a falhas, a autenticação, a persistência, a configuração, o gerenciamento de permissões e de sessão são exemplos de serviços providos por containeres [D'Souza & Wills, 1998, p.401].

O container libera o desenvolvedor de implementar serviços técnicos de sistema de baixo nível, direcionando seus esforços para as regras de negócio e para a composição do sistema. Em alguns casos, o uso de container também possibilita alterar aspectos não relacionados com a lógica do negócio sem ter que alterar a aplicação, bastando re-configurar o container para mudar aspectos como segurança, permissões, logging, banco de dados, etc. A máquina virtual Java é um container para executar componentes Java (arquivos .class ou .jar). Algumas arquiteturas estendem o container para gerenciar componentes mais especializados como servlets ou applets [D'Souza & Wills, 1998, p.460]. Tomcat, JBoss, Webspheare, Pluto e JetSpeed são exemplos de containeres.

Component framework[36] é um conjunto de elementos de software, regras e contratos que governam a execução de componentes que estão em conformidade com um modelo de componentes [Szyperski, 1997, p.369]. O *component framework* define as invariantes e os protocolos de conexão entre os componentes, estabelece as "condições ambientais" para as instâncias dos componentes e regula as interações entre elas [Szyperski, 1997, p.276]. O *component framework* define uma interface chamada de *lifecycle interface* que estabelece a conexão com os componentes [Schwarz et al., 2003]. Esta interface é acessada para gerenciar,

[36] *Component framework* é um termo com diversos entendimentos na literatura. Neste trabalho, está sendo seguida a definição proposta por Szyperski [1997]. *Component framework* é equivalente ao conceito de *component model implementation* definido por Councill & Heineman [2001, p.7].

inicializar, executar e desativar os componentes. O *component framework* não acessa as interfaces de negócio dos componentes.

O *component framework* segue e oferece suporte a um modelo de componentes provendo uma infra-estrutura para apoiar sua execução. O *component framework* fornece uma base para a integração dinâmica dos componentes, até de diferentes fabricantes, de acordo com as necessidades e preferências dos usuários e desenvolvedores. O *component framework* oferece serviços de execução como criação de objetos, gerenciamento de ciclo de vida, persistência de objetos, licenciamento, acesso a dados, gerenciamento de transações, balanceamento de carga, etc. *Component frameworks* para sistemas distribuídos oferecem serviços adicionais, como modos de conexão e comunicação remota, notificação de eventos, localização de serviços e segurança [Weinreich & Sametinger, 2001, p.45]. Eventualmente, a interface ou o componente precisa ser registrado no *component framework* antes de ser utilizado [Councill & Heineman, 2001, p.12].

A relação entre um componente e um *component framework* é similar à relação entre um programa e o sistema operacional. Um sistema operacional provê um ambiente de execução para aplicativos, colocando uma camada sobre o hardware, e provendo serviços de baixo nível, como acesso a dispositivos de E/S, gerenciamento de memória, etc. [Weinreich & Sametinger, 2001, p.34]. O sistema operacional regula o acesso dos aplicativos aos recursos, oferece serviços de infra-estrutura, define os mecanismos de comunicação e interoperabilidade entre os componentes e oferece suporte à instalação e registro dos componentes (aplicativos). Os aplicativos são utilizados para montar o ambiente de trabalho e o sistema operacional define parcialmente a arquitetura do sistema [Szyperski, 1997, p.274].

O conceito de *component framework* está ligado à idéia de linha de produto [Barroca et al., 2005, p.20]. Uma linha de produtos de software é um conjunto de produtos que compartilham um conjunto de características visando satisfazer um determinado segmento de mercado ou uma missão específica [Clements & Northrop, 2001]. Para reduzir os custos de desenvolvimento e manutenção, os produtos são gerados de uma maneira sistemática a partir de um núcleo de artefatos.

A.7.
Component Kits

Um *component kit* é uma coleção de componentes que foram projetados para trabalhar em conjunto [D'Souza & Wills, 1998, p.404]. *Component kits* são freqüentemente utilizados para construir interfaces gráficas para aplicações a partir de botões, formulários, listas, etc. De um kit de componentes gera-se uma família de soluções, fazendo diferentes arranjos e eventualmente desenvolvendo alguns sob medida [Wills, 2001, p.309]. D'Souza & Wills [1998] ilustram a utilização de *component kits* na eletrônica. A partir de um conjunto de componentes interoperáveis e padronizados, são construídos circuitos para os mais variados propósitos.

A Figura A.11 ilustra a geração de um kit de componentes a partir de um conjunto de aplicações similares e a posterior construção de novas aplicações a partir deste kit. Inicialmente, o desenvolvedor analisa aplicações similares, identificando e generalizando componentes comuns, criando um kit de componentes. Tendo o kit, os componentes são utilizados para montar novas aplicações da mesma família [D'Souza & Wills, 1998, p.385].

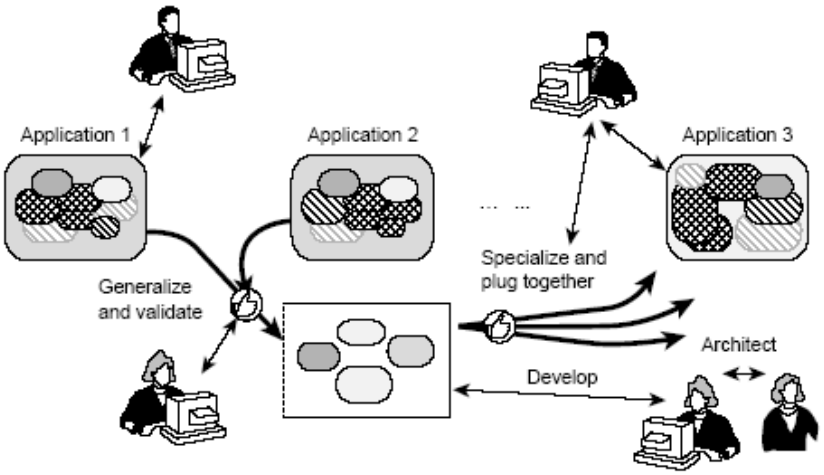

Figura A.11. Desenvolvimento de um kit de componente
e construção de aplicações a partir dele [D'Souza & Wills, 1998, p.385]

Um *component kit* é projetado para um modelo de componentes específico, o que favorece a interoperabilidade [D'Souza & Wills, 1998, p.428]. Um kit não necessariamente possui um conjunto fixo de componentes, podendo ser adicionado novos, respeitando a arquitetura definida. As diversas aplicações

compartilham os componentes, que por sua vez executam na infra-estrutura de execução.

Um toolkit é um tipo de SDK (*Software Development Kit*), que apresenta, além dos componentes, um conjunto de ferramentas para criar aplicações para uma determinada plataforma, sistema ou framework. Os toolkits são mais comumente encontrados para componentes de interface com o usuário, também chamados de widgets. Alguns exemplos de toolkits de widgets para a linguagem Java são o AWT (*Abstract Windowing Toolkit*), Swing e SWT (*Standard Widget Toolkit*).

Toolkits possibilitam que programadores desenvolvam software a partir de componentes prontos, mesmo sem muita experiência de programação, no estilo das ferramentas RAD (*Rapid Application Development*). Os toolkits favorecem a criatividade dos desenvolvedores, o que é fundamental para áreas não bem estabelecidas. Ao encapsular detalhes de implementação de baixo nível, os toolkits liberam os desenvolvedores para pensar em novas interfaces e mecanismos de interação, e possibilitam uma prototipação rápida para testar, refinar e validar as idéias [Greenberg, 2006].

A.8. Arquitetura de Software Baseada em Componentes

Uma arquitetura define a estrutura[37] de um software, descrevendo as propriedades, restrições e relacionamentos de suas partes [Stafford & Wolf, 2001, p.373]. Ela representa um conjunto de restrições e regras, definindo os elementos, conectores, protocolos, componentes, propriedades visíveis e a interação e a função de cada parte no contexto do sistema [D'Souza & Wills, 1998, p.481; Bass et al., 2003]. A arquitetura é uma representação abstrata de alto nível do projeto de um software. A granularidade dos componentes da arquitetura varia de pequenos pedaços de código a aplicações inteiras, como SGBDs ou servidores de e-mail. As

[37] Na literatura, muitas vezes o termo arquitetura também é utilizado para representar a infra-estrutura da aplicação, em frases do tipo "o componente interage com a arquitetura da aplicação".

conexões entre os componentes abstraem como eles de fato interagem, como por exemplo, chamada de método, compartilhamento de dados, pipes, RPCs (*Remote Procedure Calls*), sockets, etc. [D'Souza & Wills, 1998].

Um estilo arquitetural descreve uma família de arquiteturas de software que compartilham propriedades, como por exemplo, os componentes permitidos, as restrições e interações entre os componentes, as invariantes, o modelo computacional, etc. [Stafford & Wolf, 2001, p.383]. Fluxo de dados, máquina virtual, chamada de procedimento, MVC (*Model View Controller*), cliente-servidor, peer-to-peer, blackboard, camadas e orientação a serviços são exemplos de estilos arquiteturais [Barroca et al., 2005, p.17]. Cada estilo normalmente endereça um aspecto específico do sistema, sendo portanto possível de utilizar mais de um estilo na mesma arquitetura. Além disto, cada componente de um sistema pode ter uma arquitetura e estilo arquitetural próprios, desde que a parte externa do componente seja compatível com a arquitetura da aplicação.

Apesar da arquitetura ser única, pode-se fornecer várias visões sobre ela [D'Souza & Wills, 1998, p.483]. Cada visão enfoca um determinado aspecto da arquitetura, omitindo os demais [Stafford & Wolf, 2001, p.386]. Algumas visões são mais apropriadas para o desenvolvimento do sistema, outras para o reuso, outras para o teste e implantação.

O desenvolvimento baseado em componentes considera pelo menos duas visões da arquitetura: arquitetura de aplicação e arquitetura técnica [D'Souza & Wills, 1998, p.483]. Na arquitetura de aplicação, a preocupação é com a estrutura dos componentes do domínio, representando um projeto lógico de alto nível independente da tecnologia de suporte. Nesta arquitetura, são mostradas a função de cada componente no contexto do sistema e a interação entre eles. A arquitetura de aplicação consiste de um conjunto de decisões sobre a plataforma, um conjunto de *component frameworks* e os mecanismos de interoperação entre eles [Szyperski, 1997, p.275]. A arquitetura técnica considera os detalhes da tecnologia de componentes a ser utilizada e é totalmente independente do domínio da aplicação. A arquitetura técnica retrata as tecnologias de comunicação entre os componentes (TCP/IP, ODBC, etc.) e aspectos referentes a escalabilidade e performance.

A.9. Frameworks

O conceito de frameworks está relacionado ao de componentes; são conceitos complementares que contribuem para o reuso de software [Gimenes & Huzita, 2005]. Um framework é uma infra-estrutura reusável de todo ou de parte de um sistema, com o objetivo de ser instanciado para resolver uma família de problemas [Govoni, 1999]. As partes invariantes de um domínio são implementadas no framework e reusadas nas instanciações. O framework oferece uma estrutura comum para um domínio de aplicações, promovendo o reuso do conteúdo conceitual do domínio de um software ou da solução de um problema [Gimenes & Huzita, 2005].

Ao definir uma arquitetura parcialmente implementada e encapsular detalhes de implementação, o framework libera os desenvolvedores de terem que entender as complexidades envolvidas com a solução do problema e com o domínio utilizado [Pree, 1995]. O framework é normalmente construído por especialistas em um domínio particular e utilizado por leigos naquele domínio [Lajoie & Keller, 1995]. O reuso proporcionado pelo uso de um framework não atinge somente a implementação, se refletindo também na análise e no projeto do sistema.

Alguns frameworks são voltados à solução de problemas ligados à tecnologia, como interface com o usuário, persistência de objetos e suporte a MVC. Outros frameworks são voltados para um determinado domínio, como por exemplo, aplicações bancárias e relacionamento com o cliente. Estes frameworks são embasados em teorias e modelos do domínio e definem uma arquitetura orientada para a área de aplicação específica.

A utilização de frameworks traz as vantagens associadas ao reuso de código: aumento da qualidade, redução do esforço de implementação, direcionamento dos esforços para os aspectos específicos da solução, etc. Contudo, conforme discutido anteriormente, o reuso, principalmente de código proveniente de terceiros, pode trazer complicações adicionais ao desenvolvimento. No caso de

framework, como muitas vezes ele assume o fluxo da aplicação[38], para alterar os processos de execução pode ser necessário alterar o código do framework.

A construção de um framework não é simples, e deve ser planejada para o reuso [Mattsson, 2000]. O framework deve ser: flexível (reusar as abstrações em diversos contextos), extensível (permitir a adição ou modificação de funcionalidades) e compreensível (bem documentado, seguindo padrões e provendo exemplos de utilização) [Gimenes & Huzita, 2005]. Para desenvolver um framework, deve-se identificar e caracterizar o domínio do problema e definir a arquitetura, o projeto da solução e as maneiras de interagir e estender o framework. Como é muito difícil prever em um primeiro momento todas variabilidades e requisitos de um framework, seu desenvolvimento normalmente é feito iterativamente.

Um framework normalmente é implementado orientado a objetos, composto de classes abstratas e concretas, interfaces e arquivos de configuração. Para especializar o framework utiliza-se herança, composição ou configuração. Os pontos de extensão do framework são chamados *hot-spots* ou *plug points* [Johnson, 1997; Govoni, 1999]. O reuso do framework, também chamado de instanciação, consiste em preencher os hot-spots para obter a aplicação final. As partes do framework que são invariantes são chamadas de frozen-spots.

A construção de um framework ou componente pode ser feita independente dos outros elementos, desde que mantida a interface e as propriedades requeridas pela arquitetura. Um framework pode ser instanciado a partir de componentes e um componente pode ser construído a partir de um framework [Oliveira, 2001]. Muitas vezes frameworks OO são utilizados para gerar uma família de componentes. Da mesma forma, frameworks complementares podem ser utilizados, tanto no nível de aplicação quanto no nível do domínio. Pode-se, por exemplo, utilizar um framework de infra-estrutura para a visão, um outro para persistência e o componente estar estruturado de acordo com um framework de domínio.

[38] A inversão de controle costuma ser utilizada em frameworks para reduzir o acoplamento entre a aplicação e o framework. Este princípio é chamado na literatura de princípio de Hollywood: “Não nos ligue, nós ligamos para você” [Larman, 2004]. Em vez de a aplicação chamar o framework, o framework chama a aplicação em métodos pré-definidos.

A.10. Considerações Finais

Nas linguagens de programação, o suporte ao reuso de código iniciou-se com a possibilidade do agrupamento de funcionalidade, onde linhas de código são encapsuladas e rotuladas em unidades denominadas procedimentos ou funções para serem reusadas em diversos pontos do código [Salus, 1998]. Com o tempo, passou-se a perceber a utilidade de agrupar funções relacionadas na forma de bibliotecas, para serem reusadas entre aplicações distintas. Linguagens de programação orientadas a objetos aumentaram a possibilidade de reuso através de unidades denominadas classes, que encapsulam dados e funções e possibilitam a herança de código e o polimorfismo de objetos. Da mesma maneira que o reuso de funções específicas evoluiu para bibliotecas de funções, o reuso de conjuntos relacionados de classes evoluiu para frameworks e componentes [Oliveira, 2001]. Um framework provê um conjunto genérico de classes que deve ser completado para instanciar uma aplicação específica, enquanto o uso de componentes possibilita construir um sistema compondo-o a partir de unidades de execução.

A idéia de que o software deveria ser componentizado surgiu na conferência da NATO, que lançou o termo Engenharia de Software, na Alemanha em 1968. Nesta conferência, Doug McIlroy [1968], em um artigo intitulado “Mass Produced Software Components”, levantou a necessidade de agrupar coerentemente rotinas relacionadas de modo a montar componentes reusáveis em diversos sistemas. Este reuso, além de economizar esforço de desenvolvimento, possibilitaria a produção de programas a partir de partes já testadas e amplamente utilizadas [Gimenes & Huzita, 2005]. O surgimento dos componentes para interface com o usuário, popularizado em 1992, com o Visual Basic eXtension, ou VBX, ajudou a alavancar a utilização de componentes [Oliveira, 2001]. Diversos estudos apontam a tecnologia de componentes como promissora para melhorar o reuso e a manutenibilidade de software [Szyperski, 1997, p.18].

Componentes de software são autocontidos (são relativamente independentes, fracamente acoplados e encapsulam seu projeto e implementação, incluindo dados e funcionalidades); reusáveis (podem ser utilizados por terceiros, que não necessariamente precisam ter acesso ao seu código fonte); substituíveis

(podem ser trocados por outros componentes compatíveis) e provêem serviços específicos de uma maneira coesa, bem definida e padronizada. Um componente de software pode ser desenvolvido e mantido independentemente e interage com outros componentes ou com a infra-estrutura da aplicação. A orientação a objetos é uma das melhores maneiras de implementar componentes [Szyperski, 1997, p.13; D'Souza & Wills, 1998].

A montagem de sistemas a partir de componentes está no meio termo entre configuração e programação [Morch, 1997]. É mais alto nível do que a programação, mas não tão limitada quanto a configuração. Através do uso de componentes, o sistema pode ser estendido para acompanhar a evolução dos processos de trabalho, das necessidades e da experiência com o uso do sistema. Com componentes, são reduzidas as interdependências fixas no código fonte da aplicação, melhorando a extensibilidade e facilitando a adaptação e extensão.

A componentização normalmente é disponível apenas aos desenvolvedores, que liberam diferentes versões da aplicação alterando a configuração e os componentes. Quando for necessário prover flexibilidade para o usuário final, pode-se disponibilizar a ele o conceito de componente, bem como os mecanismos para instalar e desinstalar componentes da aplicação. Estes componentes muitas vezes são do estilo "plug & play", que imediatamente se tornam disponíveis para uso após o *deployment*. Ao invés de apresentar ao usuário uma aplicação que coloca um prego ou um parafuso na parede, ele é equipado com uma caixa de ferramentas contendo, entre outras coisas, martelo, chave de fenda e parafusadeira, de onde ele escolhe a ferramenta mais adequada às suas preferências e à tarefa em questão. Adicionalmente, ferramentas existentes podem ser combinadas para formar novas ferramentas mais complexas para a realização de tarefas específicas. Entretanto, esta flexibilidade para o usuário final deve ser utilizada com cautela, pois abre espaço para "aberrações", combinando componentes incompatíveis e destoantes.

Apêndice B
Descrição dos Componentes de Colaboração

A deficiência na documentação gera um acréscimo na dificuldade de entendimento e reuso dos componentes [D'Souza & Wills, 1998]. Este apêndice contém a descrição dos componentes de colaboração, classificados em comunicação, coordenação e cooperação. A estrutura de documentação utilizada para a descrição é uma adaptação do formato utilizado para descrever padrões de projeto [Paludo & Burnett, 2005]: para cada componente são descritos nome, intenção, aplicabilidade, variabilidade, estrutura, usos conhecidos e componentes relacionados. Por enquanto, os usos conhecidos limitam-se ao ambiente AulaNet, pois até o momento é o único sistema construído de fato utilizando os componentes 3C. As interfaces dos componentes são representadas utilizando a extensão da UML proposta por D'Souza & Wills [1998], apresentada no Apêndice A.

B.1.Componentes de Comunicação

Os componentes de comunicação oferecem suporte à troca de mensagens, à negociação e à argumentação. A seguir, são descritos os componentes MessageMgr, TextualMediaMgr, DiscreteChannelMgr, MetaInformationMgr, CategorizationMgr e DialogStructureMgr.

B.1.1.MessageMgr

Nome: MessageMgr

Intenção: O componente MessageMgr oferece suporte a mensagens. As mensagens podem ser públicas, privadas, do sistema ou de controle. O componente possibilita o anexo de arquivos na mensagem.

Aplicabilidade: O componente é aplicado nas ferramentas de comunicação síncronas e assíncronas que lidam com mensagens.

Variabilidade: Tipos de mensagens.

Estrutura:

O componente oferece as interfaces IMessageMgr e IMessageMgrConfig. A primeira disponibiliza as operações relativas ao gerenciamento das mensagens. A segunda possibilita a configuração do componente e dos tipos de mensagens. O componente requer o objeto Participant, que pode ser o emissor ou receptor da mensagem, e são disponibilizados os objetos da classe Message, que representa a mensagem em si, e o objeto MessageType, que representa o tipo da mensagem. Caso a mensagem não tenha remetente, a mensagem é do sistema, e caso não tenha destinatário, é endereçada a todos participantes. A mensagem possibilita o anexo de um ou mais arquivos e disponibiliza um corpo textual (*body*), que é utilizado quando não é necessário conteúdo multimídia. Nos casos em que é necessário, o serviço que utiliza este componente deve estender a mensagem. As propriedades da classe Message são implementadas através de gets e sets. Os tipos de mensagens default são PUB (mensagem pública), PVT (mensagem privada) e CTRL (mensagem de controle). A estrutura do componente está representada na Figura B.1.

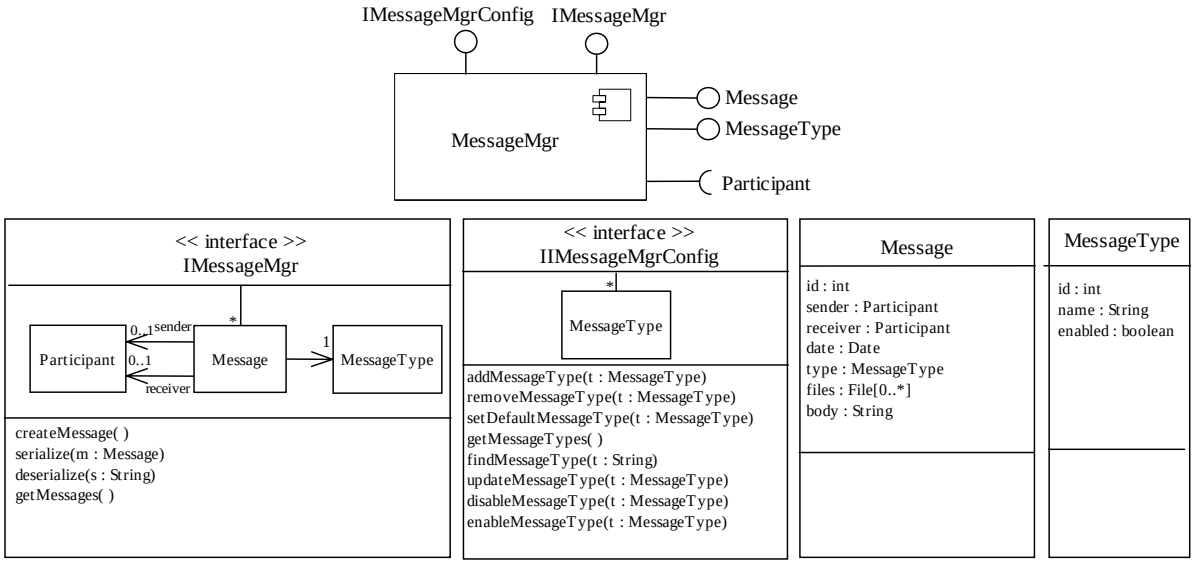

Figura B.1. Componente MessageMgr e suas interfaces

Usos conhecidos: Conferências, Debate, Correio para Turma, Correio para Participante, Mensageiro.

Componentes relacionados: TextualMediaMgr, VideoMediaMgr, AudioMediaMgr, PictorialMediaMgr e ParticipantMgr.

B.1.2.TextualMediaMgr

Nome: TextualMediaMgr

Intenção: Utilizado para mensagens com corpo textual. É possível configurar o tamanho do texto e o vocabulário disponível.

Aplicabilidade: É utilizado quando o corpo das mensagens do serviço necessita de tratamento ou codificação especial.

Variabilidade: É possível configurar o tamanho mínimo e máximo das mensagens e o vocabulário permitido ou proibido.

Estrutura:

A estrutura do componente está representada na Figura B.2. O componente oferece as interfaces ITextualMediaMgr e ITextualMediaMgrConfig, que disponibilizam serviços relativos à utilização e configuração do componente. O componente implementa a interface IMediaContent, que é o contrato geral para os conteúdos multimídia. Não é possível configurar ao mesmo tempo vocabulário permitido e proibido.

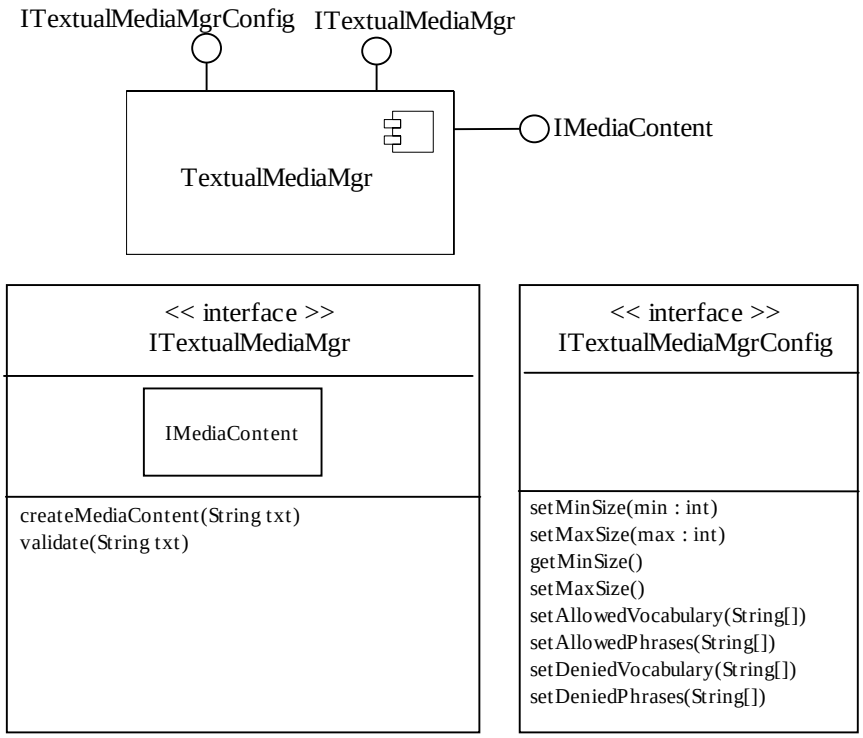

Figura B.2. Componente TextualMediaMgr e suas interfaces

Usos conhecidos: Debate

Componentes relacionados: MessageMgr

B.1.3.DiscreteChannelMgr

Nome: DiscreteChannelMgr

Intenção: Este componente é utilizado para transmitir e filtrar as mensagens. Pode ser configurado para modo síncrono ou assíncrono, que influencia a maneira como as mensagens são tratadas; *push* ou *pull*, que define como a mensagem é entregue ao participante; e o atraso na entrega, que pode ser utilizado para regular a interação em uma ferramenta síncrona [Pimentel et al., 2005].

Aplicabilidade: É utilizado quando se deseja filtrar mensagens para um determinado participante, utilizar uma interface rica (RIA) no lado cliente ou fazer *push* de mensagens (enviar sem o participante solicitar explicitamente), modo que é utilizado principalmente nas ferramentas de comunicação síncronas.

Variabilidade: Modo de entrega e atraso no envio.

Estrutura:

A estrutura do componente está representada na Figura B.3. A utilização do componente é feita pela interface IDiscreteChannelMgr, que é responsável pelo envio das mensagens para os ouvintes, que são configurados através da interface IDiscreteChannelMgrConfig.

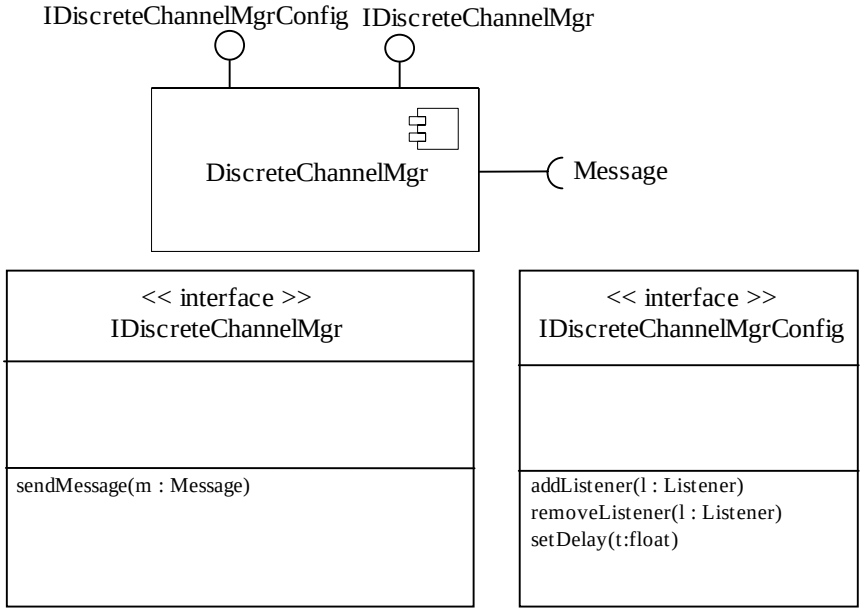

Figura B.3. Componente DiscreteChannelMgr e suas interfaces

Usos conhecidos: Debate, Correio para Participante e Mensageiro.

Componentes relacionados: MessageMgr

B.1.4.MetaInformationMgr

Nome: MetaInformationMgr

Intenção: Gerencia meta-informações associadas às mensagens.

Aplicabilidade: É utilizado quando se deseja disponibilizar meta-informações a partir das quais se deseja localizar e filtrar as mensagens.

Variabilidade: As meta-informações e sua obrigatoriedade.

Estrutura:

A estrutura do componente está representada na Figura B.4. As meta-informações são configuradas através da interface IMetaInformationMgrConfig, enquanto a atribuição das meta-informação às mensagens e a busca das

mensagens a partir das meta-informações é feita através da interface IMetaInformationMgr.

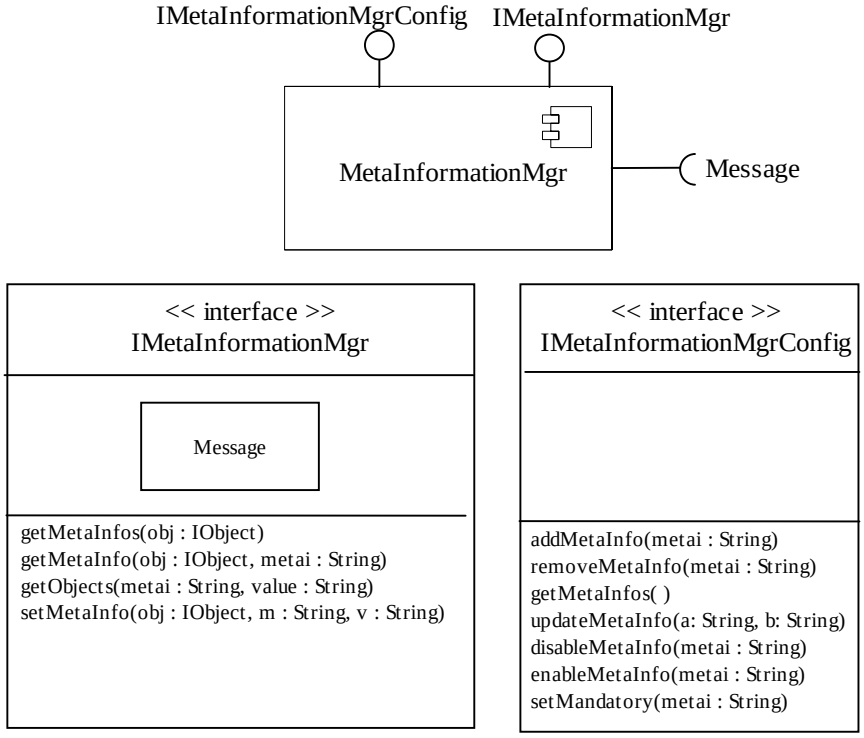

Figura B.4. Componente MetaInformationMgr e suas interfaces

Usos conhecidos: -

Componentes relacionados: MessageMgr

B.1.5.CategorizationMgr

Nome: CategorizationMgr

Intenção: Gerencia a categorização de mensagens.

Aplicabilidade: A categorização de mensagens oferece um novo elemento à linguagem da comunicação. Os participantes contextualizam a interpretação da mensagem a partir da categoria selecionada. A categorização de mensagens pode ser utilizada para complementar a estruturação do diálogo, oferecendo semântica aos inter-relacionamentos das mensagens [Gerosa et al., 2001].

Variabilidade: Pode-se configurar o conjunto de categorias e a categoria genérica.

Estrutura:

O componente CategorizationMgr possui duas interfaces fornecidas (ICategorizationMgr e ICategorizationMgrConfig) e uma interface requerida (ICategorizableObj), conforme ilustrado na Figura B.5. A interface ICategorizationMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, atribuição de categoria, recuperação dos objetos, etc. A interface ICategorizationMgrConfig disponibiliza os serviços relativos à configuração do componente, como manipulação do conjunto de categorias e seleção da categoria genérica. O componente pode ser configurado através desta interface e por seu arquivo descritor. O componente lida com objetos categorizáveis, que implementem a interface ICategorizableObj.

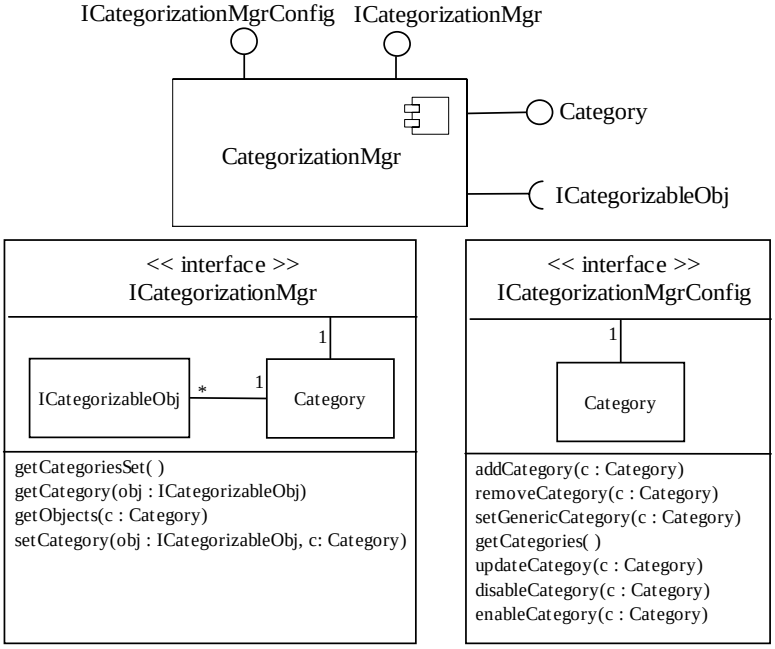

Figura B.5. Componente CategorizationMgr e suas interfaces

Usos conhecidos: Conferências e Correio para Turma.

Componentes relacionados: MessageMgr

B.1.6.DialogStructureMgr

Nome: DialogStructureMgr

Intenção: Gerencia as inter-relações entre as mensagens, que podem ser na forma de lista, árvore ou grafo.

Aplicabilidade: Este componente é utilizado principalmente nas ferramentas que adotam uma estruturação hierárquica ou em rede.

Variabilidade: O tipo de estruturação utilizado e a ordem de percurso da estrutura.

Estrutura:

A estrutura do componente está representada na Figura B.6. Através da interface IDialogStructureMgrConfig, o componente é configurado, e através da interface IDialogStructureMgr o componente é utilizado. É possível incluir as mensagens e obter os pais e filhos de uma mensagem e as raízes do diálogo.

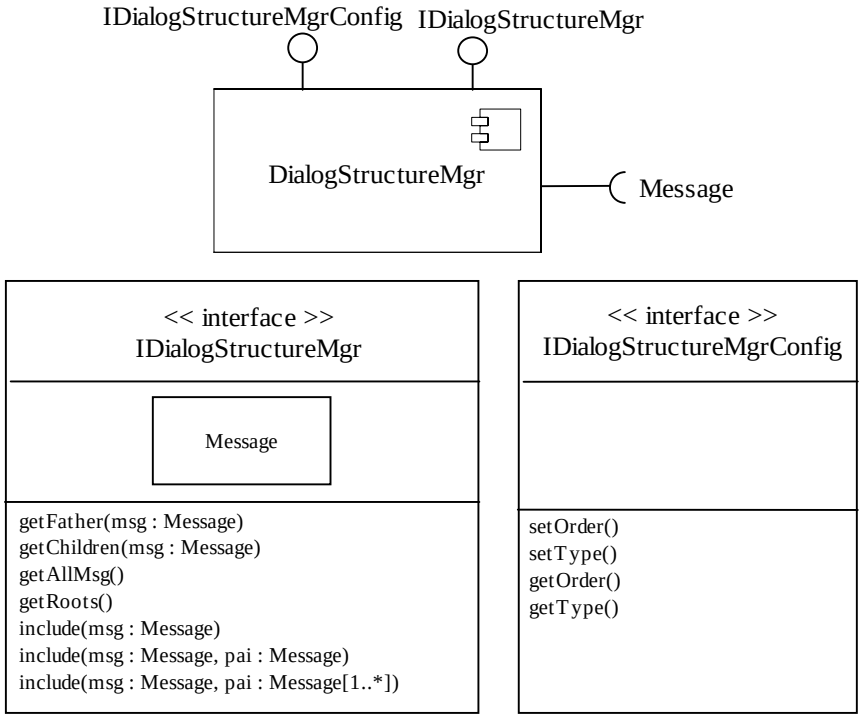

Figura B.6. Componente DialogStructureMgr e suas interfaces

Usos conhecidos: Correio para Turma, Correio para Participante e Conferências.

Componentes relacionados: MessageMgr

B.2.Componentes de Coordenação

Os componentes de coordenação oferecem suporte à organização do grupo, ao gerenciamento do acesso dos participantes e à distribuição de tarefas atribuídas. A seguir, são descritos os componentes AssessmentMgr, RoleMgr, PermissionMgr, ParticipantMgr, GroupMgr, SessionMgr, FloorControlMgr, TaskMgr, AwarenessMgr, AvailabilityMgr e NotificationMg.

B.2.1.AssessmentMgr

Nome: AssessmentMgr

Intenção: Oferece suporte à avaliação qualitativa, possibilitando a atribuição de notas aos objetos.

Aplicabilidade: A avaliação de mensagens possibilita acompanhar e incentivar a participação com qualidade nas tarefas colaborativas.

Variabilidade: Regras da avaliação, modelo de avaliação e fator de correção pela quantidade de mensagens.

Estrutura:

O componente AssessmentMgr oferece a interface para configuração do componente IAssessmentMgrConfig e para utilização IAssessmentMgr. As avaliações são referentes aos objetos de cooperação. A Figura B.7 ilustra a estrutura do componente.

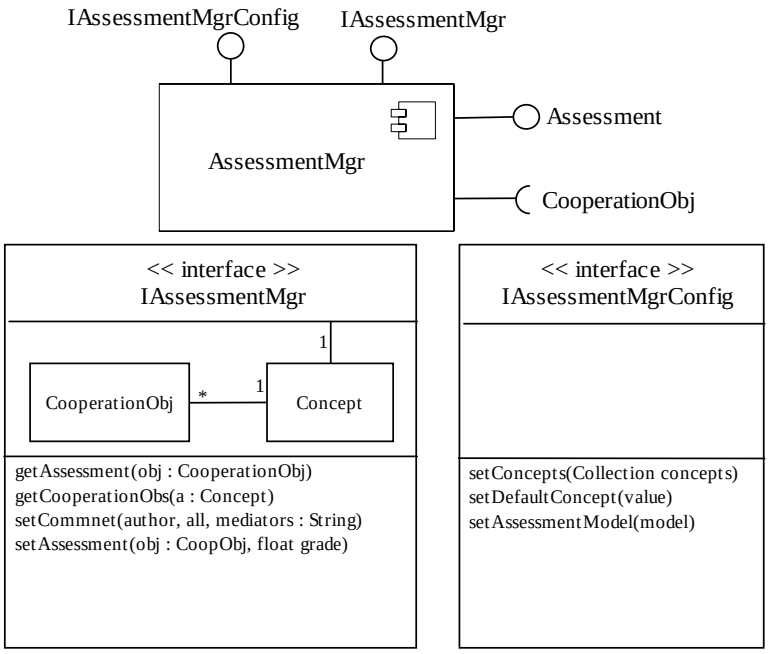

Figura B.7. Componente AssessmentMgr e suas interfaces

Usos conhecidos: Conferências, Correio para Turma e Debate.

Componentes relacionados: CooperationObjMgr

B.2.2.RoleMgr

Nome: RoleMgr

Intenção: Cuida do gerenciamento dos papéis atribuídos aos participantes.

Aplicabilidade: O gerenciamento de papéis possibilita a atribuição de permissões e tarefas específicas para determinadas funções dos participantes.

Variabilidade: Pode-se configurar o conjunto de papéis.

Estrutura:

O componente RoleMgr possui duas interfaces fornecidas (IRoleMgrMgr e IRoleMgrConfig) e uma interface requerida (Participant), conforme ilustrado na Figura B.8. A interface IRoleMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, atribuição dos papéis, recuperação dos participantes de um papel, etc. A interface IRoleMgrMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição do conjunto de papéis. O componente pode ser configurado através desta interface e por seu arquivo descritor.

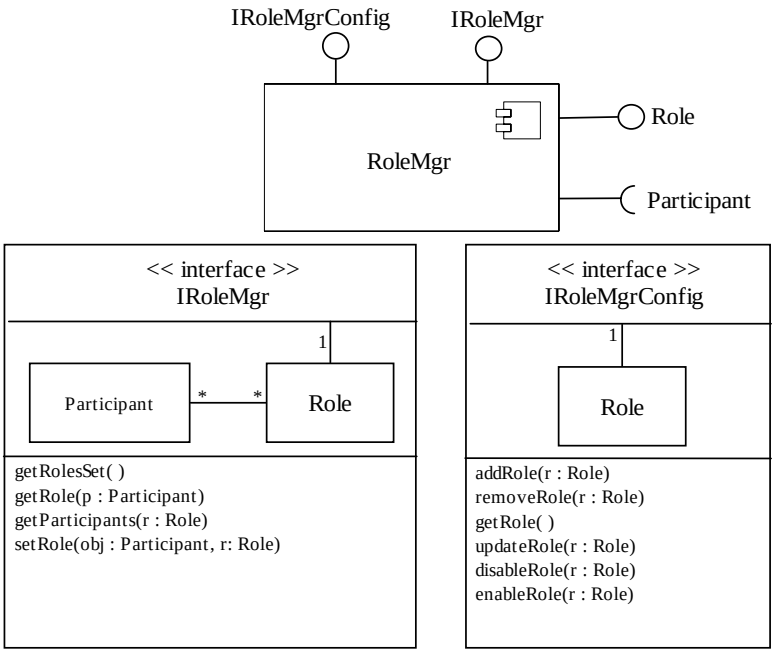

Figura B.8. Componente RoleMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet (funcionalidades específicas para os mediadores e aprendizes).

Componentes relacionados: ParticipantMgr

B.2.3.PermissionMgr

Nome: PermissionMgr

Intenção: Gerencia as permissões de execução de ações.

Aplicabilidade: Prover a segurança de acesso aos recursos e funcionalidades do sistema. Cada componente do sistema pode registrar neste componente as ações sobre as quais se deseja controlar permissões e as permissões de um determinado papel.

Variabilidade: Conjunto de ações de um serviço.

Estrutura:

O componente PermissionMgr possui duas interfaces fornecidas (IPermissionMgr e IPermissionMgrConfig), duas interfaces requeridas (Action e Role), conforme ilustrado na Figura B.9. A interface IPermissionMgr disponibiliza os serviços relativos à utilização do componente, como atribuição das permissões de um papel, referente às ações do serviço em questão, teste de

permissão, etc. A interface IPermissionMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição das ações. O componente pode ser configurado através desta interface e por seu arquivo descritor.

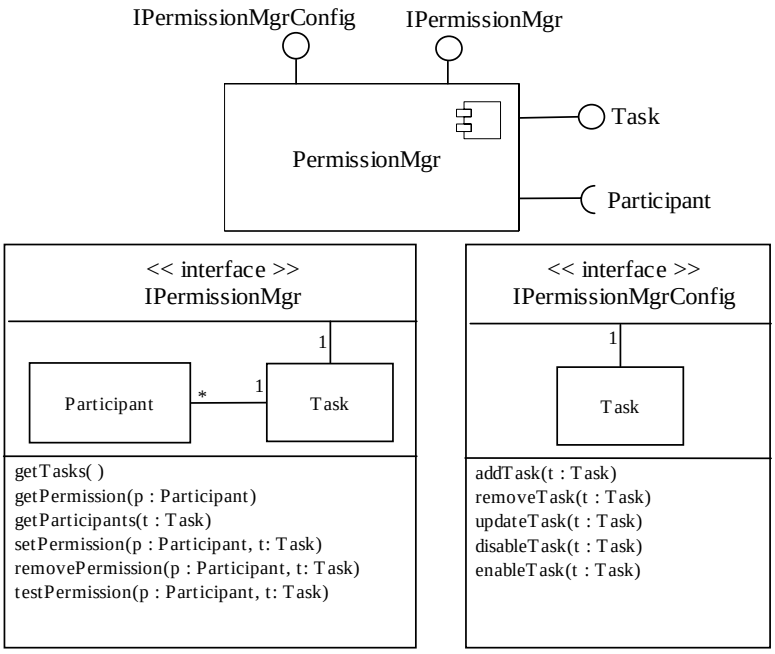

Figura B.9. Componente PermissionMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: -

B.2.4.ParticipantMgr

Nome: ParticipantMgr

Intenção: Gerencia os participantes da colaboração.

Aplicabilidade: O componente é utilizado para gerenciar a representação dos participantes.

Variabilidade: Pode-se configurar as informações que deseja-se registrar sobre os participantes.

Estrutura:

O componente ParticipantMgr possui três interfaces fornecidas (IParticipantMgr, IParticipantMgrConfig e Participant), conforme ilustrado na

Figura B.10. A interface IParticipantMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, criação, remoção e atualização de participante, busca, etc. A interface IParticipantMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição das informações a serem registradas para os participantes.

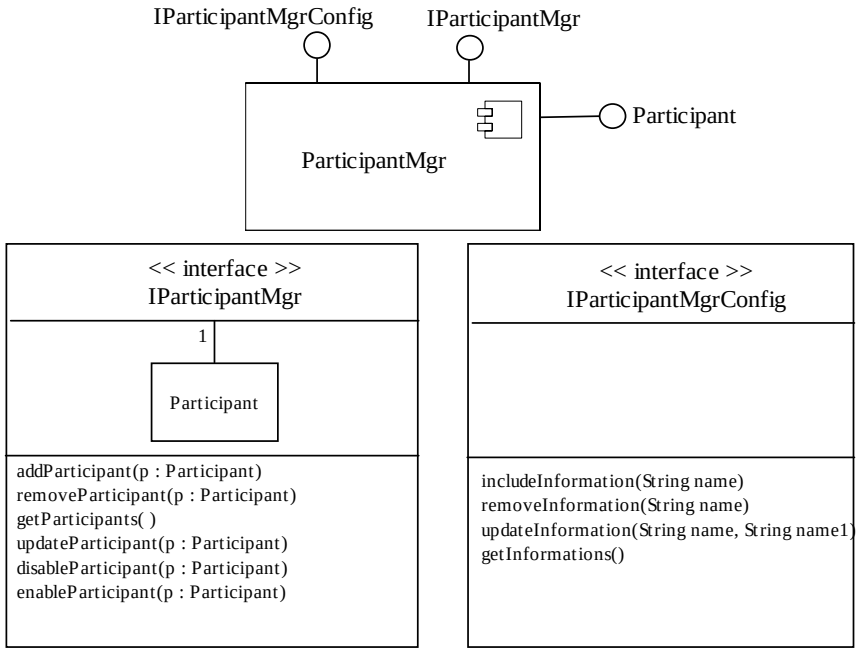

Figura B.10. Componente ParticipantMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: -

B.2.5.GroupMgr

Nome: GroupMgr

Intenção: Possibilita a definição e gestão de grupos e subgrupos de participantes.

Aplicabilidade: A colaboração pressupõe um grupo de participantes. Este componente possibilita organizar os participantes em grupos e subgrupos. Pode-se representar informações adicionais para o participante no escopo do grupo.

Variabilidade: tamanho do grupo

Estrutura:

O componente GroupMgr possui três interfaces fornecidas (IGroupMgr, IGroupMgrConfig e Group) e uma interface requerida (Participant), conforme ilustrado na Figura B.11. A interface IGroupMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, inclusão de participante em grupo, divisão de grupo, etc. A interface IGroupMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição dos grupos disponíveis. O componente pode ser configurado através desta interface e por seu arquivo descritor.

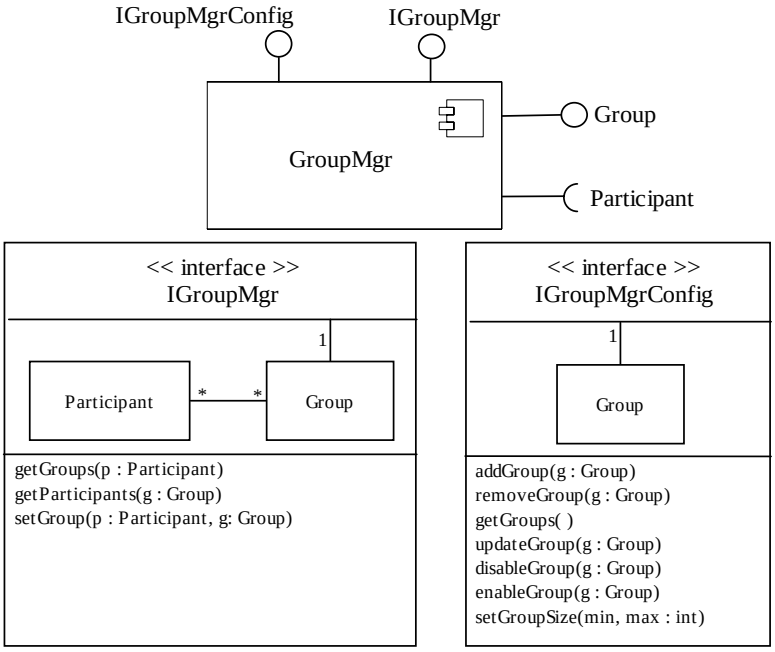

Figura B.11. Componente GroupMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: -

B.2.6.SessionMgr

Nome: SessionMgr

Intenção: Gerencia as sessões de colaboração.

Aplicabilidade: Possibilita definir a sessão, que agrega participantes, objetos compartilhados durante um período do tempo. É possível configurar a data de início e fim da sessão e o modo de entrada.

Variabilidade: Duração

Estrutura:

O componente SessionMgr possui três interfaces fornecidas (ISessionMgr, ISessionMgrConfig e Session) e requer as interfaces Group, Participant e CooperationObj, conforme ilustrado na Figura B.12. A interface ISessionMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, inclusão de participante ou objeto na sessão, verificação de disponibilidade, bloquear e desbloquear, etc. A interface ISessionMgrConfig disponibiliza os serviços relativos à configuração do componente, como criação da sessão e definição de suas propriedades. O componente pode ser configurado através desta interface e por seu arquivo descritor.

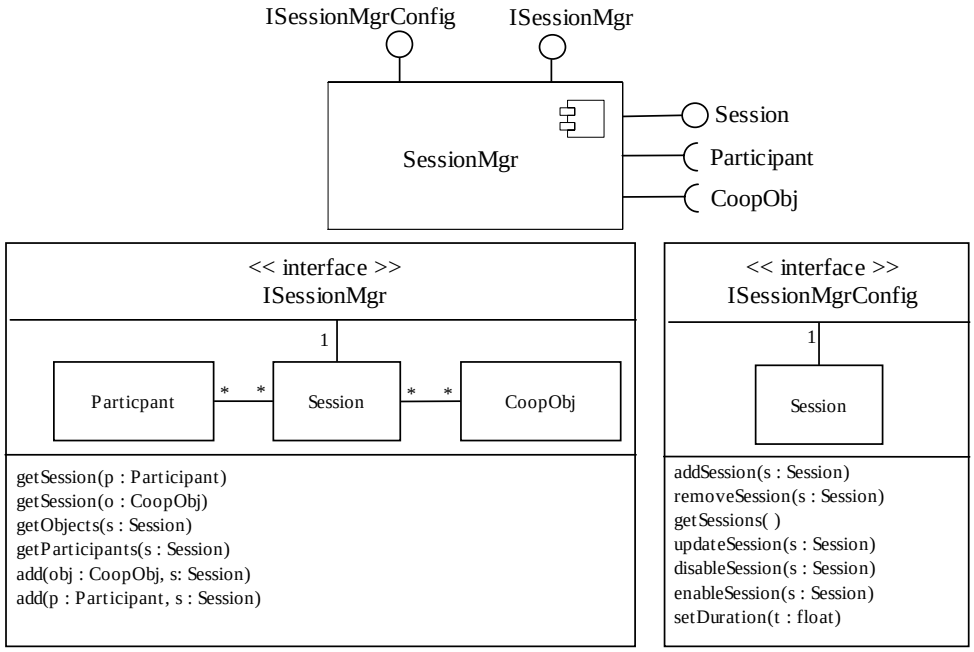

Figura B.12. Componente SessionMgr e suas interfaces

Usos conhecidos: Conferências, Debate, Correio para Turma e Correio para Participante.

Componentes relacionados: CooperationObjMgr, ParticipantMgr, GroupMgr.

B.2.7.FloorControlMgr

Nome: FloorControlMgr

Intenção: Gerencia a ordem de participação. Podem ser alocadas várias políticas de acesso.

Aplicabilidade: A definição da ordem de participação é útil para organizar a discussão ou a atuação em um espaço compartilhado.

Variabilidade: Pode-se a técnica desejada.

Estrutura:

O componente FloorControlMgr possui duas interfaces fornecidas (IFloorControlMgr e IFloorControlMgrConfig), conforme ilustrado na Figura B.13. A interface IFloorControlMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, recuperar a fila de participação, alterar a fila, alterar a técnica, etc. A interface IFloorControlMgrConfig disponibiliza os serviços relativos à configuração do componente, que incluem a definição das técnicas de participação. O componente pode ser configurado através desta interface e por seu arquivo descritor.

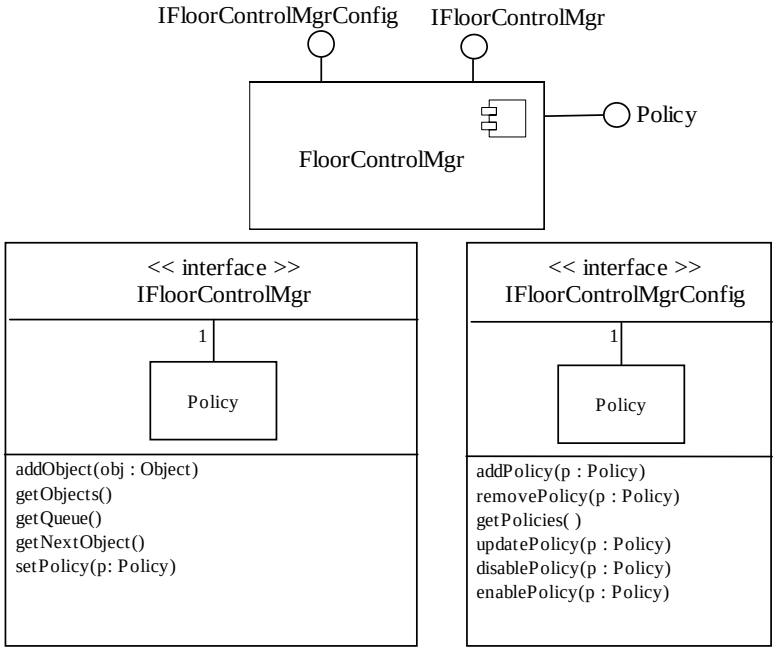

Figura B.13. Componente FloorControlMgr e suas interfaces

Usos conhecidos: Debate.

Componentes relacionados: -

B.2.8.TaskMgr

Nome: TaskMgr

Intenção: Possibilita o gerenciamento das tarefas do grupo.

Aplicabilidade: O gerenciamento das tarefas possibilita o acompanhamento da realização e da produtividade dos participantes.

Variabilidade: Podem ser configurados os tipos de tarefas.

Estrutura:

O componente TaskMgr possui duas interfaces fornecidas (ITaskMgr e ITaskMgrConfig) e duas interfaces requeridas (Task e Participant), conforme ilustrado na Figura B.14. A interface ITaskMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, atribuição de tarefas a participantes, mudança de status de tarefa, etc. A interface ITaskMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição do conjunto de tarefas e seus tipos. O componente pode ser configurado através desta interface e por seu arquivo descritor.

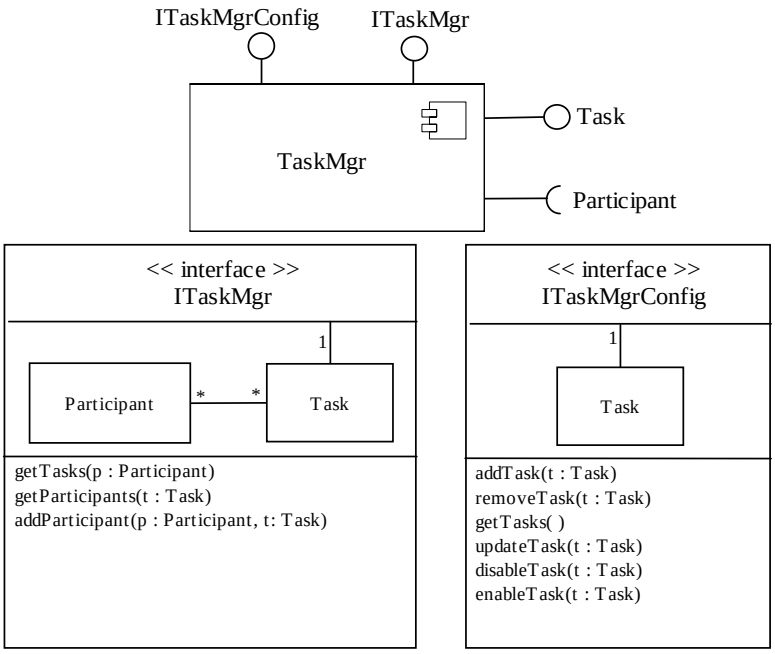

Figura B.14. Componente TaskMgr e suas interfaces

Usos conhecidos: Serviço Tarefas.

Componentes relacionados: ParticipantMgr

B.2.9.AwarenessMgr

Nome: AwarenessMgr

Intenção: Gerencia as informações de percepção em geral, registrando e operando sobre os eventos ocorridos no serviço.

Aplicabilidade: As informações de percepção contextualizam o trabalho em grupo. Alguns exemplos de informações: o que já foi feito, o que falta fazer, novidades, presença de participante, novidades desde a última visita, etc. As informações de percepção são especialmente úteis para embasar filtros utilizados na computação móvel, visto que a informação deve ser mais precisa e focada, dadas as restrições de tamanho de tela, banda e qualidade de conexão [Filippo et al., 2005].

Variabilidade: Pode-se configurar os tipos de eventos.

Estrutura:

O componente AwarenessMgr possui quatro interfaces fornecidas (IAwarenessMgr, IAwarenessMgrConfig, Event e EventType) e uma interface requerida (Participant), conforme ilustrado na Figura B.15. A interface IAwarenessMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, criação e recuperação de eventos. A interface IAwarenessMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição dos tipos de eventos. O componente pode ser configurado através desta interface e por seu arquivo descritor.

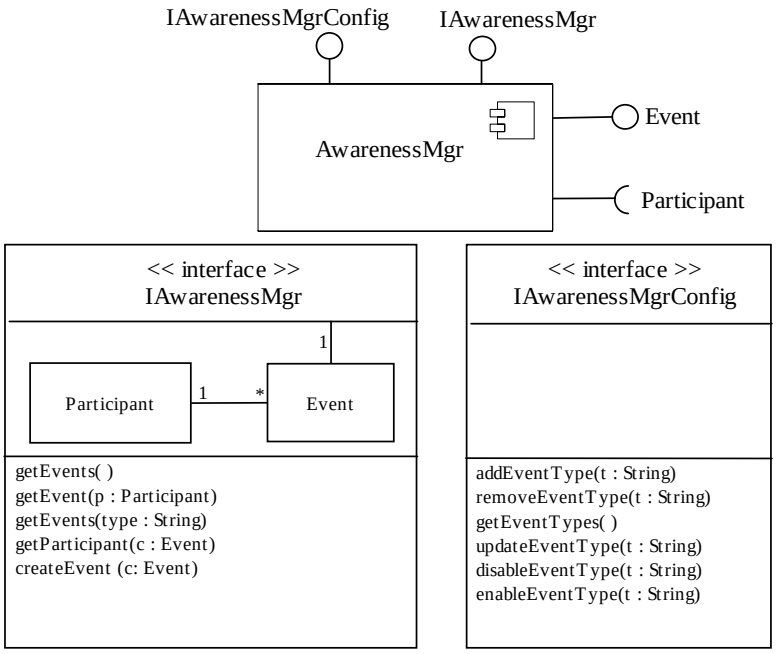

Figura B.15. Componente AwarenessMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: ParticipantMgr

B.2.10.AvailabilityMgr

Nome: AvailabilityMgr

Intenção: Possibilita que o participante configure sua disponibilidade.

Aplicabilidade: Saber a disponibilidade dos participantes contribui para sua melhor organização, diminuindo a quantidade de vezes que os participantes são interrompidos em horas impróprias.

Variabilidade: Pode-se configurar os graus de disponibilidade.

Estrutura:

O componente AvailabilityMgr possui duas interfaces fornecidas (IAvailabilityMgr e IAvailabilityMgrConfig) e uma interface requerida (Participant), conforme ilustrado na Figura B.16. A interface IAvailabilityMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, mudança de disponibilidade, consulta de disponibilidade, etc. A interface IAvailabilityMgrConfig disponibiliza os serviços relativos à configuração do

componente, incluindo a definição dos níveis de disponibilidade. O componente pode ser configurado através desta interface e por seu arquivo descritor.

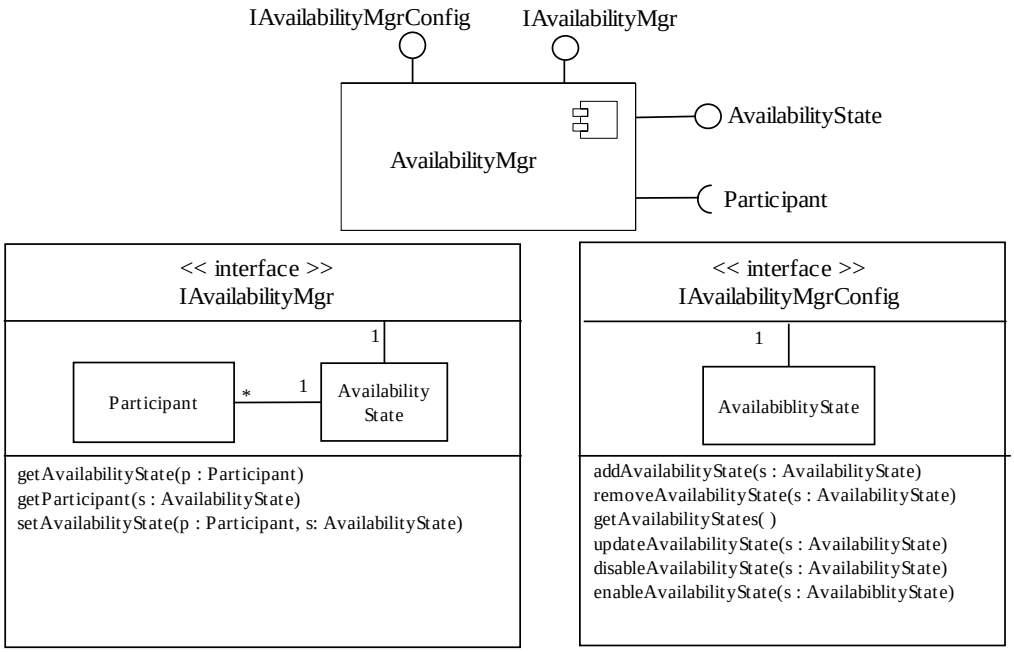

Figura B.16. Componente AvailabilityMgr e suas interfaces

Usos conhecidos: -

Componentes relacionados: ParticipantMgr

B.2.11.NotificationMgr

Nome: NotificationMgr

Intenção: Gerencia o envio das notificações aos participantes.

Aplicabilidade: Um canal único de notificação para os diversos serviços favorece o estabelecimento de modos de recebimento para cada participante e filtros específicos.

Variabilidade: Pode-se configurar os tipos de notificações.

Estrutura:

O componente NotificationMgr possui duas interfaces fornecidas (INotificationMgr e INotificationMgrConfig) e uma interface requerida (Participant), conforme ilustrado na Figura B.17. A interface INotificationMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo,

envio de notificação, recuperação de notificações anteriores, etc. A interface INotificationMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição dos tipos de notificações, cabeçalho genérico, etc. O componente pode ser configurado através desta interface e por seu arquivo descritor.

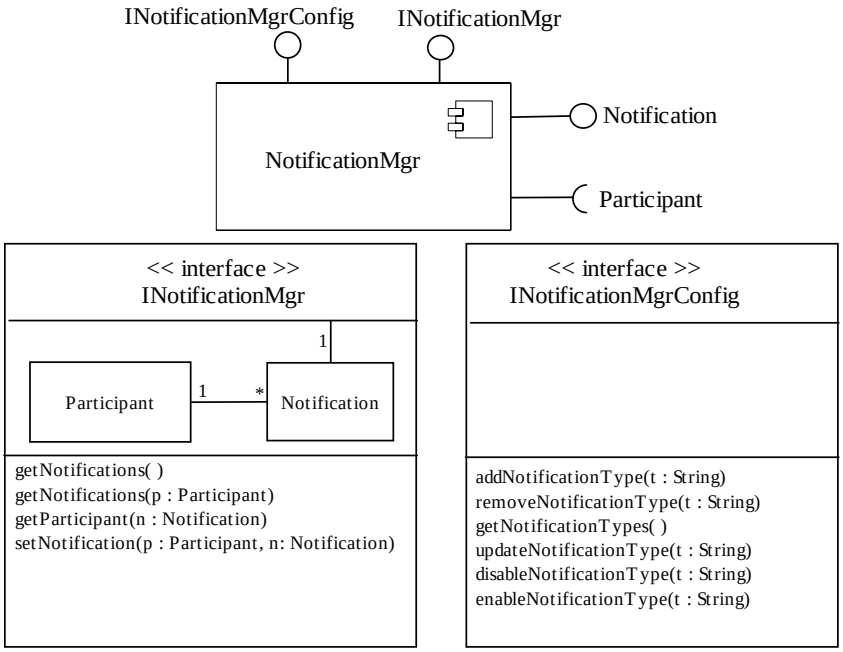

Figura B.17. Componente NotificationMgr e suas interfaces

Usos conhecidos: Conferências e Correio para Turma.

Componentes relacionados: ParticipantMgr

B.3.Componentes de Cooperação

Os componentes de cooperação oferecem suporte aos objetos compartilhados e sua manipulação. A seguir, são descritos os componentes CooperationObjMgr, SearchMgr, StatisticalAnalysisMgr, ActionLogMgr, AccessRegistrationMgr.

B.3.1.CooperationObjMgr

Nome: CooperationObjMgr

Intenção: Provê mecanismos de compartilhamento e concorrência aos objetos compartilhados. Possibilita também a persistência dos objetos.

Aplicabilidade: Os objetos compartilhados são manipulados ao longo da colaboração. Este componente oferece suporte a gestão destes objetos.

Variabilidade: -

Estrutura:

O componente CooperationObjMgr possui três interfaces fornecidas (ICooperationObjMgr e ICooperationObjMgrConfig) e uma interface requerida (Participant), conforme ilustrado na Figura B.18. A interface ICooperationObjMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, criação e recuperação dos objetos. A interface ICooperationObjMgrConfig disponibiliza os serviços relativos à configuração do componente. O componente pode ser configurado através desta interface e por seu arquivo descritor.

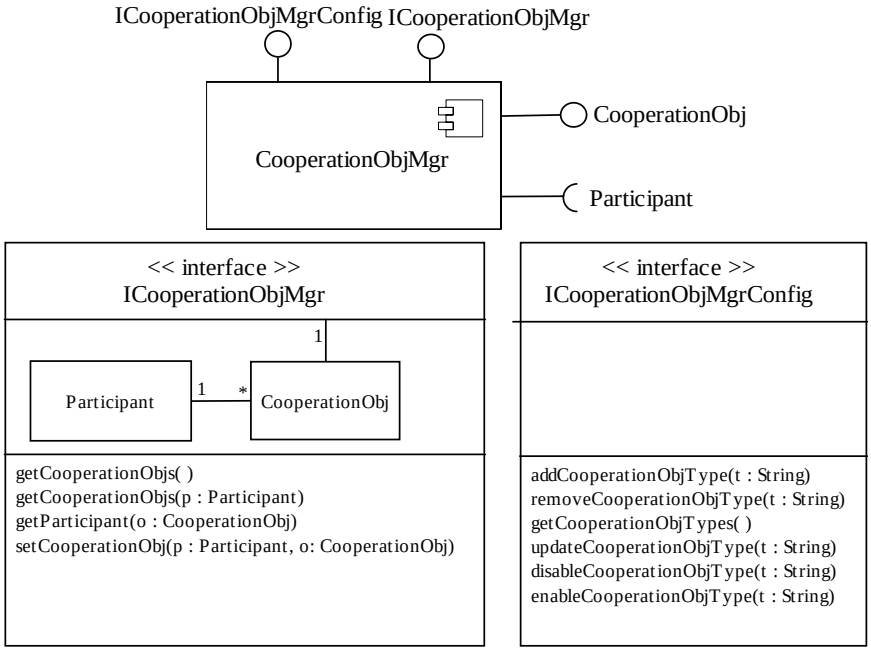

Figura B.18. Componente CooperationObjMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: ParticipantMgr

B.3.2.SearchMgr

Nome: SearchMgr

Intenção: Provê mecanismos de busca para os objetos compartilhados.

Aplicabilidade: Possibilitar utilizar critérios de busca para recuperar os objetos de cooperação de um repositório.

Variabilidade: Critérios de busca

Estrutura:

O componente SearchMgr possui duas interfaces fornecidas (ISearchMgr e ISearchMgrConfig) e uma interface requerida (CooperationObj), conforme ilustrado na Figura B.19. A interface ISearchMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, recuperação dos objetos. A interface ISearchMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição do conjunto de critérios disponívies. O componente pode ser configurado através desta interface e por seu arquivo descritor.

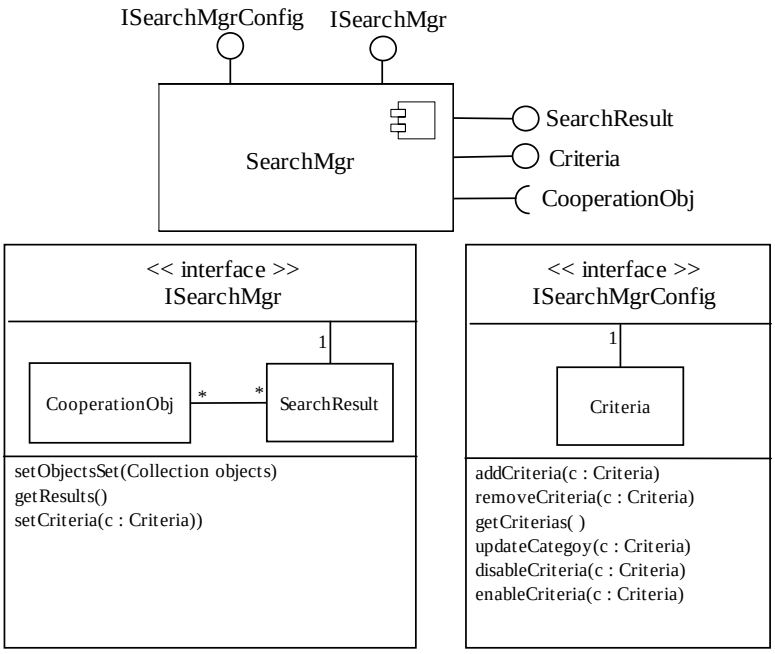

Figura B.19. Componente SearchMgr e suas interfaces

Usos conhecidos: Conferências.

Componentes relacionados: CooperationObjMgr

B.3.3.StatisticalAnalysisMgr

Nome: StatisticalAnalysisMgr

Intenção: Oferece funcionalidades de análise estatística dos objetos compartilhados.

Aplicabilidade: Oferece análises estatísticas sobre os objetos registrados, de modo a embasar a toma de decisão e a coordenação do grupo [Gerosa et al., 2005].

Variabilidade: -

Estrutura:

O componente StatisticalAnalysisMgr possui duas interfaces fornecidas (IStatisticalAnalysisMgr e IStatisticalAnalysisMgrConfig), conforme ilustrado na Figura B.20. A interface IStatisticalAnalysisMgr disponibiliza os serviços relativos à utilização do componente e a interface IStatisticalAnalysisMgrConfig disponibiliza os serviços relativos à configuração do componente. O componente pode ser configurado através desta interface e por seu arquivo descritor.

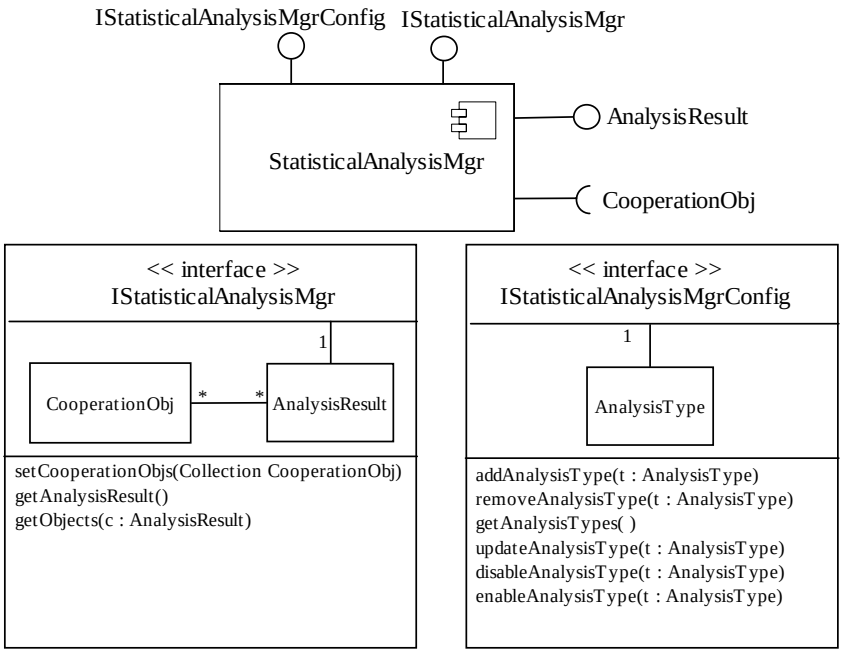

Figura B.20. Componente StatisticalAnalysisMgr e suas interfaces

Usos conhecidos: Conferências e Correio para Turma.

Componentes relacionados: CooperationObjMgr

B.3.4.ActionLogMgr

Nome: ActionLogMgr

Intenção: Possibilita registrar o histórico de ações no serviço.

Aplicabilidade: O registro do log de ações possibilita fazer auditoria, acompanhar o uso, voltar a situações anteriores, rastrear problemas, otimizar processos, etc.

Variabilidade: Pode ser configurado o nível e o tipo de log a ser gerado.

Estrutura:

O componente ActionLogMgr possui duas interfaces fornecidas (IActionLogMgr e IActionLogMgrConfig), conforme ilustrado na Figura B.21. A interface IActionLogMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, registro e recuperação de ações. A interface IActionLogMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição dos tipos e níveis de log. O componente pode ser configurado através desta interface e por seu arquivo descritor.

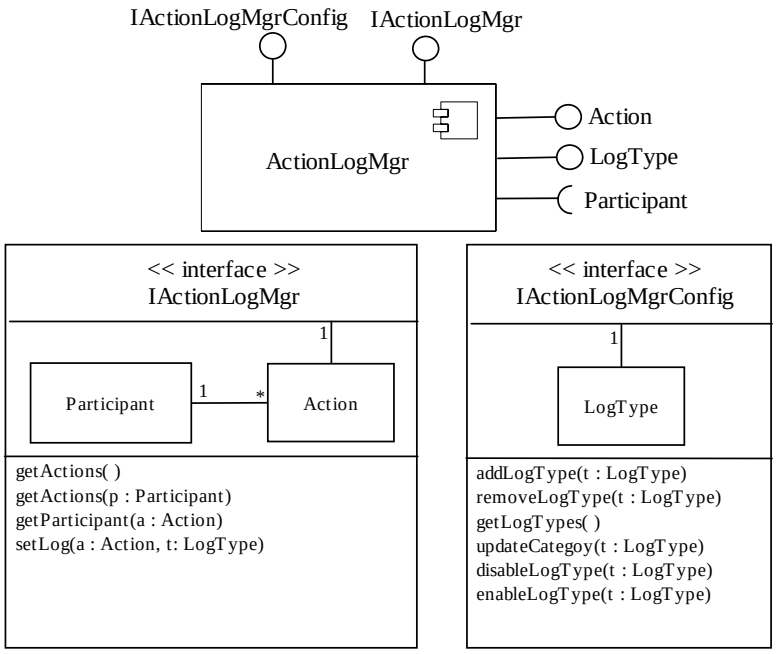

Figura B.21. Componente ActionLogMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: ParticipantMgr

B.3.5.AccessRegistrationMgr

Nome: AccessRegistrationMgr

Intenção: Registra os acessos dos participantes para cada objeto, possibilitando um controle do que já foi visitado.

Aplicabilidade: Registrando o acesso aos objetos é possível exibir diferentemente os conteúdos não visitados, embasando a navegação dos participantes.

Variabilidade: -

Estrutura:

O componente AccessRegistrationMgr possui duas interfaces fornecidas (IAccessRegistrationMgr e IAccessRegistrationMgrConfig) e duas interfaces requeridas (Participant e CooperationObj), conforme ilustrado na Figura B.22. A interface IAccessRegistrationMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, registro e recuperação dos acessos. A interface IAccessRegistrationMgrConfig disponibiliza os serviços relativos à

configuração do componente. O componente pode ser configurado através desta interface e por seu arquivo descritor.

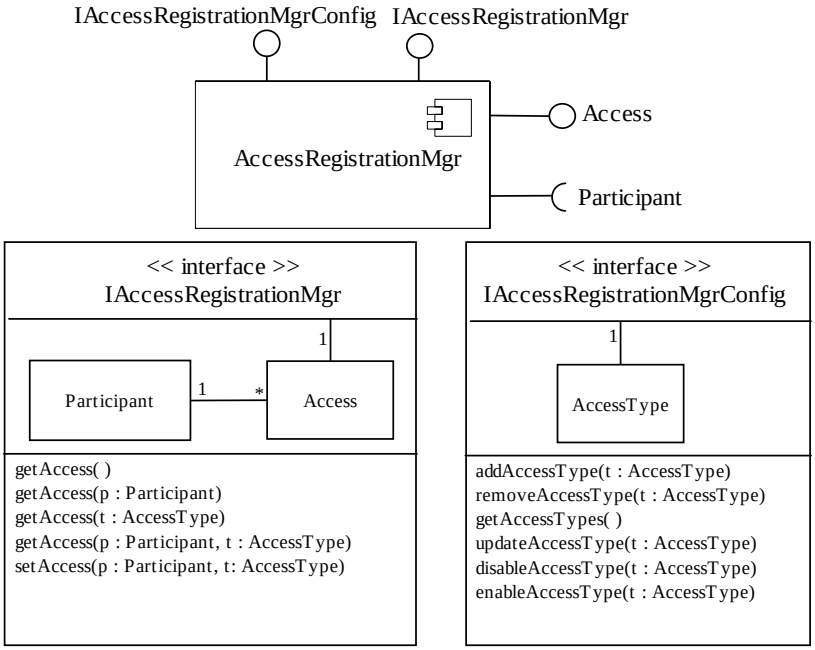

Figura B.22. Componente AccessRegistrationMgr e suas interfaces

Usos conhecidos: Conferências, Correio para Turma e Correio para Participante

Componentes relacionados: ParticipantMgr

Referências Bibliográficas

Allen, J. F. (1984) Towards a General Theory of Action and Time. Artificial Intelligence, 23, 1984, 123-154.

Alur D., Crupi, J. & Malks, D. (2001) Core J2EE Patterns: Best Practices and Design Strategies. Publisher: Prentice Hall / Sun Microsystems Press, 2001.

Amiour, M. & Estublier, J. (1998) A Support for Communication in Software Processes. 10th Conference on Software Engineering and Knowledge Engineering (SEKE'98).

Amiour, M. (1997) "A Support for cooperation in software processes". Doctoral Consortium of the 9th Conference on Advanced Information Systems Engineering (CAiSE'97), Barcelona, Spain.

Anderson, G.E., Graham, T.C. & Wright, T.N. (2000) "Dragonfly: linking conceptual and implementation architectures of multiuser interactive systems", Proceedings of the 22nd international Conference on Software Engineering (ICSE '00), Limerick, Ireland, June 04 - 11, 2000, ACM Press, pp. 252-261.

Andrade, L.V., Sampaio, F.F. & Rocha, L.L.A. (2002) O Modelo GD-IBIS: Grupo de Discussão para Web no Contexto de Educação a Distâncial, XXIX SEMISH, Anais do XXII Congresso da Sociedade Brasileira de Computação. Volume 3, pp. 183-292.

Apperly, H. (2001) The Component Industry Metaphor, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Arango, G. (1994) Domain Analysis Methods. In: Software Reusability, Schäfer, W., Prieto-Diaz, R. & Matsumoto, M. (eds), Chichester, Ellis Horwood, 1994, pp. 17-49.

Araujo, R.M., Santoro, F.M. & Borges, M.R.S. (2004) "A conceptual framework for designing and conducting groupware evaluations", Journal of Computer Applications in Technology (IJCAT), V. 19, N. 3-4, Special Issue on Current Approaches for Groupware Design, Implementation and Evaluation, pp. 139-150.

Araujo, R.M., Santoro, F.M. & Borges, M.R.S. (2004) "The CSCW Lab Ontology for Groupware Evaluation", Proceedings of the 8th Computer Supported Cooperative Work in Design (CSCWiD 2004), pp. 148-153.

Baker, K., Greenberg, S. & Gutwin, C. (2001) Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration. 8th IFIP International Conference (EHCI 2001). Lecture Notes in Computer Science Vol. 2254, pp. 123-139, Springer-Verlag.

Banavar, G., Doddapaneti, S., Miller, K. & Mukherjee, B. (1998) Rapidly Building Synchronous Collaborative Applications by Direct Manipulation. In Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCW'98), pp. 139-148, 1998.

Bandinelli, S., Nitto, E.D. & Fuggetta, A. (1996) "Supporting cooperation in the SPADE-1 Environment", IEEE Transactions on Software Engineering, vol. 22, No. 12, December 1996, pp. 841-865

Barreto, C.G. (2006) Agregando Frameworks de Infra-Estrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente AulaNet, Dissertação de Mestrado, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, 2006.

Barreto, C.G., Fuks, H. & Lucena, C.J.P. (2005) "Agregando Frameworks em uma Arquitetura Baseada em Componentes no Ambiente AulaNet", Anais do 5º Workshop de Desenvolvimento Baseado em Componentes - WDBC 2005, 7-9 de novembro de 2005, Juiz de Fora, MG, ISBN 85-88279-47-9, pp. 25-32.

Barroca, L., Gimenes, I.M.S. & Huzita, E.H.M. (2005) "Conceitos Básicos", in: Desenvolvimento Baseado em Componentes, Gimenes, I.M.S. & Huzita, E.H.M. (eds), Editora Ciência Moderna, Rio de Janeiro, 2005. ISBN 85-7393-406-9, pg. 57-103.

Barros, L.A. (1994) Suporte a Ambientes Distribuídos para Aprendizagem Cooperativa, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro.

Bass, L., Clements, P. & Kazman, R. (2003) Software Architecture in Practice, Addison-Wesley 2003. ISBN: 0-321-15495-9

Beck, K. (2004): Programação extrema explicada: acolha as mudanças. Porto Alegre: Bookman.

Becker, K. & Zanella, A.N. (1998) "A Cooperation Model for Teaching/Learning Modeling Disciplines", International Workshop on Groupware (CRIWG 1998), Brasil.

Begole, J.B., Rosson, M.B. & Shaffer, C.A. (1999). Flexible collaboration transparency: Supporting worker independence in replicated application-sharing systems. ACM Transactions on Computer–Human Interaction, 6(2), 95–132.

Benbunan-Fich, R. & Hiltz, S. R. (1999) Impacts of Asynchronous Learning Networks on Individual and Group Problem Solving: A Field Experiment, Group Decision and Negotiation, Vol.8, pp. 409-426.

Blikstein, I. (2000) Técnicas de comunicação escrita. Coleção Princípios, Ed. Ática, ISBN 8508094884.

Blois, A.P.T.B. & Becker, K.A. (2002) "Component-based Architecture to Support Collaborative Application Design", 8th International Workshop on Groupware (CRIWG), LNCS Vol. 2440, Springer-Verlag, p. 134-146

Blois, A.P.T.B., Werner, C.M.L. & Becker, K. (2004) "Um Processo de Engenharia de Domínio com foco no Projeto Arquitetural Baseado em Componentes", IV Workshop de Desenvolvimento Baseado em Componentes, João Pessoa, PB, pp. 15-20.

Blois, M., Choren, R., Laufer, C., Ferraz, F. & Fuks, H. (1999) "Desenvolvendo Aplicativos para a Web com o Scriba", Anais do XXVI SEMISH - Seminário Integrado de Software e Hardware, Sociedade Brasileira de Computação (SBC), Rio de Janeiro, pp 119-133.

Bloom, B.S. (1956) "Taxonomy of educational objectives: handbook 1, cognitive domain", New York, Longman.

Boehm, B.W. (1988) A Spiral Model of Software Development and Enhancement, IEEE Computer, Vol. 21, No. 5, pp. 61-72

Booch, G. (1987) Software Components with Ada: Structures, Tools, and Subsystems. Benjamin-Cummings, Redwood City, CA.

Booch, G., Rumbaugh, J. & Jacobson, I. (2000) UML: Guia do Usuário. Editoria Campus, Rio de Janeiro. ISBN 85-352-0562-4

Borges, M.R. & Pino, J.A. (1999) "Awareness mechanisms for coordination in asynchronous CSCW," Procs. of the 9th Workshop on Information Technologies and Systems (WITS'99), 1999, pp. 69-74.

Borges, M.R.S., Brézillon, P., Pino, J.A. & Pornerol, J.C. (2004) "Bringing Context to CSCW", Proceedings of The 8th International Conference on Computer Supported Cooperative Work in Design (CSCWiD 2004), China, IEEE, pp. 161-166.

Borges, M.R.S., Mendes, S. & Motta, C.L.R. (2002) "Improving Meetings by identifying informal roles played by participants", 7th International Conference on Computer Supported Cooperative Work in Design, 2002, Rio de Janeiro, pp. 368-372.

Borges, M.R.S., Pino, J.A. & Salgado, A.C. (2000) Requirements for Shared Memory in CSCW Applications, Proceedings of the 10th Workshop on Information Technology and Systems (WITS'00), Brisbane, Australia.

Borghoff, U.M. & Schlichter, J.H. (2000) Computer-Supported Cooperative Work: Introduction to Distributed Applications. Springer, USA.

Braga, R. (2000) "Busca e Recuperação de Componentes em Ambientes de Reutilização de Software". Tese de Doutorado. COPPE Sistemas, 2000.

Bretain, I., Fredin, L., Frost, W., Hedman, L.R., Kroon, P., McGlashan, S., Sallnas, E.L. & Virtanen, M. (1997) Leave the Office, Bring Your Colleagues: Design Solutions for Mobile Teamworkers. Proc. CHI'97, ACM Press, pp.335-336

Brna, P. (1998) "Modelos de colaboração", Revista Brasileira de Informática e Educação, 3, pp. 1-15.

Brockschmidt, K. (1996) What OLE Is Really About, MSDN Archive, Microsoft Comporation. msdn.microsoft.com/archive/en-us/dnarolegen/html/msdn_aboutole.asp

Brown, A.W. (1996) Component-Based Software Engineering. IEEE Computer Society.

Buzko, D., Lee, W. & Helal, A. (2000) Decentralized Ad-Hoc Groupware API and Framework for Mobile Collaboration. In Proceedings of Group 2001, Collaborative Editing Workshop.

Calvary, G., Coutaz, J. & Nigay, L. (1997) From Single-User Architectural Design to PAC*: a Generic Software Architectural Model for CSCW. Conference on Human Factors in Computing Systems (CHI'97), pp 242-249.

Castellani, S., Ciancarini, P. & Rossi, D. (1996) "The ShaPE of ShaDE: a Coordination System", Technical Report UBLCS 96-5, Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy, March, 1996.

Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S. & Seguin, C. (1998) Java Object-Sharing in Habanero. Communications of the ACM, Vol. 41 # 6, June 1998.

Cheesman, J. & Daniels, J. (2000) UML Components: A SimpleProcess for Specifying Component-Based Software, Addison-Wesley, Reading, Massachusetts.

Chung, G. & Dewan, P. (2004) "Towards Dynamic Collaboration Architectures", Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'04), November 6-10, Volume 6, Issue 3.

Churcher, N. & Cerecke, C. (1996) "GroupCRC: Exploring CSCW support for software engineering", Proceedings of the 6th Australian conference on computer-human interaction, Hamilton, New Zealand, November 24-27, 1996. IEEE Computer Society Press, Los Alamitos, CA, USA, 1996, pp. 62-68.

Clements, P. & Northrop, L. (2001) Software Product Lines – Practices and Patterns. Readding, Addison-Wesley, 2001.

Conklin, J. & Begeman, M. (1988) "gIBIS: A hypertext tool for exploratory policy discussion", ACM Transactions on Office Information Systems, Vol. 3, No. 3

Councill, B. & Heineman, G.T. (2001) Definition of a Software Component and Its Elements, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Cousineau, G. & Mauny, M. (1998) The Functional Approach to Programming, Cambridge University Press, English edition, December 1998. ISBN: 0521576814

Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2003) "A Adaptação do Ambiente AulaNet para Dar Suporte a Grupos de Aprendizagem e sua Formação Utilizando os Conceitos de Agentes de Software", Revista Brasileira de Informática na Educação, ISSN 1414-5685, Sociedade Brasileira de Computação, V. 11, No. 1, Abril de 2003, pp. 26-46.

D'Souza, D.F. & Wills, A.C. (1998) Objects, Components and Frameworks with UML: The Catalysis Approach. Addison Wesley, ISBN 0-201-31012-0, 1998.

Daft, R.L. & Lengel, R.H. (1986). Organizational information requirements, media richness and structural design. Management Science 32(5), 554-571.

David, J.M.N. & Borges, M.R.S. (2001) "Selectivity of awareness components in asynchronous CSCW environments", Proceedings of 7th International Workshop on Groupware (CRIWG 2001), IEEE, Darmstadt, Germany.

David, J.M.N. & Borges, M.R.S. (2004) "Designing collaboration through a web-based groupware infrastructure", International Journal of Computer Applications in Technology, v.19, n.3/4, pp. 175-183.

Dewan, P. (1998) Architectures for Collaborative Applications. In M.Beaudouin-Lafon (Ed.), Computer Supported Cooperative Work (CSCW) (7 ed., pp. 169-194). John Wiley & Sons Ltd.

Dias, M.S. & Borges, M.R.S. (1999) "Development of groupware systems with the COPSE infrastructure", International Workshop on Groupware (CRIWG 1999), Cancun, IEEE Computer Society, pp. 278-285.

Dillenbourg, P. & Self,. J.A. (1992) "A computational approach to socially distributed cognition", European Journal of Psychology of Education, Vol VII, No 4, pp 252-273, 1992.

Dillenbourg, P. (1999) “What do you mean by collaborative learning?”, Collaborative-learning: Cognitive and Computational Approaches, Oxford, Elsevier, 1999, pp. 1-19

Dourish, P. & Belloti, V. (1992) “Awareness and coordination in shared workspaces,” Proceedings of CSCW’92, Chapel Hill NC, 1992.

Dourish, P. (1998) Using Metalevel techniques in a flexible toolkit for CSCW applications, ACM Transactions on Computer-Human Interaction, Vol 5, No 2, pp. 109-155.

Dron J., Boyne C. & Mitchell R. (2001) “Footpaths in the stuff swamp” Proceedings of WebNet'2001, Fowler, W. & Hasebrook, J. (eds.), WorldConference of the WWW and Internet, Orlando, FL, AACE, pp. 323-328.

Durfee, E.H. (1988) Coordination of Distributed Problem Solvers, The International Series in Engineering and Computer Science, Vol. 55, ISBN: 0-89838-284-X.

Edutools (2005) http://www.edutools.info (date 06/04/2005)

Ellis, C.A. & Wainer, J. (1994) A Conceptual Model of Groupware, In T. Malone (ed) Conference on Computer-Supported Cooperative Work (CSCW), pp. 79-88.

Ellis, C.A., Gibbs, S.J. & Rein, G.L. (1991) Groupware - Some Issues and Experiences. Communications of the ACM, Vol. 34, No. 1, pp. 38-58.

Engelbart, D. & English, W. (1968) Research Center for Augmenting Human Intellect, Proc. Fall Joint Computing Conference, AFIPS Press, 395-410

Farias, C.R.G. (2004) Um Metamodelo para Sistemas Cooperativos. In: Workshop Brasileiro de Tecnologias para Colaboração, Ribeirão Preto. Proceedings of WebMedia & LA-Web 2004 Joint Conference, 2004. v. 2. pp. 194-201.

Fayad, M. E. & Johnson, R. E. (2000): Domain-specific application frameworks: frameworks experience by industry. New York: J. Wiley, c2000. 681 p. ISBN 0471332801.

Fayad, M. E. & Schmidt, D. C. (1997): Object-oriented application frameworks, Communications of the ACM, v.40 n.10, p.32-38, Oct. 1997

Fayad, M. E., Schimidt & D. C., Johnson, R. E. (1999a): Implementing application frameworks: object-oriented frameworks at work. New York: J. Wiley, c1999. 729 p. ISBN 0471252018.

Fayad, M. E., Schimidt & D. C., Johnson, R. E. (1999b): Building application frameworks: object-oriented foundations of framework design. New York: J. Wiley, c1999. 664 p. ISBN 0471248754.

Fielding, R.T. (1999) Shared leadership in the Apache project, Communications of the ACM, Volume 42 , Issue 4, pp 42-43.

Filippo, D., Fuks, H. & Lucena, C.J.P. (2005) AulaNetM: Extensão do Serviço de Conferências do AulaNet destinada a usuários de PDAs. Anais do XVI Simpósio Brasileiro de Informática na Educação - SBIE 2005, Juiz de Fora, MG, 9 a 11 de novembro de 2005, pp. 623-633

Fitzpatrick, G., Kaplan, S., Mansfield, T. Arnold, D. & Segall, B. (2002) Supporting Public and Accessibility with Elvin: Experiences and Reflections. Computer Supported Cooperative Work, Vol. 11, No. 3-4, pp. 447-474.

Fitzpatrick, G., Tolone, W.J. & Kaplan, S. M. (1995) Work, Locales and Distributed Social Worlds. In Proceedings of the 4th European Conference on Computer Supported Cooperative Work (ECSCW '95), pp. 1-16, 1995.

Fowler, M. (1996) Analysis Patterns: Reusable Object Models, Addison-Wesley.

Fowler, M. (2002) Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.

Fuggetta, A. (2000) Software Process: A Roadmap, 22nd International Conference on Software Engineering – ICSE 2000, Future of Software Engineering Track.

Fuks, H. (2000) "Aprendizagem e Trabalho Cooperativo no Ambiente AulaNet", Revista Brasileira de Informática na Educação, N6, Abril 2000, ISSN 1414-5685, Sociedade Brasileira de Computação, pp 53-73, 2000.

Fuks, H., Cunha, L.M., Gerosa, M.A. & Lucena, C.J.P. (2003) "Participação e Avaliação no Ambiente Virtual AulaNet da PUC-Rio". in: Silva, M.; Educação Online: Teorias, Práticas, Legislação e Formação Corporativa; Edições Loyola, Rio de Janeiro, 2003, ISBN 85-15-02822-0, Cap. 15, pp. 231-254.

Fuks, H., Gerosa, M.A. & Lucena, C.J.P. (2002) "The Development and Application of Distance Learning on the Internet". Open Learning - The Journal of Open and Distance Learning, Vol. 17, N. 1, ISSN 0268-0513, pp 23-38.

Fuks, H., Gerosa, M.A. & Pimentel, M. (2003) "Projeto de Comunicação em Groupware: Desenvolvimento, Interface e Utilização". XXII Jornada de Atualização em Informática, Anais do XXIII Congresso da Sociedade Brasileira de Computação, V2, Cap. 7, ISBN 85-88442-59-0, pp. 295-338.

Fuks, H., Raposo, A.B. & Gerosa, M.A. (2002) "Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas", XXI Jornada de Atualização em Informática, Anais do XXII Congresso da Sociedade Brasileira de Computação, V2, Cap. 3, ISBN 85-88442-24-8, pp. 89-128.

Fuks, H., Raposo, A.B., Gerosa, M.A. & Lucena, C.J.P. (2005) Applying the 3C Model to Groupware Development. International Journal of Cooperative Information Systems (IJCIS), v.14, n.2-3, Jun-Sep 2005, World Scientific, ISSN 0218-8430, pp. 299-328.

Funaro, G.M. & Montell, F. (1999) "Pedagogical roles and implementation guidelines for online communication tools", ALN Magazine Volume 3, Issue 2, December 1999

Fussell, S.R., Kraut, R. E., Learch, F.J., Scherlis, W.L., McNally, M.M. & Cadiz, J.J. (1998) "Coordination, overload and team performance: effects of team communication strategies", Proceedings of CSCW '98, Seattle, USA, p. 275-284. ISBN 1-58113-009-0

Gaines, B. (1999) "Modeling and forecasting the information sciences", *Information Sciences* 57/58, pp. 13-22.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1994) Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, ISBN 0201633612.

Gandon, F. & Sadeh, N. (2004) Semantic Web Technology to Reconcile Privacy and Context Awareness, Web Semantics Journal. Vol. 1, No. 3, 2004.

Gerosa, M.A. (2002) "Categorização e Estruturação de Mensagens Textuais em Ambientes Virtuais de Colaboração", Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 28 de fevereiro de 2002.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2001) "Um groupware baseado no ambiente AulaNet desenvolvido com componentes", Anais eletrônicos do 1º Workshop de Desenvolvimento Baseado em Componentes, 21-22 Junho, Maringá-PR.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2001) "Use of Categorization and Structuring of Messages in order to Organize the Discussion and Reduce Information Overload in Asynchronous Textual Communication Tools". 7th International Workshop on Groupware (CRIWG 2001), M. Borges, J. Haake and U. Hoppe (eds.), IEEE, 6-8 September, Darmstadt - Germany, pp 136-141

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2002) "Resultados da avaliação de um curso baseado na Web", VIII Workshop de Informática na Escola (WIE 2002), XXII Congresso da Sociedade Brasileira de Computação, V. 5, 17 a 19 de julho, Florianópolis, pp 477-485.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2003) Analysis and Design of Awareness Elements in Collaborative Digital Environments: A Case Study in the AulaNet Learning Environment. The Journal of Interactive Learning Research, 14(3), ISSN: 1093-023X, Association for the Advancement of Computing in Education, USA, pp. 315-332.

Gerosa, M.A., Pimentel, M., Fuks, H. & Lucena, C.J.P. (2005) "No Need to Read Messages Right Now: Helping Mediators to Steer Educational Forums Using Statistical and Visual Information". Computer Supported Collaborative Learning, Taiwan, July 2005, ISBN 0805857826, Lawrence Erlbaum Associates, pp. 160-169.

Gimenes, I.M.S. & Huzita, E.H.M. (2005) Desenvolvimento Baseado em Componentes, Editora Ciência Moderna, Rio de Janeiro, 2005. ISBN 85-7393-406-9.

Goldratt, E.M. (1997) "Critical Chain", The North River Press Publishing Corporation, Great Barrington.

Govoni, D. (1999) Java Application Frameworks, Wiley & Sons, ISBN 0-471-32930-4.

Graham, M., Scarborough, H. & Goodwin, C. (1999) Implementing Computer Mediated Communication in an Undergraduate Course - A Practical Experience. Journal of Asynchronous Learning Networks, Vol. 3, No. 1, 1999, pp. 32-45.

Greenberg, S. & Fitchett, C. (2001) Phidgets: Easy Development of Physical Interfaces through Physical Widgets. Proceedings of the UIST 2001 - 14th Annual ACM Symposium on User Interface Software and Technology, November 11-14, Orlando, Florida, ACM Press, pp. 209-218.

Greenberg, S. (2003) Enhancing Creativity with Groupware Toolkits. Invited keynote talk. Proceedings of the 9th International Workshop on Groupware (CRIWG 2003), Sept 28 - Oct 2, Autrans, France, LNCS vol. 2806, Springer-Verlag, pp. 1-9.

Greenberg, S. (2006) “Toolkits and Interface Creativity”, Journal of Multimedia Tools and Applications, Special Issue on Groupware, Kluwer. In Press. Disponível em http://grouplab.cpsc.ucalgary.ca/papers (consulta 15 de janeiro de 2006)

Greif, I. (1988) Computer Supported Cooperative Work - A book of readings. Morgan Kaufmann Publishers, USA, 1988. ISBN 0-934613-57-5.

Griss, M.L. (2001) Product-Line Architectures, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Gross, T. (1997) “Towards flexible support for cooperation: group awareness in shared workspaces,” DEXA’97, França, IEEE, Los Alamitos, CA, pp. 406-411.

Grosz, B.J. (1996) Collaborative systems, AI Magazine 17 (2), pp. 67–85

Groupware Patterns Swiki (2005) http://www.groupware-patterns.org

Grudin, J. (1989) Why Groupware Applications Fail: Problems In Design And Evaluation. Office: Technology and People, Vol. 4, No. 3, pp. 245-264.

Grundy, J.C., Mugridge, W.B. & Hosking, J.G. (1997) “A Java-based Componentware Toolkit for constructing Multi-view Editing Systems”, Proceedings of the 2nd Component Users' Conference (CUC'97), Munich July 15-18.

Guerrero, L. & Fuller, D. (1999) “A Web-Based OO Platform for the Development of Multimedia Collaborative Applications”, Decision Support Systems, v.27, n.3, pp.255-268.

Guicking, A., Tandler, P. & Avgeriou, P. (2005) “Agilo: A Highly Flexible Groupware Framework”, Proceeding of the 11th International Workshop on Groupware, CRIWG 2005, Porto de Galinhas-PE, September 2005, Fuks, H., Lukosch, S. & Salgado, A.C. (eds), Lecture Notes in Computer Science, Vol 3706, pp. 49-56.

Gutwin, C. & Greenberg, S. (2000) The Mechanics of Collaboration: Developing Low Cost Usability Evaluation Methods for Shared Workspaces. IEEE 9th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises -WETICE (2000), p. 98-103.

Gutwin, C. & Greenberg, S. (2002) A Descriptive Framework of Workspace Awareness for Real-Time Groupware, Computer Supported Cooperative Work, Vol. 11, No. 3-4, pp. 411-446.

Gutwin, C., Stark, G. & Greenberg, S. (1995) "Support for workspace awareness in educational groupware". Computer Support for Collaborative Learning, Lawrence Erlbaum Associates, New York, 1995, pp. 147-156.

Hansen, R.P., Pinto, S.C.C.S. & Hansen, C.R. (2005) “Integrando Web Services e Recursos Educacionais Através de Composição” XATA2005 - XML: Aplicações e Tecnologias Associadas, 2005, Braga, Portugal, pp. 290-301.

Harasim, L., Hiltz, S. R., Teles, L., & Turoff, M. (1997) “Learning networks: A field guide to teaching and online learning”, 3rd ed., MIT Press, 1997.

Heineman, G.T. (2000) “A model for designing adaptable software components”, ACM SIGSOFT Software Engineering Notes archive, Volume 25, Issue 1, January 2000, ISSN 0163-5948, pg 55-56

Hess, H., Cohen, S., Holibaugh, B., Kyang, K., Peterson, A.S., Novak, W. & Carroll, P. (2000) A Domain Analysis Bibliography, Carnegie Mellon University/Software Engineering Institute, Special Report 90-SR-3.

Houaiss (2001) Dicionário Eletrônico da Língua Portuguesa, versão 1.0, Editora Objetiva.

Houston, K. & Norris, D. (2001) Software Componentes and the UML, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Hummes, J. & Merialdo, B. (2000) Design of Extensible Component-Based Groupware. Computer Supported Cooperative Work, 9(1), 53-74. ISSN 0925-9724.

Hwang, M.I. & Lin, J.W. (1999) Information dimension, information overload and decision quality. Journal of Information Science. Vol. 25, no. 3, pp. 213-218.

Ierusalimschy, R., Figueiredo, L.H. & Celes, W (1996) "Lua – an extensible extension language. Software: Practice & Experience", 26(6), 635-652.

Isenhour, P.L., Begole, J., Heagy, W.S. & Shaf-fer, C.A. (1997) "Sieve: A java-based collaborative visualization environment", Late Breaking Hot Topics Proceedings, IEEE Visualization 97, 1997

Jacobson, I., Griss, M. & Jonsson, P. (1997) Software Reuse: Architecture, Process and Organization for Business Success, Addison-Wesley, Nova York.

Johnson, R. (1997) "Components, Frameworks, Patterns", Symposium of Software Reusability 1997 USA.

Johnson, R. (2002): Expert One-on-One J2EE Design and Development. Reading: Wiley Publishing Inc., 2002.

Johnson, R. (2004): Expert One-on-One J2EE Development without EJB. Wiley Publishing Inc., 2004.

Kang, K., Cohen, S., Hess, J., Novak, W. & Peterson, A. (1990) "Feature-Oriented Domain Analysis (FODA) Feasibility Study" (CMU/SEI-90-TR-21). Pittsburg, PA, Software Engineering Institute, Carnegie Mellon University.

Kang, K., Kim, S., Lee, J., Kim, K., Shin E. & Huh, M. (1998) "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, Annals of Software Engineering, 5, pp. 143-168

Kanselaar, G., Erkens, G., Andriessen, J., Prangsma, M., Veerman, A. & Jaspers, J. (2003) Designing Argumentation Tools for Collaborative Learning, Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making, Kirschner, P., Shum, S. & Carr, C. (eds.), chap. 3, Springer-Verlag.

Kirschner, P.A., Shum, S.J.B. & Carr, C.S. (2003) Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making, Springer.

Kirsch-Pinheiro, M., Lima, J.V. & Borges, M.R.S. (2002) A Framework for Awareness Support in Groupware Systems. Proceedings of the 7th International Conference on Computer Supported Cooperative Work in Design – CSCWD'2002, Rio de Janeiro, Brazil.

Koch, M. & Koch, J. (2000) "Application of Frameworks in Groupware – The Iris Group Editor Environment", ACM Computing Surveys Symposium on Frameworks.

Kolfschoten, G.L., Briggs, R.O., Appelman, J.H. & de Vreede, G.J (2004) "ThinkLets as Building Blocks for Collaboration Processes: A Further Conceptualization Groupware", Proceeding of the 10th International Workshop on Groupware, CRIWG 2004, San Carlos, Costa Rica, September 5-9, Lecture Notes in Computer Science, Vol 3198, pp. 137-152.

Kraut, R.E. & Attewell, P. (1997) "Media use in global corporation: electronic mail and organizational knowledge," Research milestone on the information highway, Mahwah, NJ: Erlbaum.

Krebs, A.M., Ionescu, M., Dorohomceanu, B. & Marsic, I. (2003) "The DISCIPLE System for Collaboration over the Heterogeneous Web" Proceedings of the Hawaii International Conference on System Sciences – HICSS 2003.

Kreifelts, T., Hinrichs, E. & Woetzel, G. (1993) Sharing To-Do Lists with a Distributed Task Manager. In Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93), pp. 31-46, 1993.

Krueger, C.W. (1992) Software Reuse, ACM Computing Surveys, Volume 4 Issue 2 p131-183.

Kulesza, U., Garcia, A., Lucena, C.J.P. & Staa, A.V. (2004) "Integrating Generative and Aspect-Oriented Technologies", Simpósio Brasileiro de Engenharia de Software 2004, Brasília, pp. 130-146.

Kuutti, K. (1991) "The concept of activity as a basic unit of analysis for CSCW research", Proceedings of the Second European Conference on Computer Supported Cooperative Work (ECSCW'91), pp. 249-264.

Lajoie, R. & Keller, R.K. (1995) "Design and reuse in object-oriented frameworks: Patterns, contracts, and motifs in concert", Object-Oriented Technology for Database and Software Systems, Alagar, V. & Missaoui, R. (eds), Singapore, 1995, World Scientific, pp. 295-312.

Lange, B.M. & Gershman, A. (1992) "OMNI: A Corporate Knowledge Environment for Collaborative Work", IEEE Conference on Systems, Man and Cybernetics, 1992.

Larman, C. (2004) Utilizando UML e Padrões: Uma introdução à análise e ao projeto orientados a objetos e ao Processo Unificado, 2ª edição, Bookman, Porto Alegre.

Laufer, C. & Fuks, H. (1995) "ACCORD: Conversation Clichés for Cooperation", Proceedings of The International Workshop on the Design of Cooperative Systems, Juan-les-Pins, França, pp 351-369.

Laurillau, Y. & Nigay, L. (2002) "Clover architecture for groupware", Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW 2002), pp. 236 - 245

Lee, D., Lim, M. & Han, S. (2002) "ATLAS: a scalable network framework for distributed virtual environments", Proceedings of the 4th International Conference on Collaborative Virtual Environments.

Li, D. & Muntz, R.R. (1998) "COCA: Collaborative Objects Coordination Architecture", Proceedings of the ACM Conference on Computer Supported Cooperative Work 1998, pp. 179-188.

Litiu, R. & Prakash, A. (2000) "Developing Adaptive Groupware Applications Using a Mobile Computing Framework", Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'00), pp. 107-116.

Liu, Y., Shi, Y. & Xu, G. (2001) "Supporting Group Awareness in Collaborative Design", Proceeding of Computer Supported Cooperative Work in Design 2001, London, Jul 2001.

Long, B. & Baecker, R. (1997) "A taxonomy of Internet communication tools", Proceedings of WebNet - World Conference of the WWW, Internet, and Intranet, Toronto, Canada, ISBN 1-880094-27-4, pp. 318-323.

Lucena, C.J.P. & Fuks, H. (2000) Professores e Aprendizes na Web: A Educação na Era da Internet, ISBN 85-88011-01-8, Editora Clube do Futuro, Rio de Janeiro, 2000.

Lucena, C.J.P., Fuks, H., Milidiu, R., Macedo, L., Santos, N., Laufer, C., Blois, M., Fontoura, M.F., Choren, R., Pinto, S.C.C.S., Torres, V., Daflon, L. & Lukowiecki, L. (1998) AulaNet - An Environment for the Development and Maintenance of Courses on the Web. Proceedings of ICEE'98 - International Conference On Engineering Education, Rio de Janeiro, 1998

Lukosch, S. & Schümmer, T. (2004) Patterns for Managing Shared Objects in Groupware Systems, Proceedings of the 9th European Conference on Pattern Languages and Programs, Irsee, Germany.

Mackenzie, J. (1985) No Logic before Friday, Synthese, V.63, pp 329-341.

Magnusson, M. & Svensson, L. (2000) Studying how students study: Work-orientation and collaboration in Distance Education, Proceedings of IRIS 23, Svensson et al (eds.), University of Trollhättan Uddevalla. Sweden

Malone, T.W. & Crowston, K. (1990) What is Coordination Theory and How Can It Help Design Cooperative Work Systems? Conference on Computer-Supported Cooperative Work (CSCW), pp. 357-370.

Mandviwalla, M. & Olfman, L. (1994) What do groups need? A proposed set of generic requirements. ACM Transactions on Computer-Human Interaction, Vol. 1, No. 3, pp. 245-268.

Markus, M.L. & Connolly, T. (1990) Why CSCW applications fail: Problems in the adoption of interdependent work tools. Proceedings of CSCW'90 (Los Angeles, Oct. 7-10).

Marsic, I. & Dorohonceanu, B. (2003) "Flexible User Interfaces for Group Collaboration". International Journal of Human-Computer Interaction, Vol.15, No.3, pp. 337-360

Marsic, I. (1999) DISCIPLE: a framework for multimodal collaboration in heterogeneous environments. ACM Computing Surveys, 31 (2es), Article No. 4.

Mattsson, M. (2000) Evolution and Composition of Object-Oriented Frameworks, PhD Thesis, Department of Software Engineering and Computer Science, University of Karlskrona/Ronneby.

McIlroy, M.D. (1968) "Mass Produced Software Components", Software Engineering, NATO Science Committee, pp. 138-150.

Michailidis, A. & Rada, R. (1996) “A review of collaborative authoring tools”, Groupware and authoring. Academic Press, London, 1996, p. 9-43.

Mitchell, L.H.R.G., Fuks, H. & Lucena, C.J.P. (2004) Contribuições da Gestão de Competências para a Educação a Distância: Experimento com o Ambiente AulaNet. Informática na Educação: Teoria e Prática, Vol 7, No. 2, Porto Alegre, UFRGS, ISSN 1516-084X, pp. 83-98.

Moore, J. M. & Bailin, S.C. (1991) “Domain Analysis: Framework For Reuse”, Prieto-Diaz, R. & Arango, G. (eds.), Domain Analysis and Software System Modeling. LosAlamitos, CA: IEEE Computer Society Press, pp. 179-203.

Morch, A.I. (1997) “Three Levels of End-user Tailoring: Customization, Integration, and Extension” In: Computers and Design in Context, Edited by M. Kyng and L. Mathiassen, MIT Press, USA.

Motta, C.L.R. & Borges, M.R.S. (2000) “A Cooperative Approach for Information Recommendation and Filtering”, Proceedings of the International Workshop on Groupware, IEEE Computer Society, Madeira, Portugal, October 2000, pp. 42-49.

Muhammad, A., Enríquez, A.M.M. & Decouchant, D. (2005) “Awareness and Coordination for Web Cooperative Authoring”, Advances in Web Intelligence: Third International Atlantic Web Intelligence Conference, AWIC 2005, Lodz, Poland, June 6-9, 2005. ISBN: 3-540-26219-9.

Myers, B. (1995) “State of the Art in User Interface Software Tools”, In Baecker, R., Grudin, J., Buxton, W. & Greenberg, S. (eds), *Readings in Human Computer Interaction: Towards the Year 2000*. Morgan Kaufmann , pp. 323-343.

Neale, D.C., Carroll, J.M. & Rosson, M.B. (2004) “Evaluating computer-supported cooperative work: models and frameworks”, Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW '04), Chicago, Illinois, USA, November 06 - 10, ACM Press, New York, pp. 112-121.

Neisser, U. (1976) Cognition and Reality, Ed. W.H. Freeman, San Francisco.

Nielsen, J. (1994) “Heuristic Evaluation”, Usability Inspection Methods, Chapter 2, Nielsen, J. & Mack, R. (eds), John Wiley and Sons, New York, pp. 25-62.

Nunamaker, J.F., Romano, N.C. & Briggs, R.O. (2001) A Framework for Collaboration and Knowledge Management. In: Proceedings of 34th Hawaii International Conference on System Sciences - HICSS'01.

Oliveira, R.F., Blois, A.P.T.B., Vasconcelos, A.P.V. & Werner, C.M.L. (2005) “Representação de Variabilidades em Componentes de Negócio no Contexto da Engenharia de Domínio”, V Workshop em Desenvolvimento Baseado em Componentes, Juiz de Fora, MG, novembro 2005, pp. 73-80.

Oliveira, T.C. (2001) Uma Abordagem Sistemática para a Instanciação de Frameworks Orientados a Objetos, Tese de Doutorado, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, 2001.

OMG (2005) Unified Modeling Language Superstructure Specification, version 2.0, formal/05-07-04, Agosto de 2005.

Orfali, R., Harkey, D. & Edwards, J. (1996) The Essential Distributed Objects Survival Guide, John Wiley & Sons, New York.

Ostwald, J. (1995) "Supporting collaborative design with representations for mutual understanding", CHI'95 Proceedings, Doctoral Consortium.

Osuna, C.A. & Dimitriadis, Y.A. (1999) "A Framework for Development of Educacional-Collaborative Applications Based on Social Constructivism". Proceedings of International Workshop on Groupware - CRIWG'99, IEEE Press, Cancun, Mexico.

Paludo, M.A. & Burnett, R.C. (2005) "Desenvolvimento Baseado em Componentes e Padrões", in: Desenvolvimento Baseado em Componentes, Gimenes, I.M.S. & Huzita, E.H.M. (eds), Editora Ciência Moderna, Rio de Janeiro, 2005. ISBN 85-7393-406-9, pg. 199-231.

Peterson, A.S. (1991) Coming to Terms with Software Reuse Terminology: a Model-based Approach, ACM SIGSOFT Software Engineering Notes, Volume 16, Issue 2, April 1991, pp. 45-51, ISSN:0163-5948

Pfleeger, S.L. (2001) Software engineering: theory and practice, Upper Saddle River, NJ: Prentice Hall.

Philippe, K. (2003): Introdução ao RUP – Rational Unified Process. Rio de Janeiro: Ciência Moderna.

Pimentel, M. (2006) RUP-3C-Groupware: um processo de desenvolvimento de groupware baseado no Modelo 3C de Colaboração, Tese de Doutorado, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, 2006.

Pimentel, M., Fuks, H. & Lucena, C.J.P. (2003a) Debati, debati... aprendi? Investigações sobre o papel educacional das ferramentas de bate-papo. WIE 2003 - IX Workshop de Informática na Escola, Anais do XXIII Congresso da Sociedade Brasileira de Computação, V5, Campinas-SP, 2 a 8 de agosto de 2003. pp. 167-178.

Pimentel, M., Fuks, H. & Lucena, C.J.P. (2003b) Co-text Loss in Textual Chat Tools. 4th International and Interdisciplinary Conference on Modeling and Using Context - CONTEXT 2003, LNAI 2680, Stanford, CA, USA, June, pp 483-490, 2003.

Pimentel, M., Fuks, H. & Lucena, C.J.P. (2004) "Avaliação da Participação em Conferências Textuais Assíncronas", Anais Eletrônico do X Workshop de Informática na Escola, integrante do XXIV Congresso da Sociedade Brasileira de Computação (WIE/SBC 2004), ISBN: 85-88442-94-9, 31 Julho - 6 Agosto, Salvador, BA.

Pimentel, M., Fuks, H. & Lucena, C.J.P. (2005) "Mediated Chat Development Process: Avoiding Chat Confusion on Educational Debates", Computer Supported Collaborative Learning, Taiwan, July 2005, ISBN 0805857826, Lawrence Erlbaum Associates, pp. 499-503.

Pinto, S.C.C.S. (2000) Composição em WebFrameworks, tese de doutorado, Departamento de Informática PUC-Rio.

POJO (2005): Trecho do blog do Martin Fowler onde o termo Plain Old Java Object é explicado disponível em http://www.martinfowler.com/bliki/POJO.html Última visita em 20/10/2005.

Prakash, A. & Knister, M.J. (1994) "A framework for undoing actions in collaborative systems", ACM Transactions on Computer-Human Interaction, 1(4)295–330, December 1994.

Pree, W. (1995) Design Patterns for Object-Oriented Software Development, Addison-Wesley Publishing Company, 1995.

Pressman, R.S. (2000) Software Engineering : A Practitioner's Approach, McGraw Hill, New York, NY, June 2000.

Pumareja, D., Sikkel, K. & Wieringa, R. (2004) "Understanding the dynamics of requirements evolution: a comparative case study of groupware implementation", Proceedings of the 10th Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2004), Essener Informatik Beiträge 9, pp. 177-194.

Raposo, A.B. & Fuks, H. (2002) Defining Task Interdependencies and Coordination Mechanisms For Collaborative Systems. In M. Blay-Fornarino, A.M. Pinna-Dery, K. Schmidt and P. Zaraté (eds) Cooperative Systems Design (Frontiers In Artificial Intelligence and Applications Vol. 74). IOS Press, Amsterdam, pp. 88-103.

Raposo, A.B., Gerosa, M.A. & Fuks, H. (2004) "Combining Communication and Coordination toward Articulation of Collaborative Activities", 10th International Workshop on Groupware - CRIWG 2004. Vreede, G.J., Guerrero,L.A., Raventós,G.M. (eds.). Lecture Notes on Computer Science LNCS Vol. 3198, ISBN 3540-230165, ISSN 0302-9743. San Carlos, Costa Rica: Springer-Verlag, 5-9 September 2004. pp. 121-136.

Rezende, J.L., Fuks, H. & Lucena, C.J.P. (2003) Aplicando o Protocolo Social através de Mecanismos de Coordenação embutidos em uma Ferramenta de Bate-Papo. XIV Simpósio Brasileiro de Informática na Educação - SBIE 2003, 12 a 14 de Novembro de 2003, ISBN: 85-88442-70-1, Rio de Janeiro - RJ, pp. 55-64.

Romano, D., Brna, P. & Self, J. (1998) "Collaborative decision-making and presence in shared dynamic virtual environments", Proceedings of the Workshop on Presence in Shared Virtual Environments. BT Labs, Martlesham Heath.

Roschelle, J. & Teasley, S. (1995) The construction of shared knowledge in collaborative problem solving. In O'Malley, C.E., (ed.), Computer Supported Collaborative learning. Springer-Verlag, Heildelberg, pp 69-97.

Roseman, M. & Greenberg, S. (1996) "Building real time groupware with GroupKit, a groupware toolkit". ACM Transactions on Computer-Human Interaction, 3, March 1996, 1, p. 66-106.

Roth, J. & Unger, C. (2000) Developing synchronous collaborative applications with TeamComponents. In Designing Cooperative Systems: the Use of Theories and Models, Proceedings of the 5th International Conference on the Design of Cooperative Systems (COOP'00), pp. 353-368.

Roussos, M., Johnson, A.E., Leigh, J., Bames, C.R., Vasilakis, C.A. & Moher, T.G. (1997) "The NICE Project: Narrative, Immersive, Constructionist/Collaborative Environments for Learning in Virtual Reality". Proceedings of Educational Multimedia and Telecommunications- ED-Media, 1997.

Rubart, J. & Dawabi, P. (2002) Towards UML-G: A UML-Profile for Modeling Groupware, 8th International Workshop on Groupware, CRIWG 2002, LNCS 2440 , Springer-Verlag, La Serena, Chile, 2002, pp. 93-113.

Sacramento, V., Endler, M., Rubinsztejn, H.K., Lima, L.S., Gonçalves, K., Nascimento, F.N. & Bueno, G.A. (2004) "MoCA: A Middleware for Developing Collaborative Applications for Mobile Users," IEEE Distributed Systems Online, vol. 5, no. 10, 2004. ISSN 1541-4922

Salmon, G. (2000) E-moderating: the key to teaching and learning online, London, Kogan Page

Salus, P.H. (1998) "Handbook of Programming Languages: Object-Oriented Programming Languages". Vol. 1. Macimillian. Indianapolis. 1998.

Sametinger, J. (1997) Software Engineering with Reusable Components. Springer Verlag, USA, 1997.

Santoro, F.M., Borges, M.R. & Santos, N. (2000) "An Infrastructure to Support the Development of Collaborative Project-Based Learning Environments" Proceedings of International Workshop on Groupware – CRIWG00, Madeira, Portugal, IEEE Press, pp. 78-85.

Santoro, F.M., Borges, M.R. & Santos, N. (2001) "Modelo de Cooperação para Aprendizagem Baseada em Projetos: Uma Linguagem de Padrões". In: 1ª Conferência Latino Americana em Linguagens de Padrão para Programação – SugarLoafPloP. Rio de Janeiro, Outubro de 2001.

Sauter, C., Morger, O., Muhlherr, M., Thutchytson, A. & Teusel, S. (1995) CSCW for Strategic Management in Swiss Enterprises: an Empirical Study. Proceedings of the 4th European Conference on Computer Supported Cooperative Work (ECSCW'95), Stockholm, Sweden, 117-132

Schmidt, K. & Rodden, T. (1996) Putting it all Together: Requirements for a CSCW Platform. In: Shapiro, D., Tauber, M., Traunmüller, R. (eds.) The Design of Computer Supported Cooperative Work and Groupware Systems. North Holland, Holland, pp. 157-176.

Schmidt, K. & Simone, C. (1996) Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. Computer Supported Cooperative Work, 5(2-3), 155-200.

Schmidt, K. (1991) "Riding a Tiger, or Computer Supported Cooperative Work", Proceedings of The 2nd European Conference on Computer-Supported Cooperative Work, Kluwer, 1-16.

Schön, D. & Bennet, J. (1996) "Reflective Conversation with Materials", In: Bringing Design to Software, Winograd, T. (ed), ACM Press, USA. ISBN 0-201-85491-0

Schön, D.A. (1983) The reflective practitioner: How professionals think in action, Basic Books, NY

Schrage, M. (1995) No more teams! Mastering the dynamics of creative collaboration, Nova York, EUA: Currency Doubleday

Schrage, M. (1996) Cultures Of Prototyping. In T. Winograd (ed) Bringing Design To Software, ACM Press, USA, pp. 191-205.

Schwarz, J., Sommer, H. & Farris, A. (2003) The ALMA Software System, ASP Conf. Ser., Vol. 314 Astronomical Data Analysis Software and Systems XIII, Ochsenbein, F., Allen, M. & Egret, D. (eds), San Francisco, ASP, 643

Segal, L. (1994) "Effects of Checklist Interface on Non-Verbal Crew Communications," Contractor Report 177639, NASA Ames Research Center.

Shum, S.B. & Hammond, N. (1994) “Argumentation-based design rationale: what use at what cost?,” Human-Computer Studies, USA, 40, p. 603-652.

Siebra, S.A., Salgado, A.C., Tedesco, P.A. & Brézillon, P. (2005) “Identifying the Interaction Context in CSCLE”, Proceedings of the 5th International and Interdisciplinary Conference (CONTEXT 2005), Paris, France, July 5-8, Lecture Notes in Computer Science, Vol 3554, Springer-Verlag.

Siegel, J. (2000) CORBA 3 Fundamentals and Programming, 2nd Edition, John Wiley & Sons, ISBN 0471295183

Simon, H.A. (1996) Models of my life, MIT Press, ISBN 0-262-69185-X

Simon, J.A. (1997) “Towards Mixed-Initiative Discursive Networking”, Computational Models for Mixed Initiative Interaction, March 24-26, 1997, Stanford University, California.

Siqueira, S.W.M., Braz, M.H.L.B. & Melo, R.N. (2003) “E-Learning Environment Based on Framework Compositio”, Third IEEE International Conference on Advanced Learning Technologies (ICALT'03), pp. 468.

Sire, S., Chatty, S., Gaspard-Boulinc, H. & Colin, F. (1999) How can groupware preserve our coordination skills? Designing for direct collaboration. Proceedings of Human-Computer Interaction INTERACT’99, IOSPress, pp 304–312.

Slagter, R.J. & Biemans, M.C.M. (2000) “Component Groupware: A Basis for Tailorable Solutions that Can Evolve with the Supported Task”, in Proceedings of the International ICSC Conference on Intelligent Systems and Applications (ISA 2000), Wollongong, Australia.

Slagter, R.J. & ter Hofte, H. (1999) Component models and component groupware architectures. TI/RS/99023, GIGACSCW/D2.1.1, Telematica Instituut, the Netherlands.

Slagter, R.J., Biemans, M, & Ter Hofte, G.H. (2001) “Evolution in use of groupware: Facilitating tailoring to the extreme” In: M. Borges, J. Haake and U. Hoppe (eds.), Proceedings of the 7th International Workshop on Groupware (CRIWG 2001), 6-8 September 2001, Darmstadt, Germany, 2001

Sommerville, I. (2003): Engenharia de Software. 6 ed. São Paulo: Addison Wesley, 2003.

Stafford, J.A. & Wolf, A.L. (2001) Software Architecture, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Stahl, G. (2001) WebGuide: Guiding collaborative learning on the Web with perspectives, Journal of Interactive Media in Education.

Stiemerling, O., Hallenberger, M. & Cremers, A.B. (2001) "A 3D Interface for the Administration of Component-Bases, Distributed Systems", Proceedings of the 5th International Symposium on Autonomous Decentralized Systems, p. 119.

Stiemerling, O., Hinken, R. & Cremers, A.B. (1999) The EVOLVE Tailoring Platform: Supporting the Evolution of Component-Based Groupware. In Proceedings of the 3rd International Enterprise Distributed Object Computing Conference (EDOC’99), pp. 106-115.

Streitz, N., Haake, J., Hannemann, J., Lemke, A., Schuler, W., Scheutt, H. & Thuring M. (1992) SEPIA: a cooperative hypermedia authoring environment. Proceedings of Proceedings of ACM Conference on Hypertext (ECHT'92), pp. 11-22.

Szyperski, C. (1997) Component Software: Beyond Object-Oriented Programming, Addison-Wesley, ISBN 0-201-17888-5

Szyperski, C. (2003) Component technology – what, where, and how? Procedings of the 25th International Conference on Software Engineering (ICSE'03), IEEE Computer Society, pp 684-693.

Tam, J. & Greenberg, S. (2004) A framework for asynchronous change awareness in collaboratively-constructed documents, CRIWG 2004, LNCS 3198, p. 67-83.

Tapscoot, D., Ticoll, D. & Lowy, A. (2000) Digital Capital: Harnessing the Power of Business Webs. Harvard Business School Press, USA.

Tatikonda, M.V & Stock, G.N. (2003) "Product Technology Transfer in the Upstream Supply Chain", The Journal of Product Innovation Management, Vol. 20, Product Development & Management Association, pp. 444-467

Teege, G. (1996) Object-Oriented Activity Support: A Model for Integrated CSCW Systems. Computer Supported Cooperative Work (CSCW): The Journal of Collaborative Computing, 5(1), pp. 93-124, 1996.

Teixeira, B. & Chagas, E.F. (2005) "Co-Autoria: Avaliação e Proposta de Requisitos para Ferramentas Segundo o Modelo 3C", Workshop de Informática na Escola, Congresso da Sociedade Brasileira de Computação 2005.

Teles, V.M. (2004) Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade, ed. Novatec, ISBN 8575220470.

Tietze, D.A., A Framework for Developing Component-based Co-operative Applications. PhD Dissertation, Computer Science, Technischen Universität Darmstadt, Germany, 2001.

Tripathi, A., Ahmed, T., Kumar, R. & Jaman, S. (2002) "Design of a Policy-Driven Middleware for Secure Distributed Collaboration", Proceedings of the 22nd International Conference on Distributed Computing System (ICDCS), Vienna, Austria, July 2002.

Tse, E. & Greenberg, S. (2004) "Rapidly Prototyping Single Display Groupware through the SDGToolkit", Proceedings of the 5th Australasian User Interface Conference, Volume 28 in the CRPIT Conferences in Research and Practice in Information Technology Series, Dunedin, NZ, pp. 101-110.

van der Veer, G.C., Lenting, B.F. & Bergevoet, B.A.J. (1996) GTA: Groupware Task Analysis - Modeling Complexity. Acta Psychologica 91, 1996, pp. 297-322

Wan, D. & Johnson, P.M. (1994) "Computer Supported Collaborative Learning Using CLARE: the Approach and Experimental Findings". Proceedings of ACM Conference on Computer Supported Cooperative Work, Chapel Hill, North Carolina, 1994.

Weinreich, R. & Sametinger, J. (2001) Component Models and Component Services: Concepts and Principles, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Werner, C.M.L. & Braga, R.M.M.B (2005) "A Engenharia de Domínio e o Desenvolvimento Baseado em Componentes", in: Desenvolvimento Baseado em Componentes, Gimenes, I.M.S. & Huzita, E.H.M. (eds), Editora Ciência Moderna, Rio de Janeiro, 2005. ISBN 85-7393-406-9, pg. 57-103.

Whorf, B.L. (1956) Language, Thought and Reality. Cambridge, MA: MIT Press.

Wills, A.L. (2001) Components and Connectors: Catalysis Techniques for Designing Component Infrastructures, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Winograd, T. & Flores, F. (1987) Understanding Computers and Cognition. Addison-Wesley, USA, 1987.

Won, M., Stiemerling, O. & Wulf, V. (2005) "Component-Based Approaches to Tailorable Systems", End User Development, Lieberman, H.;, Paterno, F. & Wulf, V. (eds), Kluwer, pp. 1-27.

Yang, Y. (1995) "Coordination for process support is not enough!" European Workshop on Software Process Technology (EWSPT4), LNCS 913, Leiden, The Netherlands, April 1995. Springer Verlag.

Artigos Publicados pelo Autor da Tese

Os artigos estão disponíveis para download no site http://groupware.les.inf.puc-rio.br

Capítulos de Livros

Fuks H., Pimentel, M.G., **Gerosa, M.A.**, Fernandes, M.C.P. & Lucena, C.J.P. (2006), "Novas Estratégias de Avaliação Online: Aplicações e Implicações em um Curso totalmente a Distância através do Ambiente AulaNet", *EaD Online: Avaliação* (título provisório), Silva, M. (ed.), Edições Loyola, Rio de Janeiro. (a ser publicado)

Fuks H., **Gerosa, M.A.** & Lucena, C.J.P. (2003), "Using the AulaNet Learning Environment to Implement Collaborative Learning via Internet", in: *Innovations 2003 – World Innovations in Engineering Education and Research*, Aung et al. (ed), iNEER, USA, 2003, Chap. 23, ISBN 0-9741252-0-2, pp. 225-235.

Fuks H., Cunha, M.L., **Gerosa, M.A.** & Lucena, C.J.P. (2003), "Participação e Avaliação no Ambiente Virtual AulaNet da PUC-Rio", *EaD Online: Teorias e Práticas*, Silva, M. (ed.), Edições Loyola, Rio de Janeiro, 2003, ISBN 85-15-02822-0, Cap. 15, pp. 231-254.

Fuks, H., **Gerosa, M.A.** & Pimentel, M.G. (2003), "Projeto de Comunicação em Groupware: Desenvolvimento, Interface e Utilização", *XXII Jornada de Atualização em Informática*, Anais do XXIII Congresso da Sociedade Brasileira de Computação, V2, Cap. 7, ISBN 85-88442-59-0, pp. 295-338.

Fuks, H., Raposo, A.B. & **Gerosa, M.A.** (2002), "Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas", *XXI Jornada de Atualização em Informática*, Anais do XXII Congresso da Sociedade Brasileira de Computação, V2, Cap. 3, ISBN 85-88442-24-8, pp. 89-128.

Periódicos / Journals

Fuks, H., **Gerosa, M.A.**, Pimentel, M.G., Filippo, D. & Lucena, C.J.P. (2006), "Informações Estatísticas e Visuais para a Mediação de Fóruns Educacionais", *Revista Brasileira de Informática na Educação*, ISSN 1414-5685, Sociedade Brasileira de Computação. (aceito para publicação)

Fuks, H., Raposo, A.B., **Gerosa, M.A.**, Lucena, C.J.P. (2005), "Applying the 3C Model to Groupware Development", *International Journal of Cooperative Information Systems* (IJCIS), v.14, n.2-3, Jun-Sep 2005, World Scientific, ISSN 0218-8430, pp. 299-328.

Fuks, H., **Gerosa, M.A.**, Raposo, A.B. & Lucena, C.J.P. (2004), "O Modelo de Colaboração 3C no Ambiente AulaNet", *Informática na Educação: Teoria e Prática*, Vol 7, No. 1, Porto Alegre, UFRGS, ISSN 1516-084X, pp. 25-48.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2003), "Analysis and Design of Awareness Elements in Collaborative Digital Environments: A Case Study in the AulaNet Learning Environment", *The Journal of Interactive Learning Research*, V. 14, No. 3, AACE, USA, ISSN 1093-023X, pp. 315-332.

Gerosa, M.A., Fuks, H., & Lucena, C.J.P. (2003), "Suporte à Percepção em Ambientes de Aprendizagem Colaborativa", *Revista Brasileira de Informática na Educação*, V. 11, No. 2, Julho/Dezembro 2003, ISSN 1414-5685, Sociedade Brasileira de Computação, pp. 75-85.

Fuks, H., **Gerosa, M.A.** & Lucena, C.J.P. (2002), "The Development and Application of Distance Learning on the Internet", *Open Learning Journal*, V. 17, No. 1, February 2002, ISSN 0268-0513, Cartafax Pub, pp. 23-38.

Fuks, H., **Gerosa, M.A.** & Lucena, C.J.P. (2002), "Usando a Categorização e Estruturação de Mensagens Textuais em Cursos pelo Ambiente AulaNet", *Revista Brasileira de Informática na Educação*, V. 10, No. 1, Abril 2002, ISSN 1414-5685, Sociedade Brasileira de Computação, pp. 31-44.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2001), "Tecnologias de Informação Aplicadas à Educação: Construindo uma Rede de Aprendizagem Usando o Ambiente AulaNet", *Informática na Educação: Teoria e Prática*, V. 4, No. 2, dezembro 2001, ISSN 1516-084X, UFRGS, pp. 63-74.

Fuks, H., **Gerosa, M.A.** & Lucena, C.J.P. (2001), "Sobre o Desenvolvimento e Aplicação de Cursos Totalmente a Distância na Internet", *Revista Brasileira de Informática na Educação*, No. 9, Setembro 2001, ISSN 1414-5685, Sociedade Brasileira de Computação, pp 61-75.

Anais de Congressos / Conferências

Gerosa, M.A., Pimentel, M.G., Filippo, D., Barreto, C.G., Raposo, A.B., Fuks, H. & Lucena, C.J.P. (2005) "Componentes Baseados no Modelo 3C para o Desenvolvimento de Ferramentas Colaborativas", *Anais do 5º Workshop de Desenvolvimento Baseado em Componentes - WDBC 2005*, 07-09 de Novembro, Juiz de Fora-MG, ISBN 85-88279-47-9, pp. 109-112.

Pimentel, M.G., **Gerosa, M.A.**, Filippo, D., Barreto, C.G., Raposo, A.B., Fuks, H. & Lucena, C.J.P. (2005) "AulaNet 3.0: desenvolvendo aplicações colaborativas baseadas em componentes 3C", *Anais do WCSCW2005 - Workshop Brasileiro de Tecnologias para Colaboração*, 07-08 de Novembro, Juiz de Fora-MG, pp. 761-770.

Gerosa, M.A., Pimentel, M.G., Fuks, H. & Lucena, C.J.P. (2005) "No Need to Read Messages Right Now: Helping Mediators to Steer Educational Forums Using Statistical and Visual Information", *Proceedings of the Computer Supported Collaborative Learning Conference – CSCL 2005*, 01-04 June, Taipei, Taiwan, ISBN 0-8058-5782-6, pp. 160-169.

Pimentel, M.G., **Gerosa, M.A.**, Fuks, H. & Lucena, C.J.P. (2005) "Assessment of Collaboration in Online Courses", *Proceedings of the Computer Supported Collaborative Learning Conference – CSCL 2005*, 01-04 June, Taipei, Taiwan, ISBN 0-8058-5782-6, pp. 494-498.

Gerosa, M.A., Pimentel, M., Raposo, A.B., Fuks, H., Lucena, C.J.P., "Towards an Engineering Approach for Groupware Development: Learning from the AulaNet LMS Development", *Proceedings of the 9th International Conference on CSCW in Design (CSCWiD)*, Vol. 1, Coventry, U.K., May 2005, ISBN 1-84600-004-1, pp. 329-333.

Gerosa, M.A., Raposo, A.B., Fuks, H. & Lucena, C.J.P. (2004) "Uma Arquitetura para o Desenvolvimento de Ferramentas Colaborativas para o Ambiente de Aprendizagem AulaNet", *Anais do XV Simpósio Brasileiro de Informática na Educação – SBIE 2004*, 09-12 de Novembro, Manaus-AM, ISBN 85-7401-161-4, pp. 168-177.

Fuks, H., Raposo, A.B., **Gerosa, M.A.**, Pimentel, M.G. & Lucena, C.J.P. (2004) "Suporte à Coordenação e à Cooperação em uma Ferramenta de Comunicação Textual Assíncrona: Um Estudo de Caso no Ambiente AulaNet", *Anais do I Workshop Brasileiro de Tecnologias para Colaboração – WCSCW 2004*, em conjunto ao WebMidia & LA-Web 2004, 13-14 de Outubro, Ribeirão Preto-SP, ISBN 85-7669-010-1, pp. 173-180.

Gerosa, M.A., Barreto, C.G, Raposo, A.B., Fuks, H. & Lucena, C.J.P. (2004) "O Uso de uma Arquitetura Baseada em Componentes para Incrementar um Serviço do Ambiente AulaNet", *Anais do 4º Workshop de Desenvolvimento Baseado em Componentes - WDBC 2004*, 15-17 de Setembro, João Pessoa-PB, ISBN 85-7669-001-2, pp. 55-60.

Raposo, A.B., **Gerosa, M.A.** & Fuks, H. (2004), "Combining Communication and Coordination toward Articulation of Collaborative Activities", *10th International Workshop on Groupware - CRIWG 2004*, 5-9 September, San Carlos, Costa Rica, Lecture Notes on Computer Science LNCS 3198, Springer-Verlag, ISBN 3540-230165, ISSN 0302-9743, pp. 121-136.

Gerosa, M.A., Pimentel, M.G., Fuks, H. & Lucena, C.J.P. (2004), "Analyzing Discourse Structure to Coordinate Educational Forums", *The 7th International Conference on Intelligent Tutoring Systems – ITS 2004*, Maceió-AL, August 30th to September, 3rd 2004, Lecture Notes on Computer Science LNCS 3220, Springer-Verlag, ISBN 3540-229485, ISSN 0302-9743, pp. 262-272.

Raposo, A.B., Pimentel, M.G., **Gerosa, M.A.**, Fuks, H. & Lucena, C.J.P. (2004), "Prescribing e-Learning Activities Using Workflow Technologies", *The 1st International Workshop on Computer Supported Activity Coordination – CSAC 2004*, International Conference on Enterprise Information Systems – ICEIS 2004, Porto, Portugal, April 13, 2004, pp. 71-80. ISBN 972-8865-08-2

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2004), "Estruturação e Categorização de Mensagens em Ferramentas de Comunicação Textuais Assíncronas", Electronic *Proceedings of the World Congress on Engineering and Technology Education - WCETE'2004*, March 14-17, Santos-SP.

Gerosa, M.A., Fuks, H., Raposo, A.B. & Lucena, C.J.P. (2004), "Awareness Support in the AulaNet Learning Environment", *Proceedings of the IASTED International Conference on Web-Based Education - WBE 2004*, ACTA Press, February 16-18, Innsbruck, Austria, pp. 490-495.

Gerosa, M.A., Raposo, A.B., Fuks, H., Lucena, C.J.P. (2003), "Combinando Comunicação e Coordenação em Groupware", *3ª Jornada Ibero-Americana de Engenharia de Software e Engenharia de Conhecimento – JIISIC 2003*, 26-28 de Novembro, Valdivia, Chile, pp. 145-154.

Gerosa, M.A., Pimentel, M.G., Fuks, H. & Lucena, C.J.P. (2003), "Coordenação de Fóruns Educacionais: Encadeamento e Categorização de Mensagens", *XIV Simpósio Brasileiro de Informática na Educação – SBIE 2003*, 12-14 de Novembro, Rio de Janeiro-RJ, pp. 45-54.

Fuks, H., Raposo, A.B. & **Gerosa, M.A.** (2003) "Do Modelo de Colaboração 3C à Engenharia de Groupware", *IX Simpósio Brasileiro de Sistemas Multimídia e Web – Webmidia 2003*, Trilha especial de Trabalho Cooperativo Assistido por Computador, 03-06 de Novembro, Salvador-BA, pp. 445-452.

Fuks, H., Mitchell, L.H.R.G, **Gerosa, M.A.** & Lucena, C.J.P. (2003) "Competency Management for Group Formation in the AulaNet Learning Environment", *9th International Workshop on Groupware - CRIWG 2003*, 28 September - 02 October, Grenoble, France, ISBN: 3-540-20117-3, Lecture Notes in Computer Science 2806, Springer-Verlag, pp. 183-190.

Raposo, A.B., **Gerosa, M.A.** & Fuks, H. (2003) "Modeling Coordination in Business-Webs", *Proceedings of the 3rd IFIP Conference on E-commerce, E-business and E-government*, 21-24 September 2003, Guarujá-SP, pp. 549-559.

Fuks, H., **Gerosa, M.A.**, Pimentel, M.G., Raposo, A.B., Mitchell, L.H.R.G. & Lucena, C.J.P. (2003) "Evoluindo para uma Arquitetura de Groupware Baseada em Componentes: o Estudo de Caso do Learningware AulaNet", *Anais eletrônico do 3º Workshop de Desenvolvimento Baseado em Componentes - WDBC 2003*, 10-12 de Setembro, São Carlos-SP.

Mitchell, L.H.R.G., **Gerosa, M.A.** & Fuks, H. (2003) "Comparação da Resolução Colaborativa de Problemas em Sala de Aula e através do Ambiente AulaNet", *WIE 2003 - IX Workshop de Informática na Escola*, Anais do XXIII Congresso da Sociedade Brasileira de Computação, V5, 2-8 de agosto. Campinas-SP, pp. 135-147.

Fuks, H., Cunha, L.M., **Gerosa, M.A.** & Lucena, C.J.P. (2003), "Analyzing and Assessing Collaborative Learning Activities in a Web-Based Environment", *Electronic Proceedings of the ICEE 2003 – International Conference on Engineering Education*, July 21-25, Valencia, Spain.

Fuks, H., Raposo, A.B. & **Gerosa, M.A.** (2002), "Engineering Groupware for E-Business", *First Seminar on Advanced Research in Electronic Business (EBR'2002)*, November 7-8, Rio de Janeiro-RJ, pp. 78-84.

Gerosa, M.A., Fuks, H., Raposo, A.B. & Mitchell, L.H.R.G. (2002), "Using Groupware Tools to Extend the Organizational Memory with Collaboration Aspects", *The 7th International Conference on CSCW in Design (CSCWiD)*, September 25-27, Rio de Janeiro-RJ, pp. 314-319.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2002), "Resultados da avaliação de um curso baseado na Web", *VIII Workshop de Informática na Escola - WIE 2002*, XXII Congresso da Sociedade Brasileira de Computação, V. 5, 17-19 de julho, Florianópolis-SC, pp. 477-485.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2002), "AulaNet-eLabora: Developing a groupware to bring the work and learning environment together", *Electronic Proceedings of the 7th International Conference on Engineering and Technology Education - INTERTECH 2002*, March 17-20, Santos-SP.

Fuks, H., **Gerosa, M.A.** & Lucena, C.J.P. (2002), "Using a Groupware Technology to Implement Cooperative Learning via the Internet - A case study", *Electronic Proceedings of the 35th Annual Hawaii International Conference on System Sciences – HICSS 35*, January 7-10, Hawaii.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2001), "Elementos de percepção como forma de facilitar a colaboração em cursos via Internet", *XII Simpósio Brasileiro de Informática na Educação – SBIE 2001*, 21-23 de Novembro, Vitória-ES, pp. 194-202.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2001), "Use of categorization and structuring of messages in order to organize the discussion and reduce information overload in asynchronous textual communication tools", *7th International Workshop on Groupware - CRIWG 2001*, September 6-8, Germany, pp 136-141.

Fuks, H., **Gerosa, M.A.**, Cunha, L.M. & Lucena, C.J.P. (2001), "Groupware Technology for cooperative learning via the Internet", *Electronic Proceedings of the International Conference on Engineering Education - ICEE 2001*, August 6-10, Oslo, Norway.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2001), "The application and enhancement of web based courses", *IASTED International Conference on Computers and Advanced Technology in Education – CATE 2001*, June 27-29, Banff, Canada, pp 8-11.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2001), "Uso da categorização e estruturação de mensagens para dinamizar a discussão e reduzir a sobrecarga de informação em cursos via Internet", *VII Workshop de Informática na Escola – WIE 2001*, Anais Eletrônicos do XXI Congresso da Sociedade Brasileira de Computação, 31 de julho a 2 de Agosto, Fortaleza-CE.

Cunha, L.M., **Gerosa, M.A.**, Fuks, H. & Lucena, C.J.P. (2001), "Desenvolvimento e aplicação de cursos totalmente a distância na Internet", *VII Workshop de Informática na Escola – WIE 2001*, Anais Eletrônicos do XXI Congresso da Sociedade Brasileira de Computação, 31 de julho a 2 de Agosto, Fortaleza-CE.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2001), "Um groupware baseado no ambiente AulaNet desenvolvido com componentes", *Anais eletrônicos do 1º Workshop de Desenvolvimento Baseado em Componentes*, 21-22 Junho, Maringá-PR.