1
Introdução

Nesta tese é proposto um processo específico para o desenvolvimento de groupware (software para dar suporte ao trabalho em grupo). A contribuição original desta tese é o uso do Modelo 3C de Colaboração em diferentes etapas do processo. Groupware e o Modelo 3C de Colaboração são discutidos na seção 1.1. Esta pesquisa emerge do projeto AulaNet, apresentado na seção 1.2, que consiste num groupware voltado para o ensino-aprendizagem pela web. Esta tese faz parte de um consórcio de pesquisa, apresentado na seção 1.3. O problema e o método de pesquisa desta tese são discutidos na seção 1.4. Uma breve revisão da literatura sobre processo de desenvolvimento de software e de groupware é apresentada na seção 1.5. A organização da escrita desta tese é apresentada na seção 1.6.

1.1.
Groupware e o Modelo 3C de Colaboração

O termo groupware, cunhado por Johnson-Lenz e Johnson-Lentz (1982), designa as aplicações computacionais projetadas para dar suporte ao trabalho colaborativo. Ellis et al. (1991, p.40) definem:

> "O objetivo de groupware é auxiliar grupos na comunicação, na colaboração e na coordenação de suas atividades. Especificamente, definimos groupware como um sistema baseado em computador para dar suporte a grupos de pessoas engajadas numa tarefa (ou objetivo) comum e que provê uma interface para um ambiente compartilhado."

Nesta tese, diferentemente da nomenclatura apresentada por Ellis *et al*., colaboração designa o trabalho realizado em conjunto enquanto o termo cooperação designa a ação de operar em conjunto, o ato de executar a tarefa em comum no espaço compartilhado. Dada esta ressalva sobre as diferenças entre nomenclaturas, a colaboração nesta tese, assim como proposto por Ellis *et al*., também é organizada em função das 3 dimensões: Colaboração = Comunicação + Coordenação + Cooperação, aqui denominado de Modelo 3C de Colaboração, representado esquematicamente na Figura 1.

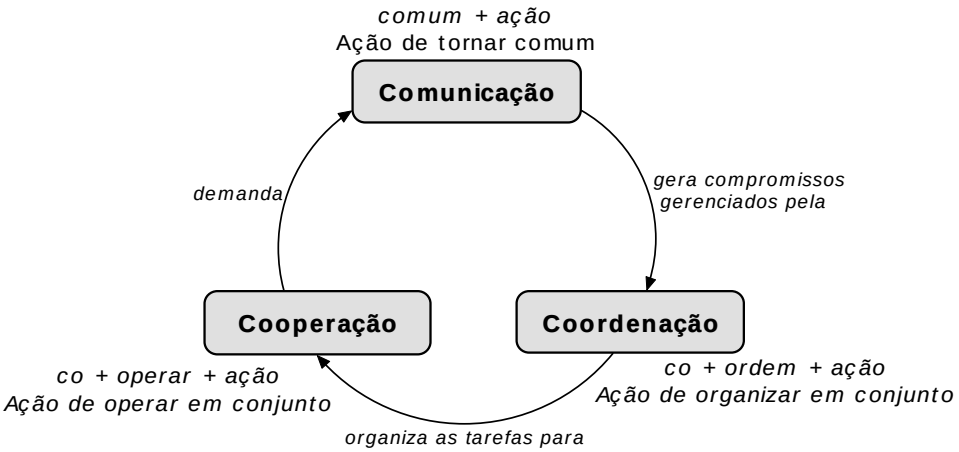


Figura 1. Modelo 3C de Colaboração

Colaboração, do latim *co + laborar + ação*, designa a ação de trabalhar em conjunto, a realização de um trabalho em comum realizado por duas ou mais pessoas (Ferreira, 1986). A colaboração tem sido compreendida nesta pesquisa a partir do Modelo 3C que enfatiza que um grupo, para colaborar, tem que estabelecer adequada Comunicação, Coordenação e Cooperação.

Comunicação, *comum + ação*, é a ação de tornar comum, trocar mensagens objetivando o entendimento mútuo, conversar, dialogar. Na colaboração, normalmente os membros do grupo se comunicam para a ação: negociam, tomam decisões e firmam compromissos (Winograd, 1988). Num grupo há pessoas com pontos de vista diferentes que podem gerar a complementação de entendimentos individuais (Fuks et al., 2002).

Coordenação, *co + ordem + ação*, é a ação de dispor segundo uma certa ordem e método, organizar, arranjar. A coordenação de um trabalho colaborativo objetiva organizar os membros do grupo para que os compromissos resultantes das negociações sejam realizados na ordem e tempo previstos cumprindo seus objetivos e restrições. Também objetiva evitar que esforços de comunicação e de cooperação sejam desperdiçados (Raposo et al., 2004).

Cooperação, *co + operar + ação*, é a ação de operar conjuntamente. Os membros do grupo atuam em conjunto, num espaço compartilhado, para a realização das tarefas definidas e organizadas durante a coordenação. Ao cooperarem, os indivíduos têm necessidade de se comunicar para renegociar e tomar decisões sobre situações não previstas, reiniciando o ciclo de colaboração esquematizado na Figura 1.

Para desenvolver groupware, é necessário entender de colaboração. A colaboração tem sido analisada nesta pesquisa a partir do Modelo 3C de Colaboração, que já foi usado em outros trabalhos para analisar, classificar e desenvolver groupware (Ellis, 2000; Baker et al., 2001; Laurillau e Nigay, 2002). Por exemplo, os tipos de aplicações groupware identificados originalmente por Ellis et al. (1991) foram classificados de acordo com do grau de suporte à comunicação, coordenação e cooperação, sendo posicionados no espaço triangular da Figura 2 (Teufel et al., 1995 *apud* Borghoff e Schlichter, 2000).

Figura 2. Classificação das aplicações groupware em função do Modelo 3C (Teufel et al., 1995 *apud* Borghoff e Schlichter, 2000)

Dentre os objetivos do grupo de pesquisa Groupware@LES[1], destaca-se a elaboração de uma Engenharia de Groupware baseada no Modelo 3C de Colaboração (Fuks et al., 2005). Esta tese contribui com a definição de um processo de desenvolvimento de groupware baseado neste modelo.

[1] http://groupware.les.inf.puc-rio.br

1.2.
Projeto AulaNet

AulaNet (Lucena e Fuks, 2000; Fuks et al., 2002) é um ambiente computacional para ensino-aprendizagem na web que vem sendo desenvolvido desde junho de 1997 pelo Laboratório de Engenharia de Software (LES) da Universidade Católica do Rio de Janeiro (PUC-Rio). O ambiente AulaNet é distribuído gratuitamente nas versões português, inglês e espanhol a partir de http://groupware.les.inf.puc-rio.br e http://www.eduweb.com.br

O ambiente AulaNet é um LMS, Learning Management System, onde são disponibilizados serviços para o docente selecionar e configurá-los de acordo com as dinâmicas educacionais que serão realizadas nas turmas de seu curso. Como ilustrado na Figura 3, os serviços configurados para o curso são acessados pelos aprendizes a partir de um controle remoto.

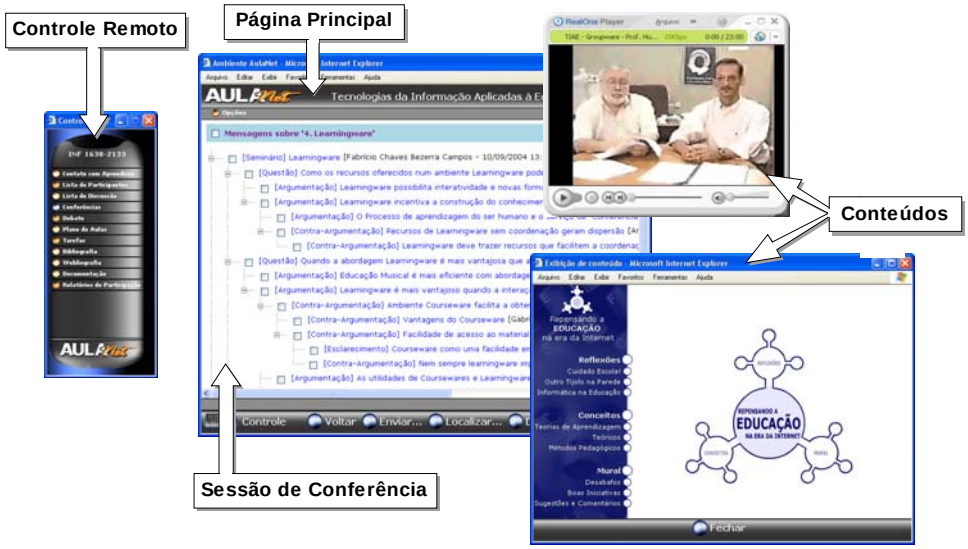
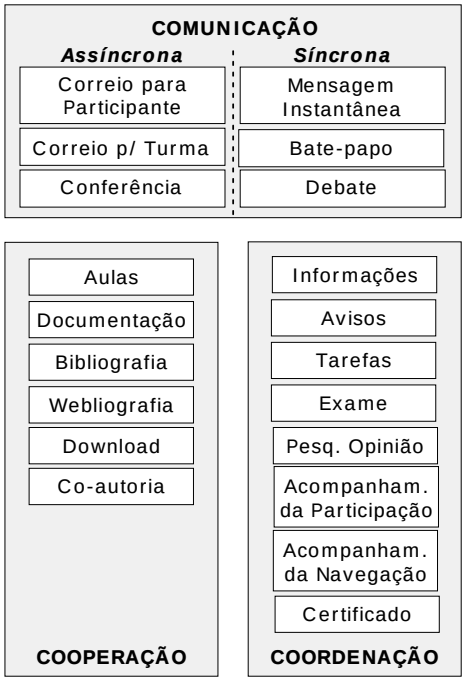


Figura 3. AulaNet 2.1

Os serviços do AulaNet são classificados em função do Modelo 3C de Colaboração como esquematizado na Figura 4.a. Cada serviço do AulaNet é caracterizado, nesta tese, como uma aplicação groupware conforme o mapeamento apresentado na Figura 4.b.



(a)

(b)

Figura 4. Serviços 3C do AulaNet 2.1

Atualmente é distribuída a versão AulaNet 2.1. Nesta versão, tem-se enfrentado dificuldades para desenvolver novos serviços e evoluir os existentes, sendo identificadas causas como a baixa modularidade e o uso de um paradigma funcional, ainda que o código esteja implementado através de classes. Estes problemas têm dificultado a integração de novos membros à equipe de desenvolvedores AulaNet, bem como a atuação de equipes externas. Para resolver

estes problemas, iniciou-se o Projeto AulaNet 3.0 cujo objetivo é reestruturar o código tornando-o baseado em componentes (Fuks et al., 2003).

1.3. Consórcio de Pesquisa

Para investigar o desenvolvimento de groupware e sua aplicação no desenvolvimento do AulaNet 3.0, nosso grupo de pesquisa Groupware@LES consorciou três trabalhos: Barreto (2006), em sua dissertação de mestrado, propõe a integração de *frameworks* na constituição da arquitetura técnica do AulaNet; Gerosa (2006), em sua tese de doutorado, propõe a montagem de groupware a partir da agregação de serviços e componentes baseados no Modelo 3C de Colaboração; e esta tese de doutorado, onde proponho um processo de desenvolvimento de groupware usando o Modelo 3C de Colaboração em diferentes etapas do processo. Estes trabalhos consorciados reduzem a distância semântica entre a implementação e os conceitos do domínio referentes à colaboração, o que favorece a manutenção e a evolução do groupware. Com o objetivo de contextualizar os três trabalhos, esta seção é replicada na introdução da dissertação e das teses resultantes. As subseções seguintes resumem cada trabalho.

1.3.1. Agregando Frameworks de Infra-Estrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente AulaNet

No desenvolvimento de um groupware, o projetista se depara com desafios em diferentes níveis: entender do domínio e lidar com questões de infra-estrutura. O desenvolvimento de groupware é difícil devido ao seu caráter multidisciplinar e à heterogeneidade dos diversos grupos de trabalho. O desenvolvedor deveria se concentrar mais nos aspectos funcionais utilizando uma infra-estrutura que trate as questões técnicas.

Na dissertação de Barreto (2006), foi elaborada uma arquitetura técnica multicamadas que faz uso do padrão MVC (Fowler, 2002) e que integra frameworks de infra-estrutura (Fayad & Schmidt, 1997; Fayad et al. 1999a; Fayad

et al., 1999b; Fayad & Johnson, 2000). A abordagem multicamadas com o padrão MVC proporciona a separação entre a lógica da aplicação e a interface com o usuário, considerada uma boa prática de design de software (Fowler, 2002). Frameworks de infra-estrutura proporcionam uma maneira de lidar com as questões de baixo nível como persistências de dados, controle de transações, segurança, etc.

O diagrama esquematizado na Figura 5 mostra a arquitetura técnica proposta para o AulaNet 3.0. As setas indicam o fluxo de controle da aplicação, os retângulos representam classes, os círculos representam as interfaces, e as linhas pontilhadas representam a divisão entre as camadas. Esta arquitetura, baseada na Arquitetura de POJOs descrita por Johnson (2002; 2004), é organizada nas seguintes camadas: apresentação, negócios e recursos.

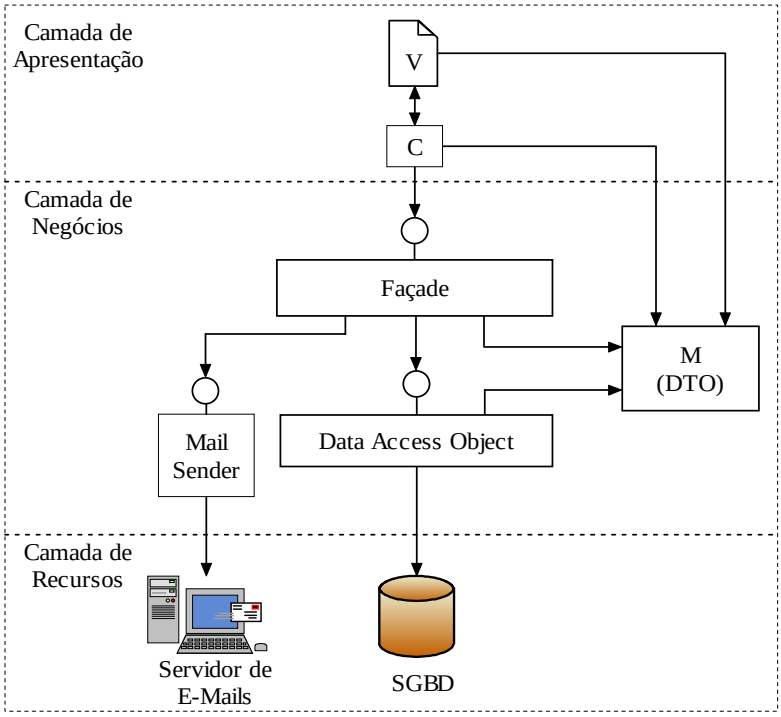


Figura 5. Arquitetura Técnica do AulaNet 3.0.

A camada de recursos relaciona os recursos externos necessários para que a aplicação seja executada. Na arquitetura do AulaNet 3.0 estão previstos o uso de banco de dados relacional (SGBD) e um servidor de e-mails.

A camada de negócios implementa a lógica da aplicação utilizando POJOs (2005) – Plain Old Java Objects.

> "O termo [POJO] foi cunhado enquanto eu [Martin Fowler], Rebbecca Parsons e Josh MacKenzie estávamos nos preparando para uma conferência em Setembro de 2000. Na palestra estávamos levantando os vários benefícios de codificar a lógica de negócios usando objetos Java comuns em vez de usar Beans de Entidade [EJB]. Questionávamos por que as pessoas eram tão contra usar objetos comuns em seus sistemas, e concluímos que era pela falta de um nome pomposo para os objetos simples. Então inventamos um, e o termo pegou muito bem." (POJO, 2005)

Na camada de negócios, o modelo (representado no diagrama da Figura 5 pela letra M do MVC) é implementado por classes que realizam o padrão de projetos *Data Transfer Object* (Fowler, 2002), usadas para transportar os dados das entidades de negócio entre camadas. O acesso à base de dados é encapsulado através de classes que implementam o padrão de projetos *Data Access Objects* - DAO (Alur et al., 2001), o que possibilita variar a maneira de persistir as classes do modelo sem que seja preciso reescrever o código cliente. Mail Sender é a classe para enviar e-mails que, de forma similar ao DAO, encapsula o acesso ao servidor de e-mails. A lógica de negócios é exposta para a camada de apresentação através de um *Façade* (Gamma et al., 1995), que é o padrão de projeto para prover uma interface para acesso às funcionalidades de um serviço.

A camada de apresentação expõe a lógica de negócios ao usuário-final. Na arquitetura do AulaNet 3.0, esta camada é composta pelo controlador (representado no diagrama da Figura 5 pela letra C do MVC) e por páginas JSP que implementam a camada de Visão (representado no diagrama da Figura 5 pela letra V do MVC). O controlador chama os métodos do *Façade*, acessando a camada de negócios. Os DTOs resultantes de operações são passados à visão que exibe as informações ao usuário.

Frameworks de infra-estrutura foram selecionados e acrescentados a esta arquitetura para prover persistência de dados, gerenciamento de transações entre outros aspectos. Estes frameworks possibilitam ao desenvolvedor tratar estes aspectos com uma visão em alto nível, concentrando-se em seu domínio de aplicação, no caso, colaboração.

1.3.2. Desenvolvimento de Groupware Componentizado com base no Modelo 3C de Colaboração

Um groupware é composto de ferramentas colaborativas como Fórum, Agenda, Documentação, etc. Estas ferramentas, disponíveis em diversas

aplicações groupware, compartilham funcionalidades relativas ao suporte computacional à colaboração, tais como canal de comunicação, gerenciamento de participantes, registro de informações, etc.

Na tese de Gerosa (2006), para dar suporte ao desenvolvimento de groupware, foram estabelecidos dois níveis de componentização. O primeiro nível é constituído de serviços colaborativos que, por sua vez, são montados com componentes 3C (segundo nível) que implementam funcionalidades relacionadas à colaboração. Estes componentes são distribuídos em *component kits* organizados em função do Modelo 3C de Colaboração para que desenvolvedores montem aplicações colaborativas. Nesta abordagem, cada serviço usa componentes de comunicação, coordenação e de cooperação independentemente da classificação 3C do serviço. Foi aplicado um método de Engenharia do Domínio para elaborar o conjunto de componentes. Os componentes são iterativamente refinados em função da realimentação obtida com o desenvolvimento dos serviços do AulaNet 3.0 e em função de estudos de caso variando as configurações do suporte à colaboração.

Component frameworks (Syzperski, 1997) são usados para oferecer suporte ao gerenciamento e à execução dos componentes. Conforme apresentado naFigura 6, nesta tese foi elaborado um *component framework* para cada nível de componentização (serviço e componente 3C). Os serviços são acoplados no Service Component Framework, e os componentes 3C são acoplados no Collaboration Component Framework. Estes *component frameworks* são responsáveis por tratar a instalação, remoção, atualização, ativação, desativação, localização, configuração e monitoramento de componentes. O Service Component Framework gerencia as instâncias dos serviços e a ligação com os componentes de colaboração correspondentes. O Collaboration Component Framework gerencia as instâncias dos componentes de colaboração, que são provenientes do Collaboration Component Kit. Algumas funcionalidades dos *component frameworks* são recorrentes, sendo então elaborado um *framework* para instanciar os *component frameworks*. Este tipo de *framework* é chamado de *component framework framework* (CFF) (Szyperski, 1997, p.277). Um *component framework framework* é visto como um *component framework* de segunda ordem, onde seus componentes são *component frameworks* (Szyperski, 1997, p.276). Na

arquitetura da aplicação, o *component framework* de segunda ordem foi denominado Groupware Component Framework Framework.

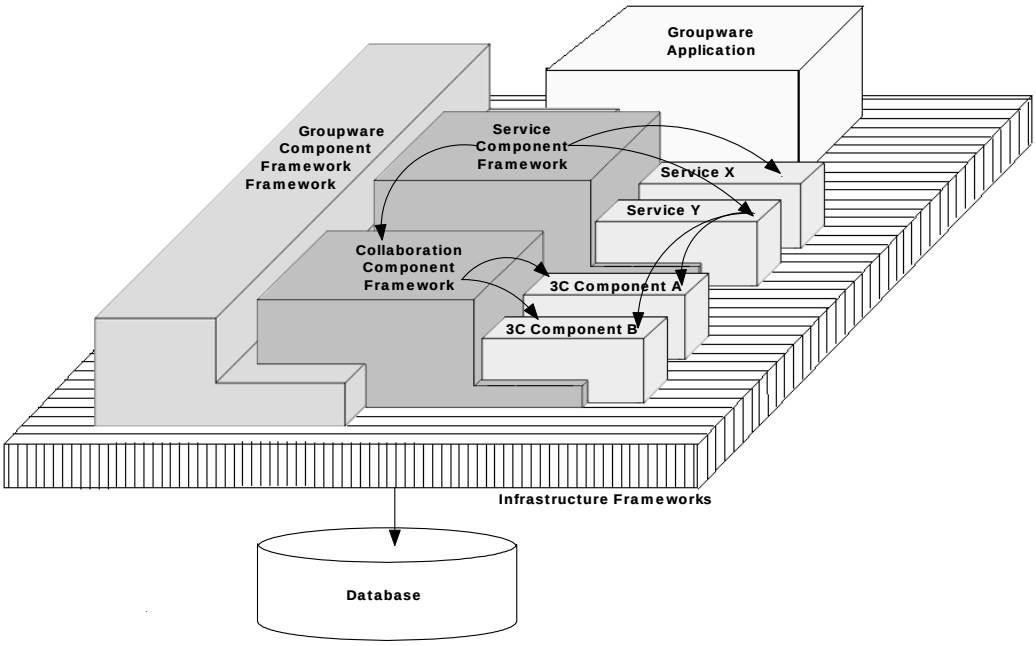


Figura 6. A arquitetura de aplicação proposta

Os *component frameworks*, serviços e componentes 3C oferecem suporte computacional aos conceitos do Modelo 3C de Colaboração, instrumentando o desenvolvimento da camada de negócio. A arquitetura de aplicação proposta estrutura os componentes do domínio, representando um projeto lógico de alto nível independente da tecnologia de suporte (D'Souza & Wills, 1998). Já os aspectos de infra-estrutura, tratados na dissertação de Barreto (2006), são independentes do domínio de aplicação.

Os componentes da arquitetura de aplicação são implementados segundo a arquitetura técnica. Os serviços do AulaNet são criados com um único *Façade* que expõe as operações deste serviço para a camada de apresentação. Os componentes de colaboração por sua vez, podem utilizar vários DTOs e DAOs, dependendo da complexidade do componente. Estes componentes podem ainda usar "código cola" (Szyperski, 1997) e adaptadores (D'Souza & Wills, 1998) para possibilitar a integração com componentes e outros sistemas que não são compatíveis por construção.

1.3.3.
RUP-3C-Groupware: um Processo de Desenvolvimento de Groupware baseado no Modelo 3C de Colaboração

Os *frameworks* de infra-estrutura elaborados por Barreto (2006) se encarregam de soluções para aspectos de infra-estrutura de baixo nível, visando possibilitar o desenvolvedor se concentrar nos aspectos funcionais. Os *kits* de serviços e componentes 3C elaborados por Gerosa (2006) fornecem os elementos para compor um groupware. O processo elaborado na tese aqui apresentada estabelece os passos a serem seguidos na montagem do groupware, pois ainda que se construa uma aplicação groupware para um grupo com uma determinada dinâmica, com o tempo surgem novas situações onde são identificados novos problemas. A aplicação necessitará ser modificada para não se manter inadequada.

Um processo organiza, em linhas gerais, uma seqüência de passos onde são incorporadas diretrizes e boas práticas que, quando seguidas, levam à produção de um software razoável (Sommerville, 2003; Beck, 2004; Philippe, 2003). Coexistem abordagens diferentes para o desenvolvimento de software, dentre elas, o desenvolvimento baseado em componentes, que é uma estratégia recente que tem se tornado cada vez mais usada (Sommerville, 2003; Gimenes & Huzita, 2005). Seguindo esta abordagem, tornaram-se conhecidos processos como Catalysis (D'Souza e Wills, 1998), *UML Components* (Cheesman & Daniels, 2001) e RUP – Rational Unified Process (Philippe, 2003). O processo formalizado nesta tese, denominado RUP-3C-Groupware, também faz uso da abordagem baseada em componentes, estendendo o RUP para o desenvolvimento específico de aplicações groupware.

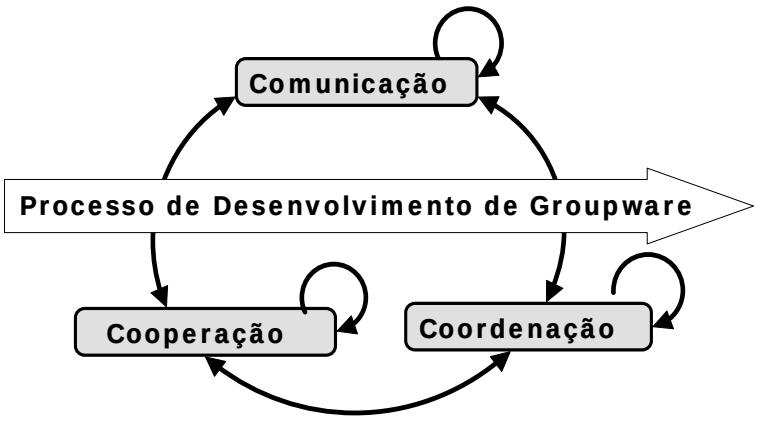


Figura 7. Foco para o desenvolvimento de uma versão da aplicação groupware com base no Modelo 3C de Colaboração

No processo proposto, o Modelo 3C de Colaboração é usado nas diferentes etapas do processo: na análise de domínio para classificação das aplicações groupware e de seus elementos; na construção de componentes (Gerosa, 2006); e no foco dado para o desenvolvimento de cada versão – de acordo com essa prática, a aplicação groupware é desenvolvida resolvendo um problema de comunicação, de coordenação ou de cooperação, um a cada versão ao longo do ciclo de desenvolvimento como esquematizado na Figura 7.

1.4. Problema e Método de Pesquisa

Engenheiros de Software não estão preparados para desenvolver groupware. Groupware é difícil de desenvolver porque requer conhecimento multidisciplinar, como informática, colaboração, sociologia, cognição etc (Gutwin & Greenberg, 2000). O problema focado nesta tese pode ser assim enunciado: como auxiliar Engenheiros de Software no desenvolvimento de groupware?

Para este problema, diversas soluções não-excludentes podem ser propostas, tal como a construção de uma arquitetura (Celso, 2006) ou o desenvolvimento de *toolkits* (Gerosa, 2006). Nesta tese, investiga-se o uso de um processo específico de desenvolvimento de groupware.

O processo aqui proposto consiste na formalização da experiência adquirida ao longo dos 8 anos do Projeto AulaNet, e mais especificamente a partir da minha experiência de 5 anos de desenvolvimento da ferramenta Mediated Chat que compõe o serviço Debate do ambiente AulaNet. A abordagem adotada de pesquisa e desenvolvimento do Mediated Chat é de Projeto Iterativo (Collins *et al.*, 2004): uma nova funcionalidade é introduzida, a ferramenta é usada em casos reais para promover os debates de um curso online, os resultados são analisados e então a tecnologia é reprojetada numa nova iteração. A pesquisa relacionada ao Mediated Chat já recebeu prêmios em conferências nacionais (SBIE) e têm obtido visibilidade internacional tornando-se uma referência na área.

Através do processo aqui formalizado, denominado RUP-3C-Groupware, fornecidas diretrizes e práticas que se mostraram adequadas para o desenvolvimento de groupware. Portanto, nesta tese foi usado o método científico de pesquisação (“action research”) que parte da prática, da ação pragmática

declaradamente dedicada à resolução imediata de situações problemáticas reais. O uso do método de pesquisação na área de Sistemas de Informação tem aumentado desde o final dos anos noventa, e consiste numa das poucas abordagens de investigação para estudar métodos de desenvolvimento em contextos organizacionais (Machado, 2004).

Além do processo ter surgido da prática, e ter sido especificado de acordo com o *framework* de processos RUP, procurou-se ainda avaliar se o processo proposto poderia ser usado por outros Engenheiros de Software não pertencentes ao projeto AulaNet. Foi então realizado um estudo de caso para avaliar a repetitividade do processo, investigando se alunos de Engenharia de Software conseguiriam produzir adequadamente alguns artefatos-chave específicos do processo proposto.

1.5. Processos de Desenvolvimento de Software e de Groupware

Esta seção apresenta uma breve revisão da literatura sobre processo de desenvolvimento de software e de groupware, para situar o processo proposto nessa tese. Na subseção 1.5.1 discute-se a definição e os objetivos de um processo de desenvolvimento de software. Na subseção 1.5.2 são discutidos os modelos de processos, tais como o modelo em cascata e o espiral. Os processos específicos para o desenvolvimento de groupware são abordados na subseção 1.5.3.

1.5.1. Definição e objetivos

"Um processo de software é um conjunto de atividades e resultados associados que geram um produto de software" (Sommerville, 2003, p.7). O processo é o fundamento da Engenharia de Software, é o que possibilita o desenvolvimento racional do software através da efetiva utilização da tecnologia de engenharia (Pressman, 2004, p. 18). Um processo de desenvolvimento de software constitui a base para o controle gerencial de projetos de software, e estabelece o contexto para aplicação de métodos na produção de artefatos (modelos, documentos, dados, relatórios, formulários etc.). Nesta tese, adota-se a definição de processo apresentada no RUP (detalhada na seção 3.1):

> "Um processo descreve *quem* está fazendo *o quê*, *como* e *quando*. O Rational Unified Process é representado usando quatro elementos primários de modelagem: Trabalhadores: *quem*; Atividades: *como*; Artefatos: *o quê*; Fluxos: *quando*" (Kruchten, 2003, p.29).

Dentre os objetivos que um processo de desenvolvimento deve alcançar, Tyrrell (2000) destaca:

Eficácia. Um processo eficaz auxilia o desenvolvimento do produto correto. Um bom processo deve auxiliar os desenvolvedores no levantamento das necessidades do cliente, produzir o que o cliente precisa e, principalmente, verificar se o que foi produzido é realmente o que o cliente precisa. Se o produto não for o que o cliente requisitou, não é bom. Não se deve confundir com eficiência, que está relacionada com a rapidez e a economia de desenvolvimento.

Manutenibilidade. Um dos objetivos de um bom processo é expor o raciocínio usado pelos projetistas e programadores para que se possa identificar defeitos e onde realizar mudanças. Deve-se evitar perder informações relevantes com a saída de um membro da equipe de desenvolvimento que detenha um determinado conhecimento sobre o projeto. Deve facilitar o trabalho de manutenção decorrente de alterações de requisitos. Deve auxiliar o reuso de elementos do software em outros produtos.

Preditibilidade. O desenvolvimento de um novo produto precisa ser planejado. Um plano é usado para alocar recursos como tempo e pessoas, por isso é importante predizer corretamente os recursos necessários para desenvolver o produto. Um bom processo deve ajudar a planejar os passos de desenvolvimento e estimar recursos. Acima de tudo, a consistência do processo deve possibilitar aprender com outros projetos executados.

Repetitividade. Um processo deve ser replicável em outros projetos. Processos elaborados sob demanda (*ad-hoc*) raramente podem ser reusados se não forem mantidas a mesmas condições e as mesmas pessoas trabalhando no novo projeto. É uma grande perda de tempo e sobrecarga produzir um processo a partir do zero para cada projeto. É mais adequado realizar adaptações de processos existentes.

Garantia de Qualidade. O objetivo de um processo definido é possibilitar que engenheiros de software desenvolvam produtos com alta qualidade. Qualidade pode ser definida como a adequação do produto aos propósitos. O

processo deve prover um relacionamento entre o requisitado pelo cliente e o produto desenvolvido.

Perfectibilidade (Melhoria Contínua). Um processo deve poder ser aperfeiçoado. Não se pode esperar que um processo esteja tão perfeito que não possa ser melhorado. Tecnologias, ambiente de trabalho, estratégias de desenvolvimento e os produtos requisitados estão em constante mudança, por isso o processo deve ser passível de modificação em busca de melhoria contínua.

Rastreabilidade. Um processo deve possibilitar que gerente, desenvolvedores e clientes acompanhem o projeto. Rastreabilidade relaciona-se com a predição, pois através do rastreamento do projeto é possível avaliar as predições e, desta forma, melhorá-las.

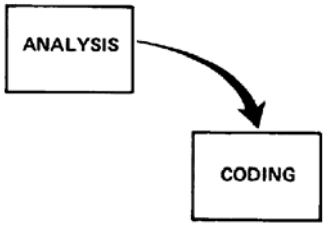
Destas características, sobre o processo aqui elaborado, procurou-se investigar a repetitividade do RUP-3C-Groupware, como discutido no capítulo 4.

1.5.2. Modelos de Processo de Software

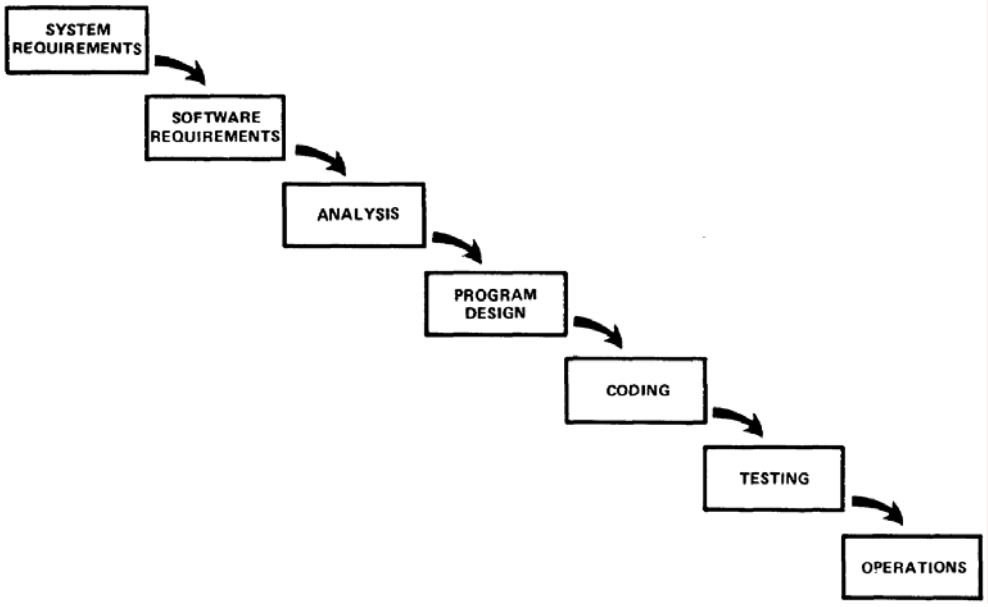
Um modelo de processo de software (“paradigmas de processo” ou “ciclo de vida”), é uma descrição simplificada de um processo de software, uma abstração útil para explicar as diferentes abordagens de desenvolvimento (Sommerville, 2003). São identificados quatro modelos: modelo em cascata, desenvolvimento evolucionário, transformação formal, e baseado em componentes.

Modelo em Cascata

O modelo em cascata, proposto por Royce (1970), também denominado “modelo seqüencial linear”, “abordagem top-down” ou “ciclo de vida clássico”, é um dos primeiros e mais importantes modelos, pois se tornou referência, uma espécie de gabarito para muitos dos modelos modernos e ainda continua sendo amplamente usado (Summerville, 2003; Pressman, 2002). Este modelo foi proposto como contraposição à abordagem composta apenas de análise e codificação, Figura 8.a, que só se aplica quando o esforço de desenvolvimento é pequeno e o produto final é para ser operado por quem o construiu; apenas estes dois passos fracassam na produção de programas grandes a serem entregues para clientes, sendo adequadas várias outras etapas, Figura 8.b. (Royce, 1970, p.1).



a) Etapas de implementação de um programa pequeno de computador



b) Etapas de implementação de um programa grande de computador

Figura 8. Modelo em Cascata (Royce, 1970)

Seguindo um ciclo convencional de engenharia, sistemático e controlado, o modelo estabelece as seguintes etapas para o desenvolvimento de software:

- Requisitos: funcionalidades, restrições e objetivos são estabelecidos junto com o cliente e os usuários do sistema de software;
- Análise: os requisitos eliciados na etapa anterior são detalhados em termos de funcionalidades, comportamento, desempenho, e interface-com-usuário para servir como especificação do software a ser construído;
- Projeto: a arquitetura geral do sistema de software é estabelecida, sendo descritas as abstrações fundamentais e as relações entre elas;
- Codificação: o projeto é traduzido para uma linguagem de programação;
- Teste: são realizados os testes para descobrir erros e verificar se os requisitos foram atendidos.
- Operação e Manutenção: o sistema de software é entregue para o cliente, sendo instalado e colocado em operação. A manutenção envolve corrigir erros

não descobertos em estágios anteriores ou modificar o sistema à medida que o cliente requisita novas funcionalidades ou melhoria de desempenho.

O princípio do modelo em cascata é que as diferentes etapas seguem uma seqüência: a saída de uma etapa 'flui' para a etapa seguinte, o desenvolvimento só prossegue quando uma etapa tiver sido concluída. O modelo original prevê ciclos de realimentação, mas as iterações são estabelecidas apenas indiretamente, e a maioria das organizações trata o modelo como se fosse estritamente linear (Pressman, 2002, p.26).

Dentre as principais críticas ao modelo em cascata, destacam-se (Hanna, 1995): é difícil estabelecer adequadamente todos os requisitos logo no começo do projeto; uma versão executável só fica disponível no término do projeto e um erro grosseiro pode ser desastroso se for detectado apenas quando o sistema estiver em execução; projetos reais raramente se atêm ao fluxo seqüencial estabelecido.

O modelo em cascata é adequado para os projetos de software em que os requisitos são bem conhecidos e definidos desde o início.

Desenvolvimento iterativo evolucionário

Nesta abordagem, um sistema é desenvolvido através de sucessivas versões. Deve-se rapidamente gerar um executável a partir de especificações iniciais. Em seguida, deve-se refiná-lo a partir de *feedback* do cliente visando produzir um sistema que satisfaça as suas necessidades. O sistema é então entregue ou, como alternativa, reimplementado usando uma abordagem mais estruturada para produzir um sistema mais robusto com maior capacidade de manutenção.

Há duas principais estratégias de desenvolvimento evolucionário: protótipos descartáveis e desenvolvimento exploratório. O objetivo de se construir protótipos descartáveis é elicitar os requisitos que estejam mal compreendidos objetivando desenvolver uma boa especificação. Com o desenvolvimento exploratório, o desenvolvimento inicia-se com as partes do sistema mal compreendidas e evolui com o acréscimo de novas características à medida com que são requisitadas pelo cliente.

Seguindo o modelo evolucionário, tornou-se bem conhecido o processo de desenvolvimento em espiral proposto por Boehm (1988), Figura 9. Em vez de uma seqüência linear de atividades, este processo é representado como uma espiral onde cada volta da espiral representa uma fase do processo: a volta mais

interna relaciona-se à viabilidade do sistema; a volta seguinte, à definição dos requisitos; a próxima volta, ao projeto; e assim por diante.

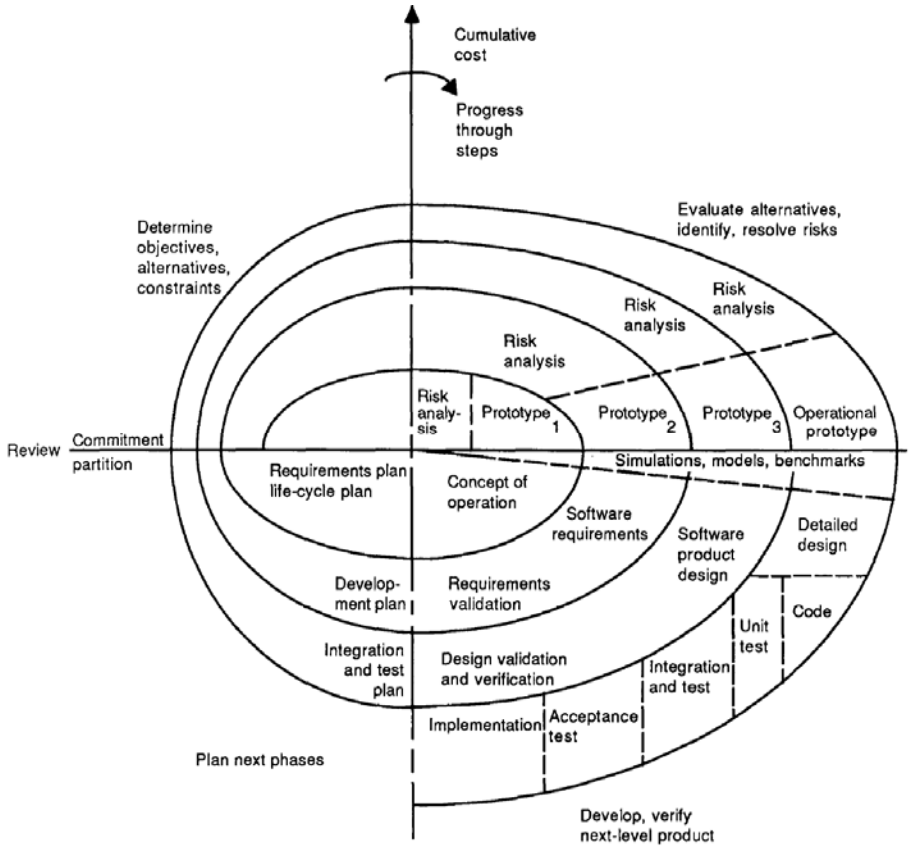


Figura 9. Modelo Espiral de processo de software (Boehm, 1988, p.64)

Cada volta da espiral, por sua vez, é dividida em quatro setores:

- Definição de objetivos: são definidos os objetivos de cada etapa do projeto sendo preparado um plano de gerenciamento incluindo os riscos e alternativas;
- Avaliação e redução de riscos: para cada risco identificado, é realizada uma análise e tomadas providências para diminuir o risco. Por exemplo, se há risco dos requisitos serem inadequados, poderá ser desenvolvido um protótipo;
- Desenvolvimento e validação: escolhe-se um modelo para o desenvolvimento do sistema;
- Planejamento: o projeto é revisto para decidir se deve ser continuado. Se a decisão for continuar, então é planejada a próxima fase do projeto, dando início a uma nova volta na espiral.

A contribuição do modelo em espiral é a explícita gerência de projeto considerando-se os riscos (Sommerville, 2003, p.45). Não há fases fixas no modelo espiral e este modelo faz uso de outros; por exemplo, a prototipação pode

ser usada para resolver dúvidas relativas aos requisitos e esta fase pode ser seguida por um desenvolvimento convencional em cascata.

Na abordagem incremental, sugerida inicialmente por Mills *et al.* (1980), os clientes identificam algumas funções e as ordenam pela relevância; em seguida, define-se uma série de entregas fornecendo um subconjunto das funcionalidades do sistema na ordem estabelecida de prioridade. O objetivo é adiar algumas decisões sobre o detalhamento de requisitos até que clientes e usuários tenham alguma experiência com o sistema. Esta abordagem encontra-se recentemente em evidência com a "programação extrema" (Beck 2004).

Em comparação com o desenvolvimento em cascata, o desenvolvimento evolucionário tem a vantagem de desenvolver a especificação gradativamente, conforme os usuários compreendem melhor o sistema, evitando assim que erros grosseiros sejam identificados somente no final do processo. Contudo, a partir da perspectiva de engenharia e de gerenciamento, são identificados os seguintes problemas (Sommerville, 2003, p.39-40):

- Os sistemas são frequentemente mal-estruturados: a mudança constante tende a corromper a estrutura do software, tornando-se cada vez mais difícil e oneroso realizar modificações (esta é a atual situação do projeto AulaNet 2.1);
- O processo não é visível: os sistemas são produzidos rapidamente, não sendo produzidos documentos que reflitam cada versão, dificultando a medição do progresso do desenvolvimento e, consequentemente, dificultando a gerência do projeto.

Transformação formal

Nesta abordagem, o sistema é especificado através de uma rigorosa notação matemática. Em seguida, são aplicados métodos formais para transformar a especificação num programa. Ambigüidades e inconsistências são descobertas e corrigidas, não através de revisões comuns, mas através da aplicação de análise matemática. Estas transformações 'preservam a correção', garantido que o programa produzido esteja livre de erros (Sommerville, 2003; Pressman, 2002).

Esta abordagem é adequada ao desenvolvimento de sistemas que tenham rigorosas exigências de segurança e confiabilidade, como os sistemas de aeronaves e dispositivos médicos. Contudo, fora destes domínios especializados,

os processos com base em transformação formal não são amplamente usados porque (Sommerville, 2003, p.40-41; Pressman, 2002, p.41):

- requer perícia especializa, tendo poucos desenvolvedores de software com treinamento necessário para aplicar métodos formais;
- o desenvolvimento de modelos formais é atualmente lento e dispendioso;
- em relação a outras abordagens, não oferece vantagens significativas de custo ou qualidade para a maioria dos sistemas;
- é difícil usar os modelos como um mecanismo de comunicação com clientes, geralmente despreparados tecnicamente.

Desenvolvimento Baseado em Componentes

O desenvolvimento baseado em componentes é uma estratégia recente que tem se tornado cada vez mais usada (Sommerville, 2003; Gimenes e Huzita 2005). Esta técnica supõe que partes do sistema já existem, e o desenvolvimento concentra-se na integração destas partes. O foco é no reuso de componentes (desenvolvimento COM reuso), mas eventualmente desenvolvendo novos componentes (desenvolvimento PARA reuso). A Figura 10 (Pressman, 2002, p.40) apresenta uma descrição simplificada das atividades realizadas no desenvolvimento baseado em componentes.

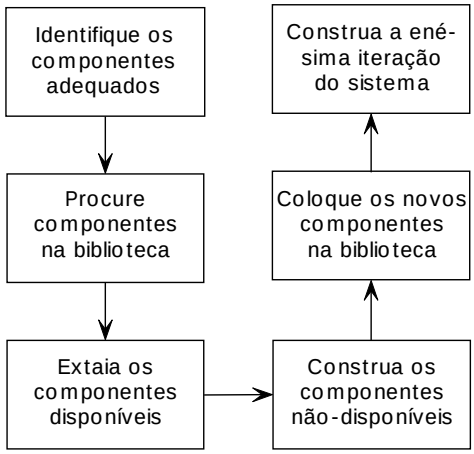


Figura 10. Desenvolvimento Baseado em Componentes (Pressman, 2002, p. 40)

Seguindo esta abordagem, tornaram-se conhecidos processos como o Catalysis (D'Souza e Wills, 1998), UML Components (Cheesman e Daniels, 2001) e Rational Unified Process (Kruchten, 2003), sendo este último adotado como base para a especificação do processo de groupware elaborado nesta tese, e encontra-se detalhado na seção 4.1.

1.5.3.
Processos de Desenvolvimento de Groupware

Dado que groupware também é software, questiona-se a necessidade de um processo específico. O desenvolvimento de groupware requer competências e procedimentos específicos, tais como: conhecimento sobre colaboração; métodos de análise de domínio; métodos etnográficos; realização de estudos de caso; e avaliação diferenciada da interface-com-usuário (que não é mais entre usuário e sistema, mas sim entre usuário e grupo). Na literatura específica desta área, não é proposto um modelo diferente dos já conhecidos (seção 1.5.2), mas sim, são elaboradas especificações de processos conhecidos para incorporar as práticas específicas para o desenvolvimento de groupware.

Há alguns poucos processos para o desenvolvimento específico de groupware, dentre eles: o processo em cascata para desenvolvimento de groupware proposto por Dewan (2001); o processo incremental SER (Fischer, Grudin *et al.*, 2001); e o processo centrado na participação do usuário OSDP (Schümmer *et al.*, 2005).

Não há um processo que possa ser considerado ideal e não se pode demonstrar que um processo é sempre melhor que outro (Sommerville 2004; Pressman 2004). Cada processo incorpora “boas práticas” que o fazem adequado para determinados tipos de projetos. O processo RUP-3C-Groupware, proposto nesta tese (capítulo 3), mostra-se adequado especificamente para o desenvolvimento evolucionário de groupware baseado em componentes e orientado ao reuso, sendo feito uso do Modelo 3C de Colaboração.

1.6.
Organização da escrita desta tese

O processo proposto nesta tese é resultado de oito anos de experiência com o desenvolvimento dos serviços do ambiente AulaNet, e mais especificamente, dos cinco anos de pesquisa e desenvolvimento de versões do Mediated Chat. As boas práticas aprendidas ao longo destas experiências foram incorporadas no processo: uso do Modelo 3C de Colaboração para guiar o desenvolvimento de groupware, desenvolvimento evolucionário focando um problema por versão num

processo investigativo, e desenvolvimento baseado em componentes e orientado ao reuso. O desenvolvimento das versões do Mediated Chat, incluindo a generalização das práticas aprendidas com este desenvolvimento, é discutido no Capítulo 2.

O processo aqui proposto, denominado RUP-3C-Groupware consiste na extensão do RUP, Rational Unified Process, na qual foram incorporadas as boas práticas aprendidas. No Capítulo 3 é apresentada uma visão geral do RUP, e são apresentados os fluxos de atividades e os artefatos estendidos ou elaborados para o RUP-3C-Groupware.

Para investigar o processo proposto, foi realizado um Estudo de Caso onde alunos de Engenheira de Software executaram algumas das atividades produzindo alguns artefatos-chave. Como discutido no Capítulo 4, do estudo de caso foram obtidos indícios sobre a repetitividade do processo e de sua adequação para o desenvolvimento de groupware.

No Capítulo 5 são apresentados conclusão e trabalhos futuros. As referências encontram-se no Capítulo 6. E o Anexo 1 contém o documento usado no estudo de caso.