UNIVERSIDADE FEDERAL DE PELOTAS
INSTITUTO DE FÍSICA E MATEMÁTICA
DEPARTAMENTO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MODELAGEM DE UM AMBIENTE DE GERENCIAMENTO DE DESENVOLVIMENTO DISTRIBUÍDO DE *SOFTWARE* BASEADO EM *WORKFLOW*

JULIANO MACHADO TEIXEIRA

Pelotas, 2007

JULIANO MACHADO TEIXEIRA

MODELAGEM DE UM AMBIENTE DE GERENCIAMENTO DE DESENVOLVIMENTO DISTRIBUÍDO DE *SOFTWARE* BASEADO EM *WORKFLOW*

Trabalho acadêmico apresentado ao Departamento de Informática do Instituto de Física e Matemática da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

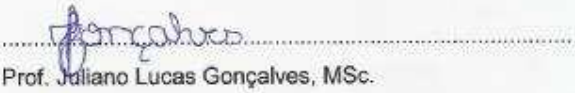
Orientadora: Prof[a] Msc. Flávia Braga de Azambuja
Co-Orientador: Hugo Vares Vieira

Pelotas, 2007

BANCA EXAMINADORA:

Profª. Flávia Braga de Azambuja, MSc. (orientadora)

Profª. Eliane Alcoforado Diniz, Msc.



Prof. Juliano Lucas Gonçalves, MSc.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me deu a vida e sempre me ajudou a escolher os caminhos certos nos momentos de incerteza. Obrigado Senhor por estar sempre ao meu lado, principalmente nos momentos difíceis.

Agradeço também a minha família, principalmente ao meu pai Mauro e minha mãe Adalvina que foram a base para que eu alcançasse todos os meus objetivos, sempre me ensinando a ser uma pessoa de bem. Precisaria de muito espaço neste trabalho para manifestar todo o meu agradecimento a vocês.

Aos meus irmãos pela paciência durante o período de realização deste trabalho, em especial ao Marcelo por nunca ter reclamado quando eu o tirava do computador. À Luciana peço desculpas por todas as vezes que disse que iria concertar seu computador e não encontrava tempo. E ao Otávio, por não poder jogar na *lan house* como prometi muitas vezes.

A minha namorada Neizy que tanto amo, agradeço do fundo do meu coração por todos esses anos de companheirismo e pela paciência, não apenas durante a conclusão deste trabalho, mas desde o início do curso, quando muitas vezes não dava atenção necessária que ela tanto merece. Amor eu te amo.

Agradeço aos meus orientadores: a professora Flávia pela orientação e apoio no desenvolvimento deste trabalho; e ao mestrando da Pontifícia Universidade Católica de Porto Alegre Hugo Vieira, pela dedicação e paciência explicando e esclarecendo todas as minhas dúvidas.

A todos os colegas e professores, que foram muito mais do que colegas e professores. Cito em especial: os colegas Mateus e Lucas São Risal pelas conversas no MSN durante a conclusão deste trabalho; o Cléber Alemão e sua impressora; e o Rodrigo pela ajuda de última hora. Com certeza as amizades construídas durante o tempo de faculdade guardarei comigo para sempre.

A todas as pessoas que direta ou indiretamente contribuíram de alguma forma para que fosse possível a conclusão desta etapa. E aquelas que eu não pude citar aqui, o meu muito obrigado.

“Até a pé nós iremos

para o que der e vier

mas o certo é que nós estaremos

com o Grêmio onde o Grêmio estiver...”

RESUMO

TEIXEIRA, Juliano Machado. **Modelagem de um Ambiente de Desenvolvimento Distribuído de *Software* baseado em *Workflow*.** 2007. 88f. Trabalho acadêmico (Graduação em Ciência da Computação) – Instituto de Física e Matemática. Universidade Federal de Pelotas, Pelotas.

A prática de desenvolvimento de *software* de maneira distribuída está avançando cada vez mais à medida que as organizações começam a buscar os melhores recursos globais para composição de suas equipes de desenvolvimento. Porém, esta distribuição afeta diretamente a gerência e o controle das atividades devido a diversos fatores, tais como: diferença de fuso horário; cultura; e a falta de confiança entre as equipes. Portanto, uma grande necessidade de controle do fluxo de trabalho entre estas equipes tem se apresentado como uma lacuna a ser preenchida pela comunidade acadêmica, em busca de uma automatização dos processos, a fim de garantir um processo cada vez mais sólido e um produto de qualidade para o cliente. Uma tecnologia que vem se destacando na automatização de processos organizacionais é a tecnologia de *workflow*. O uso do *workflow* possibilita a automatização de tarefas repetitivas, gerando um ganho de desempenho e produtividade. Baseado nisso, este trabalho de conclusão de curso tem como objetivo principal a modelagem de um ambiente de desenvolvimento distribuído de *software*, baseado na tecnologia de *workflow*.

Palavras Chave: *Workflow*. Desenvolvimento distribuído de *software*. Processo de desenvolvimento de *software*.

ABSTRACT

TEIXEIRA, Juliano Machado. **Modelagem de um Ambiente de Desenvolvimento Distribuído de *Software* baseado em *Workflow*.** 2007. 88f. Trabalho acadêmico (Graduação em Ciência da Computação) – Instituto de Física e Matemática. Universidade Federal de Pelotas, Pelotas.

Distributed software development is a growing practice once the organizations start seeking for better global resources in order to compose their development teams. However, this new environment affects directly the tasks management and control due to several aspects, like: different time zones, different cultures and the lack of trust among teams. Therefore, there is a great need of workflow control among those teams, and that emerges as a puzzle that needs to be solved by the academic community, reaching for a process automation that can assure a solid process and a quality product for the client. The workflow technology has gained acceptance on the organizational process automation. Its use allows the automation of repetitive tasks, providing performance and productivity gains. Based on the previously discussed, this work has as its main goal modeling a distributed software development environment based on the workflow technology.

Keywords: Workflow. Distributed software development. Software development process.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CSCW - *Computer Supported Cooperative Work*
DDS - Desenvolvimento Distribuído de *Software*
GSD - *Global Software Development*
HTML - *HyperText Markup Language*
IEC - *International Electrotechnical Commission*
ISO - *International Standards Organization*
PDS - Processo de Desenvolvimento de *Software*
RUP - *Rational Unified Process*
UML - *Unified Modeling Language*
WAPI - *Workflow Application Programming Interface*
WES - *Workflow Enactment Service*
WfMS - *Workflow Management System*

SUMÁRIO

1 INTRODUÇÃO

No atual momento da era da informação a tecnologia tem avançado em direção à globalização dos negócios, em especial na área de desenvolvimento de *software*. O *software* tornou-se um componente vital para quase todos os negócios. Neste sentido, para as organizações que buscam o sucesso, é clara a necessidade do uso da tecnologia da informação como um diferencial competitivo (HERBSLEB; MOITRA, 2001). Em busca dos melhores recursos globais, diversas empresas têm adotado o desenvolvimento distribuído de *software* - DDS, a fim de obter um produto com melhor qualidade. O DDS tem sido caracterizado principalmente pela colaboração e cooperação entre organizações internacionais e pela criação de grupos de desenvolvedores que trabalham juntos, mas localizados em áreas geograficamente distintas (KIEL, 2003). Existem diversas vantagens no desenvolvimento distribuído, porém, por outro lado, o desenvolvimento colaborativo de *software* apresenta diversos problemas de coordenação e comunicação, principalmente quando as equipes estão distribuídas geograficamente. Os ambientes atuais de desenvolvimento de *software* não mostram muita informação, levando os colaboradores a usar frequentemente ferramentas baseadas em texto para determinar o que está acontecendo no grupo (GUTWIN; PENNER; SCHNEIDER, 2004). Identifica-se, portanto, a necessidade de automatizar o fluxo de trabalho de uma equipe de desenvolvimento, com o intuito de facilitar o desenvolvimento e o gerenciamento de uma equipe distribuída.

Uma tecnologia que atingiu uma grande importância no controle do fluxo de trabalho para grupos de pesquisa e grandes empresas com instalações distribuídas, é a tecnologia de *workflow*, principalmente por oferecer um controle automatizado do andamento das atividades (TESSMANN, 2005). Com o uso de um sistema de *workflow*, podem-se definir quais atividades devem ser realizadas em paralelo e quais dependem da etapa anterior, permitindo um maior controle e um gerenciamento sobre cada etapa de desenvolvimento. O estudo realizado neste trabalho tem como objetivo principal a modelagem de um ambiente de desenvolvimento distribuído de *software*, utilizando conceitos de *workflow*.

1.2 Estrutura do trabalho

Este trabalho contém sete capítulos, sendo o primeiro referente à introdução. O restante do documento está estruturado da seguinte forma: o capítulo 2 contém informações sobre os principais processos de desenvolvimento de *software*, bem como uma explicação do que são, para que servem, e qual a importância de um processo de desenvolvimento bem definido dentro de uma organização; o capítulo 3 trata do desenvolvimento distribuído de *software*. Quais as vantagens, desvantagens e os maiores desafios encontrados na gerência dos processos; o capítulo 4 contém todas as informações referentes à tecnologia de *workflow*. Detalha os conceitos, os tipos e os componentes de um *workflow*; o capítulo 5 trata da modelagem do sistema proposto. Apresenta diversas figuras representando a modelagem do fluxo de trabalho de uma equipe de desenvolvimento distribuída de *software*; o capítulo 6 apresenta a interface do sistema, e tem como finalidade principal mostrar como seria a interação entre o usuário e o sistema através do modelo proposto; por fim, o capítulo 7 apresenta as conclusões deste trabalho, qual a importância de um modelo de *workflow* no auxílio ao desenvolvimento distribuído de *software*, algumas das dificuldades encontradas para o desenvolvimento deste trabalho e os possíveis trabalhos futuros que poderão ser desenvolvidos a partir deste trabalho de conclusão de curso. Logo em seguida são apresentadas às referências utilizadas para o desenvolvimento deste trabalho.

2 PROCESSOS DE DESENVOLVIMENTO DE *SOFTWARE*

O desenvolvimento de *software* através de um processo de desenvolvimento bem definido não é uma tarefa simples. Com o aumento da demanda de *software* das empresas, esse desenvolvimento tem se tornado uma tarefa cada vez mais complexa (PRIKLADNICKI; AUDY, 2004). Um processo de desenvolvimento de *software* - PDS pode ser caracterizado como um conjunto completo de atividades necessárias para transformar requisitos de usuários em produtos de *software* (BOOCH; JACOBSON; RUMBAUGH, 2001). Podemos dizer que um PDS, começa com um problema que um usuário quer resolver e acaba com um sistema que resolve este problema. Este processo pode envolver o desenvolvimento *de software* desde o início, embora, cada vez mais, ocorra o caso de um *software* novo ser desenvolvido mediante a expansão e a modificação de sistemas já existentes (SOMMERVILLE, 2003). Como as organizações geralmente consideram o desenvolvimento de um produto de *software* como um projeto, podemos então considerar um processo, sendo uma seqüência de passos que um projeto pode seguir para desempenhar alguma tarefa. Pressman (2001) diz que o PDS é a estrutura para as tarefas que são necessárias à construção de *software* com alta qualidade. Ainda, segundo Fuggetta (2000), o PDS pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos que são necessários para compreender, desenvolver e manter um produto de software. A definição de um PDS envolve várias informações, como recursos utilizados, atividades a ser desempenhadas, artefatos consumidos e gerados, dentre outras (ROCHA et al., 1996). Os principais conceitos, segundo Derniame, Kaba e Warboys (1999), ligados à sua modelagem são: **atividade**: etapa do processo que visa gerar ou modificar um conjunto de artefatos, incorporando e executando procedimentos, regras e políticas organizacionais. É possível que algumas atividades sejam decompostas em subatividades, embora isto não seja obrigatório; **artefato**: informação desenvolvida e mantida em um projeto de *software*. É possível decompor artefatos em subartefatos, embora isso não seja obrigatório; **direção**: que são procedimentos, regras e políticas organizacionais que dirigem

atividades e geralmente estão estruturados na forma de manuais; **recurso**: um fator necessário na execução de uma atividade. Os recursos devem ser divididos em: executores (os agentes humanos do processo) e ferramentas (agentes computadorizados), usados tradicionalmente no desenvolvimento de *software*.

Uma informação importante ligada à definição de PDS, é o que se deseja alcançar a partir de sua utilização. Pensando nisso, Tyrrel (2000) definiu um conjunto de objetivos que os conjuntos de *software* devem ter: **efetividade**: os PDS devem ajudar a determinar as necessidades do cliente e verificar se o que foi produzido satisfaz o cliente; **manutenibilidade**: o PDS deve ser capaz de expor a maneira de pensar de projetistas e programadores de forma que suas intenções sejam claras. Assim, torna-se fácil encontrar e reparar falhas no produto; **previsibilidade**: o PDS deve ajudar a predizer quanto tempo será necessário para o desenvolvimento de cada parte do produto; **repetível**: produzir um processo novo para cada projeto implica em grandes gastos para a organização. Desta forma, é importante que o PDS seja criado de forma a poder ser reutilizado em vários projetos; **qualidade**: o PDS deve permitir engenheiros de software assegurar um produto de qualidade elevada. Para isso, o processo deve fornecer uma ligação entre os desejos de um cliente e o produto de um desenvolvedor; **melhoria**: o PDS deve ser constantemente melhorado. Assim, o próprio processo deve permitir a identificação e pontos de melhoria; **rastreabilidade**: o PDS deve permitir que a gerência, os desenvolvedores e o cliente sigam o *status* do projeto.

Embora, existam conceitos e objetivos bem definidos para processos de *software*, estes podem apresentar grande complexidade e possibilitar diversas alternativas de execução de suas atividades (MACHADO, 2000). Assim, é importante para as organizações padronizar o PDS, a fim de garantir uma maior qualidade dos produtos de *software*.

2.1 Processo padrão de desenvolvimento de *software*

Um processo padrão de desenvolvimento de *software* descreve as atividades que devem ser realizadas no desenvolvimento de sistemas em todos os projetos de uma organização. O processo padrão é a definição operacional do processo básico que guia o estabelecimento de um processo comum em uma organização (BORGES; FALBO, 2002). Na definição de um processo padrão para

uma organização, é necessário considerar as características da organização (OLIVEIRA, 1999). Estas características envolvem: atividades realizadas nos projetos de *software* da organização não provenientes de um modelo proposto na literatura, que são consideradas atividades próprias da organização; o fato de a organização desenvolver *software* para uso próprio, para comercialização ou sob encomenda; o fato de a organização ser especializada no desenvolvimento de um tipo de *software* específico; os principais problemas enfrentados pela organização em seus projetos de *software*, dentre outras. Baseado nestes conceitos, o foco principal deste trabalho de conclusão de curso consiste no desenvolvimento de *software* comercial, desenvolvido por organizações que trabalham de forma cooperativa, com suas equipes localizadas em locais distintos, normalmente longe do cliente a quem o produto final deve ser entregue.

Para Ginsberg e Quinn (1995) o processo padrão de desenvolvimento de *software* é o meio pelo qual a organização expressa os requisitos que todos os processos de desenvolvimento de *software* devem atender. Sua implantação apresenta vantagens como: a redução de problemas relacionados a treinamentos, revisões e suporte de ferramentas; experiências adquiridas em cada projeto podem ser incorporadas ao processo padrão, contribuindo para sua melhora; maior facilidade em medições de processo e qualidade; maior facilidade de comunicação entre os membros da equipe de projeto; melhor desempenho, previsibilidade e confiabilidade dos processos de trabalho. Drouin et al. (1998) diz ainda que o processo padrão descreve os elementos fundamentais que devem ser incorporados em qualquer processo definido na organização e as relações entre estes elementos, como seqüências e interfaces. Ainda segundo os autores, a definição de um processo padrão deve considerar os objetivos do processo; identificar as atividades, papéis e responsabilidades; definir as entradas e saídas, os pontos de controle e os registros de qualidade e identificar interfaces internas e externas. Oliveira (1999) destaca alguns pontos relevantes em se tratando de processo padrão: a cultura organizacional deve ser considerada assim como o modo que ela desenvolve *software* e os problemas cruciais enfrentados em seu projeto de desenvolvimento, contemplando-os no processo; como diferentes tipos de *software* podem ser desenvolvidos para um mesmo domínio e em uma mesma organização, é necessário definir diversos PDS adequados para a construção dos diferentes tipos de *software*; as diversas características do projeto, como requisitos, tamanho,

número de pessoas e partes envolvidas, também devem ser consideradas em um PDS para um projeto específico; definir diversos processos de *software* para uma mesma organização pode ser muito trabalhoso e ter como resultado processos completamente distintos, o que não é adequado. Pesquisas mostram a necessidade de padronização dos processos de *software* dentro de uma organização (DROUIN et. al., 1998).

Considerando a importância da utilização de um processo padrão, um considerável esforço tem sido feito para sua modelagem e, como forma de auxiliar a sua criação, diversos processos estão surgindo. Eles podem ser adotados por completo ou permitir ainda serem adaptados de acordo com algumas características da organização. É possível também que uma organização baseie-se em mais de um processo para a definição de seu processo padrão. O item a seguir explica o mecanismo de adaptação do processo padrão.

2.2 Adaptação do processo padrão

A cada novo projeto em uma organização, o processo padrão de desenvolvimento de *software* deve ser utilizado. Entretanto este processo padrão não pode servir a qualquer tipo de projeto. As características do processo variam conforme o porte da empresa, os objetivos de projetos específicos, os recursos disponíveis, as tecnologias de desenvolvimento, o conhecimento e a experiência da equipe. Um processo padrão de adaptação consiste na modificação, exclusão ou ainda, na adição de novos elementos ao processo padrão. Segundo Fitzgerald, O'kane e Russo (2003), é mundialmente aceito que um processo padrão de desenvolvimento de *software* deva sofrer adaptações para atender a necessidade no contexto dos projetos. Para Ginsberg e Quinn (1995), o processo de adaptação deve gerar um processo de desenvolvimento específico para o projeto cada vez que executado. Ainda, de acordo com Kellner (1996), o processo de adaptação pode também ser considerado como uma atividade de reuso do processo padrão. O processo de adaptação do processo padrão é um importante requisito para as organizações de software. Segundo ISO/IEC (1995 e 1998) as atividades necessárias para adaptar um processo padrão são: identificar o ambiente do projeto, verificando quais são as características do projeto que vão influenciar o processo de adaptação. Como exemplos destas características têm-se: as **políticas**

organizacionais, como procedimentos, estratégias, tamanho, criticidade da aplicação, número de pessoas envolvidas, etc; **solicitar entradas**, ou seja, envolver as entradas da organização que vão ser afetadas pelas decisões do processo de adaptação. Podem ser considerados como entradas os usuários, pessoal de suporte, contratantes e licitantes; **selecionar processos, atividades e tarefas,** ou seja, decidir quais processos, atividades e tarefas vão ser desempenhados no projeto. Isso inclui qual documentação será desenvolvida e a distribuição de responsabilidades. Para a seleção de processos, atividades e tarefas adequadas ao projeto, é importante levar em consideração as informações coletadas anteriormente sobre ambiente do projeto e entradas da organização. É permitido que atividades sejam adicionadas ou apagadas desde que as mesmas sejam especificadas no contrato do projeto; e **documentar** as decisões do processo de adaptação no contrato do projeto. Ainda é importante que a conformidade com o processo padrão seja mantida. Para isso, os processos, atividades, e tarefas selecionadas por meio do processo de adaptação para um projeto, devem ser executados conforme documentado em contrato.

Todo o desenvolvimento de *software* pode ser caracterizado como um ciclo de solução de problemas, onde são encontrados quatro estágios distintos: situação atual, definição do problema, desenvolvimento técnico e integração da solução. As fases genéricas que caracterizam o processo de *software* são aplicáveis a todo *software.* O problema está em selecionar o modelo do processo que melhor se adapte ao *software* a ser trabalhado por uma equipe de desenvolvimento. Existem diversos modelos de processo para engenharia de *software,* todos caracterizados de forma a ajudar no controle e na coordenação de um projeto de *software* real (PRESSMANN, 2001). O item a seguir apresenta alguns modelos de processo de *software*.

2.3 Modelos de processos de *software*

Segundo Sommerville (2003), um modelo de processo de *software* é uma representação abstrata de um processo de *software*. Cada modelo de processo representa um processo a partir de uma perspectiva particular, de uma maneira que proporciona apenas informações parciais sobre o processo. A seguir, serão mostrados como exemplo, alguns modelos genéricos que, segundo o autor, não são

descrições definitivas de processos de *software*. Em vez disso, são abstrações úteis, que podem ser utilizadas para explicar diferentes abordagens do desenvolvimento de *software*. Para muitos sistemas de grande porte, não existe apenas um processo de *software* que possa ser utilizado. Processos diferentes são utilizados para desenvolver diferentes partes do sistema. Os modelos de processo segundo Sommerville (2003) são:

O **Modelo em Cascata**, representado na Fig. 1: considera as atividades de especificação, desenvolvimento e evolução, que são fundamentais ao processo, e as representa como fases separadas do processo, como a especificação de requisitos, o projeto de *software*, a implementação, os testes e assim por diante.

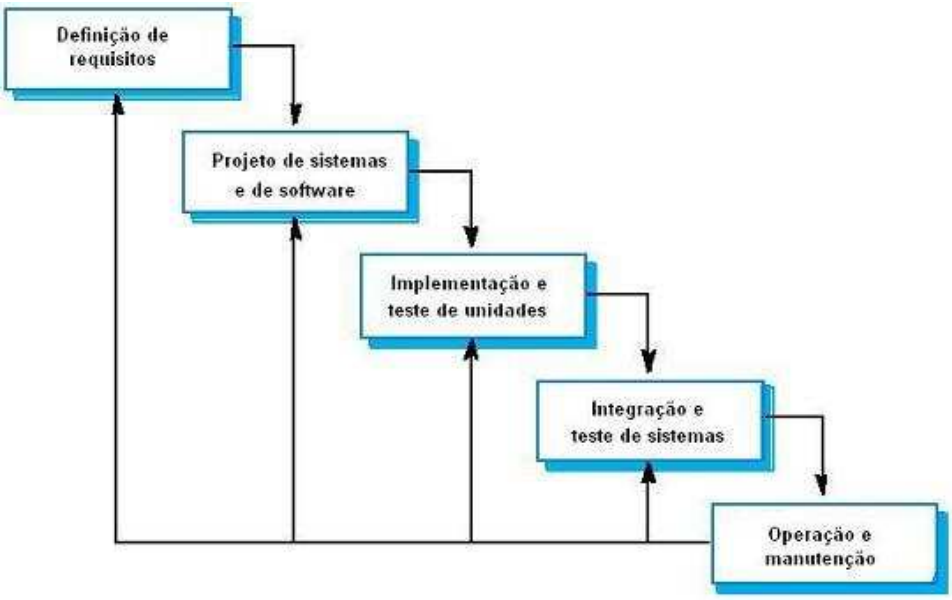


Figura 1: Modelo em cascata.
Fonte: SOMMERVILLE, 2003, p. 38.

O **Desenvolvimento Evolucionário**: abordagem que intercala as atividades de especificação, desenvolvimento e validação. Um sistema inicial é rapidamente desenvolvido a partir de especificações abstratas, que são então refinadas com informações do cliente, para produzir um sistema que satisfaça suas necessidades. A abordagem do desenvolvimento evolucionário é ilustrada na Fig. 2.

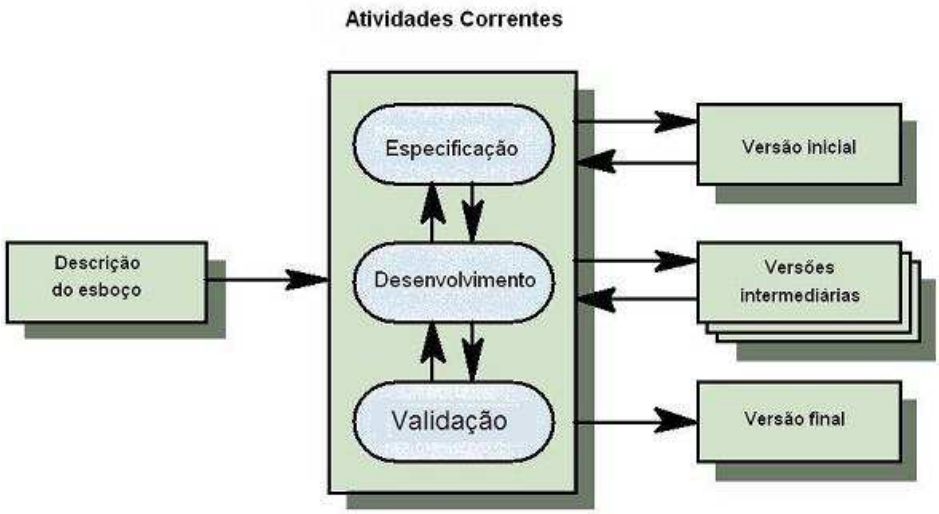


Figura 2: Desenvolvimento evolucionário.
Fonte: SOMMERVILLE, 2003, p. 39.

O **Desenvolvimento Formal de Sistemas** que se baseia na produção de uma especialização formal matemática do sistema e na transformação dessa especificação, utilizando-se métodos matemáticos, para construir um programa. A verificação de componentes do sistema é realizada mediante argumentos matemáticos, mostrando que eles atendem a suas especificações, conforme a Fig. 3.

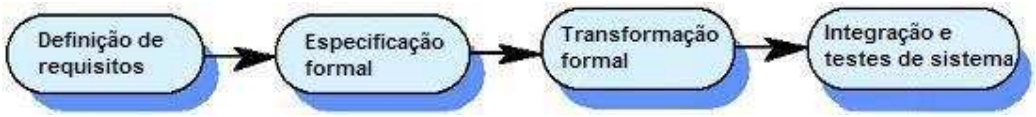


Figura 3: Desenvolvimento formal de sistemas.
Fonte: SOMMERVILLE, 2003, p. 40.

O **Desenvolvimento Orientado a Reuso**, ilustrado na Fig. 4, que tem como base a existência de um número significativo de componentes reusáveis. O processo de desenvolvimento de sistemas se concentra na integração destes componentes em um sistema, em vez de proceder ao desenvolvimento a partir do zero.

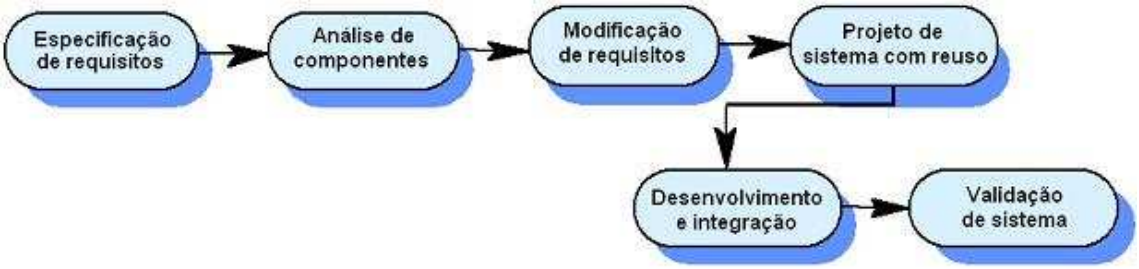


Figura 4: Desenvolvimento orientado a reuso.
Fonte: SOMMERVILLE, 2003, p. 42.

Os processos baseados no modelo em cascata e no modelo de desenvolvimento evolucionário são bastante utilizados para o desenvolvimento de sistemas práticos. Já o desenvolvimento formal de sistemas foi utilizado com sucesso em muitos projetos, mas ainda é utilizado em poucas organizações. O reuso informal é comum em muitos processos, visto que o desenvolvimento de sistemas a partir de componentes reutilizáveis é essencial para o rápido desenvolvimento de *software* (SOMMERVILLE, 2003). A seguir será detalhado o desenvolvimento orientado a reuso, visto que será um dos processos em que se baseia a modelagem do ambiente de desenvolvimento distribuído proposto.

2.3.1 Desenvolvimento orientado a reuso

Na maioria dos projetos de *software* é comum a utilização de trechos de códigos que já foram utilizados por outros sistemas. Isso, em geral, acontece informalmente quando as pessoas envolvidas no desenvolvimento de *software* conhecem projetos ou códigos similares àquele exigido (SOMMERVILLE, 2003). Eles recorrem a estes produtos, fazem as modificações necessárias e as incorporam no novo projeto. O modelo genérico de processo para o desenvolvimento orientado a reuso é mostrado na Fig. 4. Sommerville (2003) mostra que, embora o estágio inicial de especificação de requisitos e o estágio de validação sejam comparáveis com outros processos, os estágios intermediários em um processo orientado a reuso são diferentes. Esses estágios são: **análise de componentes**: considerando a especificação de requisitos, é feita uma busca de componentes para implementar esta especificação. Em geral não existe uma combinação exata e os componentes que podem ser utilizados fornecem somente parte da funcionalidade requerida;

modificação de requisitos: nesse estágio, os requisitos são analisados, utilizando-se as informações sobre os componentes que foram encontrados. Eles são então modificados para refletir os componentes disponíveis. Quando não forem possíveis as modificações, a atividade de análise de componentes poderá ser refeita, a fim de procurar soluções alternativas; **projeto de sistema com reuso**: durante essa fase, a infra-estrutura do sistema é projetada ou uma infra-estrutura existente é reutilizada. Os projetistas levam em conta os componentes que são reutilizados e organizam a infra-estrutura para lidar com esse aspecto. Um novo *software* poderá ter de ser projetado, se os componentes reutilizáveis não estiverem disponíveis; **desenvolvimento e integração**: o *software* que não puder ser comprado será desenvolvido, e os componentes serão integrados, a fim de criar um sistema. A integração de sistemas, neste modelo, pode ser parte do processo de desenvolvimento, em vez de uma atividade separada.

A vantagem do modelo orientado a reuso é a redução da quantidade de *software* a ser desenvolvido, e consequentemente, a redução de custos e riscos. Geralmente, ele também pode propiciar a entrega mais rápida do *software*. Porém, as adequações nos requisitos são inevitáveis, e isto pode resultar em um sistema que não atenda as reais necessidades dos usuários. Além disso, perde-se algum controle sobre a evolução do sistema, visto que novas versões dos componentes reutilizáveis não estão sob o controle da organização que utiliza estes componentes (SOMMERVILLE, 2003). No item que segue serão observadas algumas características de um dos modelos de processo de *software* mais utilizados atualmente, que juntamente com o processo orientado a reuso, formam o processo a ser utilizado na modelagem do ambiente de desenvolvimento distribuído proposto.

2.4 *Rational Unified Process* - RUP

O RUP é um processo cuja base é um meta-modelo que permite definir a linguagem que descreve o processo. É um sistema iterativo, que se adapta perfeitamente às exigências dos sistemas atuais que estão cada vez mais complexos e sofisticados. Embora não exista um único processo adequado para todas as empresas de desenvolvimento de *software*, uma das grandes vantagens do RUP é o fato dele ser extensível, podendo ser ajustado e redimensionado para

atender as necessidades de projetos que variam desde pequenas equipes até grandes empresas de desenvolvimento de *software* (BOOCH, 1998).

Segundo Kruchten (2000), seu processo é formado basicamente por quatro elementos primários de modelagem: **papéis**: quem; **atividades**: como; **artefatos**: o quê; e **fluxos**: quando. O papel é o conceito central no processo RUP. Um papel define as responsabilidades e o comportamento de um indivíduo ou grupo de indivíduos que trabalham juntos como uma equipe. São responsáveis também pelo desempenho de atividades do processo e cada papel pode ter responsabilidades sobre certos artefatos do processo. Uma atividade é uma unidade de trabalho que produz um resultado significante no contexto do projeto. A atividade tem um propósito bem definido, normalmente expresso em termos de criar ou modificar artefatos, como um plano, um modelo ou uma classe. Toda atividade deve ser atribuída a um papel. Um artefato é um pedaço de informação que é produzida, modificada ou usada por um processo. Os artefatos são usados como entradas para desempenhar uma atividade e são o resultado ou a saída de tais atividades. No RUP, um artefato pode ser composto de outros artefatos. Por exemplo, o modelo de caso de uso possui vários casos de uso, o modelo de projeto várias classes, o plano de desenvolvimento de *software* contém muitos outros planos, etc. Deve-se ainda considerar que os artefatos podem ter várias formas e formatos: modelos, elemento de modelo, documento, código fonte e executáveis. Os fluxos são seqüências de atividades que produzem um resultado de valor observável. Um fluxo pode ser expresso como um diagrama de seqüência, colaboração ou de atividade. Existem dois tipos de fluxos no RUP: fluxos centrais e detalhes de fluxo. Entende-se por fluxos centrais as disciplinas do processo, e representam uma partição de todos os trabalhadores e atividades em áreas de interesse. Cada disciplina do RUP é decomposta em detalhes de fluxo que nada mais são do que o agrupamento de atividades relacionadas em um fluxo de informação. Assim, uma disciplina apresenta vários detalhes de fluxo mostrando como as atividades interagem com os artefatos.

Os elementos mencionados acima são os principais elementos do RUP, mas deve-se considerar que seu desenvolvimento é realizado em fases que constituem um ciclo de vida, sendo estes importantes elementos do processo (PEREIRA, 2005). Estas fases do RUP são: a fase de **iniciação**; **elaboração**; **construção**; e **implantação**. O modelo de fases e disciplinas do RUP está representado na Fig. 5.

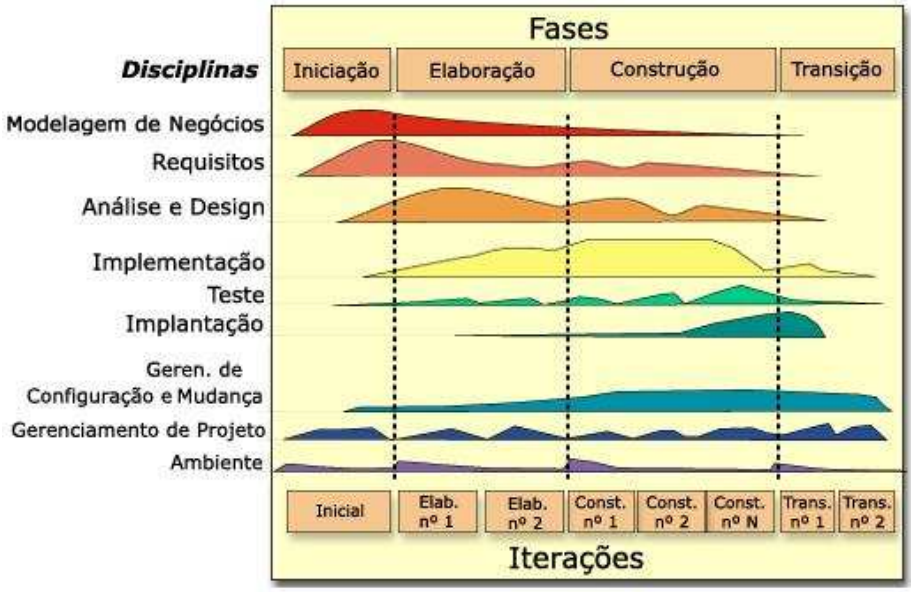


Figura 5: Visão geral das disciplinas e fases do RUP

Fonte: PEREIRA, 2005, p. 35.

A evolução da equipe do projeto de *software* e o percentual de trabalho realizado durante estas fases estão representados na Fig. 6. Percebe-se que na fase de iniciação a maior parte do trabalho está no gerenciamento e levantamento de requisitos do sistema. Na fase de elaboração, é dada uma maior ênfase na parte arquitetural do sistema a ser desenvolvido. Já na fase de construção, que é a fase mais duradoura do desenvolvimento, a maior parte do serviço está focada na codificação e desenvolvimento do *software*. E, por fim, na fase de transição, a maior parte do trabalho está na instalação do sistema junto à comunidade de usuários e na avaliação do *software*.

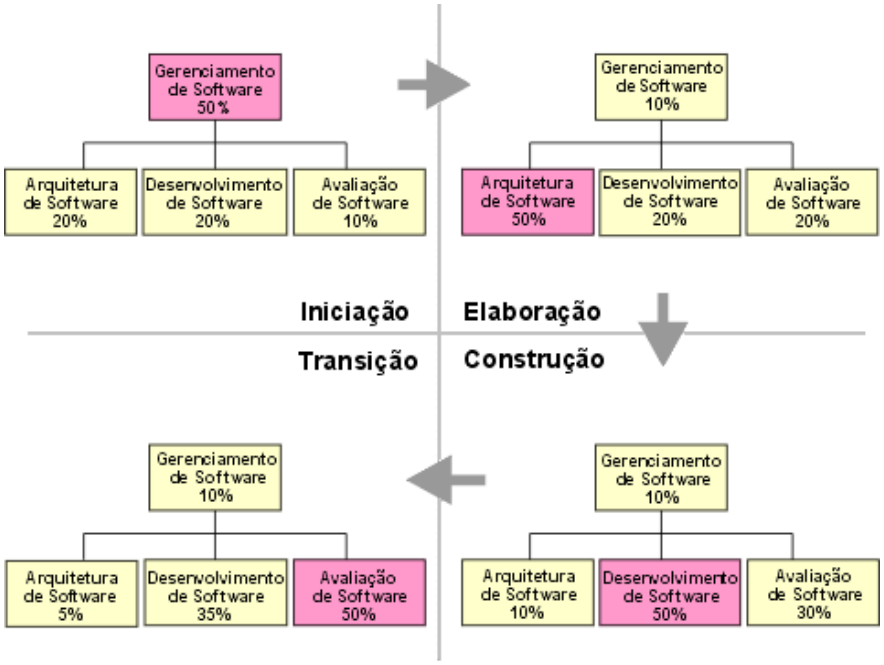


Figura 6: Evolução da equipe do projeto de *software* além do ciclo de vida

Fonte: http://www.rational.com.

De acordo com a especificação do RUP, cada uma destas fases é concluída por um marco principal, sendo cada fase um intervalo entre dois marcos principais (Fig. 7). Estes marcos têm como finalidade definir se o projeto está em condições de prosseguir e avançar a próxima fase, ou se deve ser totalmente repensado antes de prosseguir. Cada um desses marcos será explicado nas seções seguintes.

Figura 7: As fases e os marcos de um projeto

Fonte: http://www.rational.com.

2.4.1 Marco dos objetivos do ciclo de vida

Este marco está localizado entre as fases de iniciação e elaboração. Neste momento são analisados os objetivos do ciclo de vida do projeto e, então, decidido se o projeto deve prosseguir ou ser cancelado. Os critérios de avaliação são: consentimento dos envolvidos sobre a definição do escopo e as estimativas de custo/programação; consenso de que o conjunto correto de requisitos foi capturado; consenso de que as estimativas de custo/programação, as prioridades, os riscos e o processo de desenvolvimento são adequados; certeza de que todos os riscos foram identificados e existe uma estratégia atenuante para cada um. Caso o projeto não atinja este marco, ele poderá ser completamente repensado ou até mesmo anulado. Se todos os requisitos forem aceitos e os critérios de avaliação são satisfatórios, o processo deve prosseguir e passar para a fase seguinte de elaboração.

2.4.2 Marco da arquitetura do ciclo de vida

Neste marco, localizado entre as fases de elaboração e construção, são examinados os objetivos e o escopo detalhados do sistema, a opção de arquitetura e a resolução dos principais riscos. Os critérios de avaliação a serem verificados são: se a arquitetura é estável; se as principais abordagens a serem usadas no teste e na avaliação foram comprovadas; se o teste e a avaliação de protótipos executáveis demonstraram que os principais elementos de risco foram tratados e resolvidos com credibilidade; se os planos de iteração para a fase de construção têm detalhes e fidelidade suficientes para permitir o avanço do trabalho; se os planos de iteração para a fase de construção são garantidos por estimativas confiáveis; se houve consenso entre todos os envolvidos de que a visão atual do projeto poderá ser atendida se o plano atual for executado para desenvolver o sistema completo, no contexto da arquitetura atual; e a real despesa comparando-se com a despesa planejada com recursos é aceitável. Caso o projeto não atinja este marco ele poderá ser abortado ou completamente repensado. Se todos os requisitos forem satisfeitos, o projeto está pronto para a próxima fase de construção.

2.4.3 Marco da capacidade operacional inicial

Neste marco, localizado no final da fase de construção, o produto está pronto para ser passado para a equipe de transição (responsável pela implantação do sistema junto aos usuários). Toda a funcionalidade foi desenvolvida e todos os testes foram concluídos. Os critérios que devem ser avaliados são: se o produto é estável e foi desenvolvido o suficiente para ser implantado junto aos usuários, se todos os envolvidos estão prontos para a transição para a comunidade de usuários e as despesas reais com recursos ainda são aceitáveis se comparadas com as planejadas. Se estes critérios forem satisfeitos, o produto está pronto para ser entregue ao cliente.

2.4.4 Marco de lançamento do produto

No fim da fase de transição está o quarto marco mais importante do projeto. Neste momento, é decidido se os objetivos foram atendidos, e se deve ser iniciado outro ciclo de desenvolvimento. Os critérios de avaliação são: verificar se o usuário está satisfeito com o produto que foi entregue, e se as despesas reais com recursos são aceitáveis se comparadas com as planejadas. Neste marco o produto está em produção e o ciclo de manutenção posterior inicia. Isso pode envolver o início de um novo ciclo ou de algum *release* de manutenção adicional.

3 DESENVOLVIMENTO DISTRIBUÍDO DE *SOFTWARE*

No atual momento da tecnologia da informação, percebe-se um grande avanço em direção a globalização dos negócios, em particular nos negócios relacionados com um intenso investimento na tecnologia de desenvolvimento de *software*. As forças econômicas têm transformado os mercados nacionais em mercados globais e gerado novas formas de competição e cooperação. Estas mudanças estão tendo um impacto profundo, não somente no mercado e na distribuição, mas também na maneira como os produtos são projetados, construídos, testados, e entregues ao cliente. O DDS tem sido caracterizado principalmente pela colaboração e cooperação entre departamentos de organizações internacionais e pela criação de pequenos grupos de desenvolvedores que trabalham em conjunto, mas estão localizados em cidades ou países diferentes (KIEL, 2003). As Fig. 8 e 9 ilustram a mudança para este novo cenário.

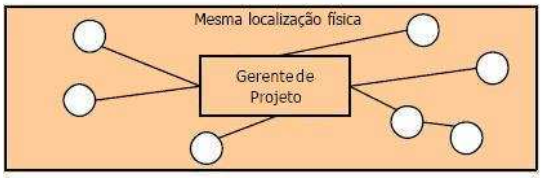


Figura 8 – Equipe centralizada

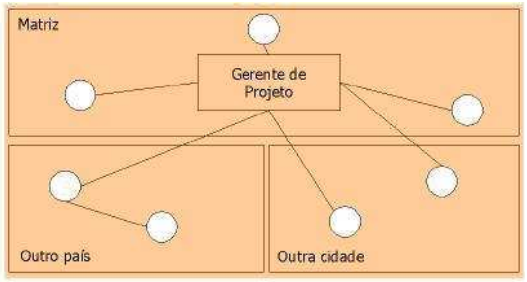


Figura. 9 – Equipe distribuída

Fonte: AUDY e PRIKLADNICKI, 2004, p. 3.

A Fig. 8 representa uma equipe centralizada, desenvolvendo seus projetos da maneira tradicional, localizada em uma mesma localização física. Já a Fig. 9 trabalha de maneira distribuída, ou seja, existem equipes junto a matriz da organização, em outras cidades e até mesmo equipes localizadas em outro país. Percebe-se, neste momento, a importância do controle no fluxo de trabalho das equipes que trabalham de maneira distribuída, principalmente para facilitar a gerência dos processos e a automatização das tarefas.

O DDS tem se tornado cada vez mais uma realidade. Com o objetivo de obter custos menores, maior qualidade no processo de desenvolvimento de *software* e a possibilidade de dispor estes recursos em âmbito global, muitas organizações começaram a investir em ambientes de desenvolvimento distribuído de *software* e *outsourcing* (contratação de uma organização externa, localizada em outro país) (AUDY; PRIKLADNICKI, 2004). Em um ambiente de desenvolvimento distribuído de *software* a comunicação, coordenação, colaboração, integração e a confiança entre equipes e diferenças culturais são aspectos relevantes que devem ser considerados. As definições de desenvolvimento distribuído e global de *software* são equivalentes. Diferem-se apenas no contexto em que são usadas. DDS pode ser aplicado em uma mesma cidade ou empresa, desde que seja de forma dispersa. Já o desenvolvimento global, é aplicado quando o desenvolvimento ocorre em países diferentes. Existem diversos conceitos relevantes que devem ser considerados em se tratando de DDS. Dentro do contexto de desenvolvimento distribuído podem ser aplicados conceitos de equipes globais e organizações virtuais. Segundo Horvath e Marquardt (2001), uma equipe global refere-se a um grupo de pessoas de diferentes nacionalidades que trabalham unidas em um projeto comum, através de culturas e fusos horário distintos, por um extenso período de tempo. Com relação às organizações virtuais, são definidas por Karolak (1998) como entidades caracterizadas por realizar partes de um projeto em locais distintos, comportando-se como se estivessem no mesmo local. Muelleitner et al. (1998), define uma organização virtual, como uma equipe formada por um grupo de empresas, localizadas em regiões geográficas distintas e que desenvolvem seu trabalho em conjunto através de uma infra-estrutura em rede para atingir os objetivos de negócio. Uma organização virtual existe desde a formação da organização até o momento de encerramento do projeto. Sua composição é descrita na Fig. 10, conforme Cyrillo e Hirama (2005).

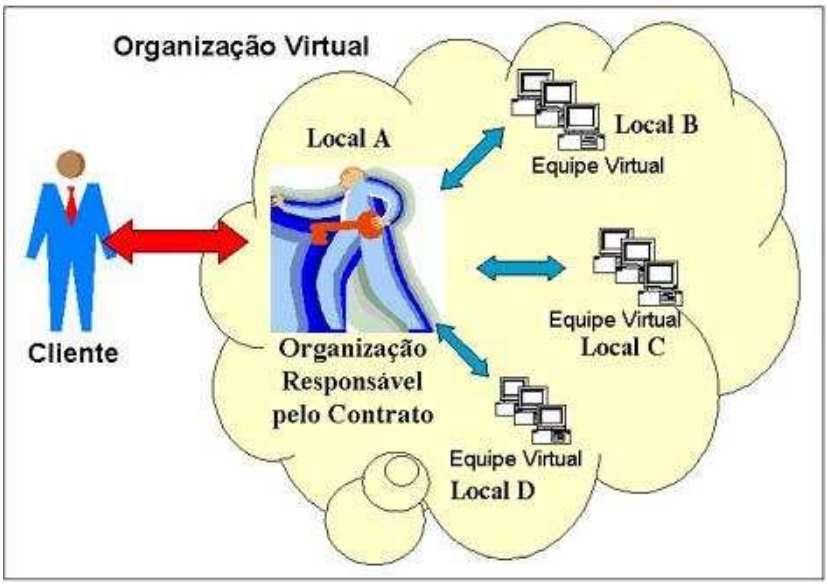


Figura 10: Organização virtual
Fonte: CYRILLO e HIRAMA, 2005, p. 19.

Quando a distância física entre os desenvolvedores em um ambiente de DDS envolve mais de um país, Karolak (1998) define uma instância do DDS chamada de desenvolvimento global de *software* (*Global Software Development* – GSD). O GSD é instanciado através de equipes globais de desenvolvimento de *software* (*Global Software Teams*), que Carmel (1999) define como sendo um grupo de pessoas distribuídas em dois ou mais países, colaborando ativamente em um projeto comum. Neste sentido, o DDS tem sido caracterizado pela colaboração e cooperação entre departamentos de organizações e pela criação de grupos de desenvolvedores que trabalham em conjunto, localizados em cidades ou países diferentes, distantes temporal e fisicamente (AUDY; PRIKLADNICKI, 2004).

3.1 Tipos de projeto de desenvolvimento de *software*

Existem diversos tipos de projeto de desenvolvimento de *software*. Com o objetivo de identificar estes tipos e a complexidade existente na gestão de projetos, Evaristo e Fenema (1999) desenvolveram uma tipologia para a classificação dos tipos de projeto possíveis. A tipologia está baseada em duas características de projetos: tipo de projeto e distribuição geográfica das equipes do projeto. Variando

estas características, foram encontrados sete tipos de projetos, representados na Fig. 11.

1. Projeto Tradicional: Projeto executado em apenas uma localização.

2. Projeto Distribuído: Projeto executado em mais de uma localização.

3. Múltiplos Projetos Distribuídos: Localização discreta. Programa formado por um conjunto de projetos executados em mais de uma localização. Cada localização possui uma equipe virtual que trabalha em apenas um projeto distribuído.

4. Múltiplos Projetos Distribuídos: Localização compartilhada. Programa formado por um conjunto de projetos executados em mais de uma localização. Pelo menos uma localização possui equipe virtual que trabalha em mais de um projeto distribuído e compartilhado com localizações diferentes.

5. Múltiplos Projetos Tradicionais: Programa formado por um conjunto de projetos que são executados em localizações diferentes.

6. Programa Localizado: Conjunto de projetos relacionados que são executados em uma mesma localização.

7. Múltiplos Programas Localizados: Múltiplos programas em que cada programa é composto por um conjunto de projetos executados em uma única localização.

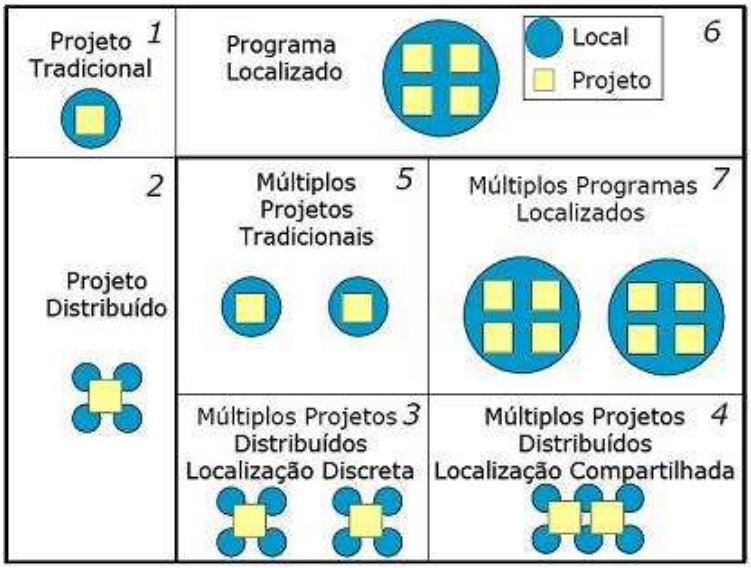


Figura 11: Tipologia de projetos.

Fonte: EVARISTO e FENEMA, 1999, p. 21.

Segundo Evartisto e Fenema (1999), a evolução na utilização dos tipos de projetos apresentados deve ser feita de forma gradual. Passar de um projeto tradicional diretamente para um projeto do tipo três ou quatro aumenta a complexidade do projeto em dois sentidos: quantidade de projetos e quantidade de localizações. Apesar do último nível (projetos tipo três e quatro) apresentar um maior ganho em termos de produtividade, também requer maior esforço no sentido de comunicação e coordenação das atividades. Desta forma, o caminho para se atingir este tipo de projeto deve ser feito em progressivos aumentos de escala de complexidade.

De acordo com Audy e Prikladnicki (2004), apesar de muitas vezes o processo de desenvolvimento de *software* ocorrer em um mesmo país, em regiões com incentivos fiscais ou de concentração de massa crítica em determinadas áreas, algumas empresas, visando um diferencial competitivo, buscam soluções globais. Neste contexto, surgem os conceitos de *offshore outsourcing* (contratação de uma organização externa, localizada em um outro país) e *offshore insourcing* (contratação de uma subsidiária da própria organização, localizada também em um outro país), o que potencializa os problemas e os desafios existentes no DDS. Ainda segundo estes autores, diversos fatores têm contribuído para o crescimento do DDS, entre eles: a necessidade de recursos globais para serem utilizados a qualquer hora; as vantagens de estar perto do mercado local, incluindo o conhecimento dos clientes e as condições locais para explorar as oportunidades de mercado; a grande pressão para o desenvolvimento *time-to-market*, utilizando as vantagens proporcionadas pelo fuso horário diferente, no desenvolvimento conhecido como *follow-the-sun* (24 horas contínuas, contando com as equipes fisicamente distantes). De acordo com Carmel (1999), tanto o desenvolvimento de *software* tradicional quanto o distribuído possuem diversas dificuldades. As principais características que os diferenciam são: dispersão geográfica (distância entre equipe de projeto, clientes e usuários), dispersão temporal (diferenças de fuso horário) e diferenças culturais (idioma, tradições, costumes, normas e comportamento). Segundo Herbsleb e Moitra (2001), estas diferenças refletem em diversos fatores. Entre eles, questões geográficas (decisão de desenvolver ou não um projeto de forma distribuída, tendo por base análises de risco e custo-benefício), questões culturais (valores, princípios, etc. entre as equipes distribuídas), questões técnicas (fatores relativos à infra-estrutura tecnológica, como redes de comunicação de dados, plataformas de *hardware*,

ambiente de *software*, e ao conhecimento técnico necessário, como processo de desenvolvimento para o desenvolvimento dos projetos distribuídos); e questões de gestão do conhecimento (fatores relativos à criação, armazenamento, processamento e compartilhamento de informações nos projetos distribuídos).

Apesar de inúmeras vantagens no desenvolvimento distribuído, também são encontradas algumas dificuldades a respeito deste tipo de desenvolvimento. A seguir serão apresentadas algumas dessas dificuldades.

3.2 Problemas relacionados com o desenvolvimento distribuído

O DDS apresenta algumas dificuldades e problemas que devem ser solucionados. Segundo Audy e Prikladnicki (2004), entre os diversos problemas relacionados estão: a falta de uma padronização nas atividades entre as equipes distribuídas; dificuldade de compartilhar informações; falta de um processo de desenvolvimento bem definido; barreiras em relação ao idioma e comunicação; diferenças culturais e contexto; e aquisição de confiança entre as equipes distribuídas. Carmel (1999) aborda a formação de equipes globais de desenvolvimento de *software* e os principais fatores que devem ser considerados ao montar uma equipe para um projeto distribuído (Fig. 12). O autor sugere a existência de cinco fatores que podem levar uma equipe distribuída ao fracasso: comunicação ineficiente, falta de coordenação, dispersão geográfica, perda do espírito de equipe e diferenças culturais, chamadas de forças centrífugas. Além disso, sugere a existência de seis fatores que podem levar a equipe ao sucesso: infra-estrutura de comunicação, arquitetura do produto, construção de uma equipe, metodologia de desenvolvimento, tecnologia de colaboração e técnicas de gerência, chamados de forças centrípetas.

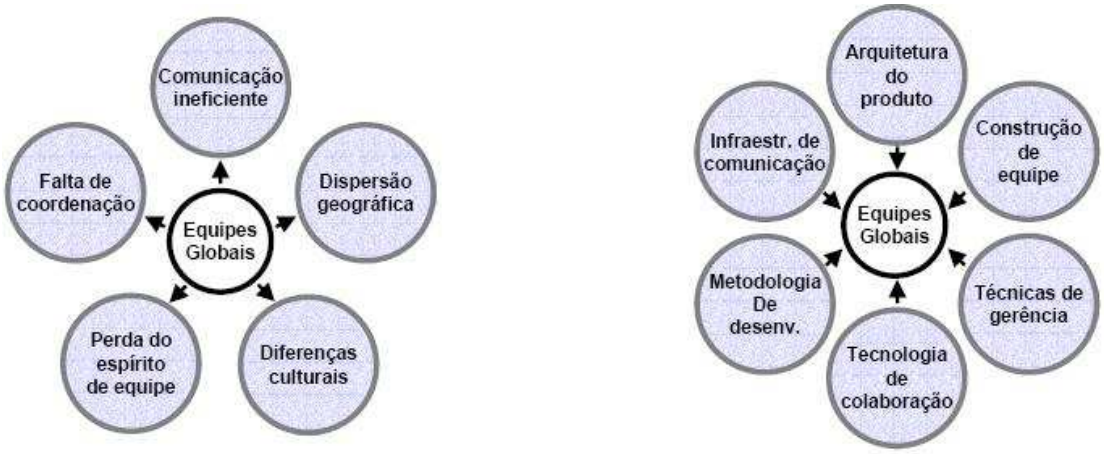


Figura 12: Forças centrífugas e centrípetas de equipes globais.
Fonte: AUDY e PRIKLADNICKI, 2004, p. 4.

Existem diversos fatores que podem comprometer o DDS, como dificuldade na comunicação entre os colaboradores, feita em horários inadequados devido à diferença de fuso horário. Perda da comunicação informal, fator essencial para a coordenação do projeto. Dificuldade de saber quando contatar uma determinada pessoa. Além disso, existe a dificuldade de saber quem é o responsável por um determinado componente, quem projetou ou implementou determinada parte do sistema a fim de resolver um problema. Existe ainda o fator cultural. Culturas diferentes geralmente possuem comportamentos diferentes. O idioma e a forma de utilização deste idioma podem influenciar negativamente, prejudicando o desenvolvimento. A distribuição do trabalho pode ser afetada por questões relacionadas a cada país, como feriados e normas religiosas diferentes em cada região. Segundo Oppenheimer (2002), fatores culturais dificultam o relacionamento entre membros de equipes virtuais e o estabelecimento de confiança entre eles. Nestes casos, mesmo quando as decisões de projeto são tomadas e comunicadas corretamente, as equipes não agem de acordo com o estabelecido. Outras dificuldades que ocorrem é na gestão e distribuição de informações. Apesar da grande quantidade de informações relativas ao andamento do projeto, em muitos casos, não estão acessíveis às equipes que precisam da informação, não são claramente entendidas e não estão atualizadas. Isto ocorre, pois, muitas vezes, o responsável pela informação esquece ou comunica os destinatários errados. Este tipo de problema dificulta o entendimento das estratégias, prioridades e decisões de

uma organização. Além disso, riscos críticos não são percebidos (OPPENHEIMER, 2002). Com o desenvolvimento distribuído, ocorre a dificuldade de coordenação e controle, pois implica num conjunto de atividades interdependentes. A distribuição geográfica aumenta as barreiras para a integração das atividades e controle dos produtos. Em um único ambiente de desenvolvimento, mecanismos informais são utilizados. Já em ambientes distribuídos isto não ocorre. No ambiente distribuído, não é possível resolver dúvidas de maneira informal, nem acompanhar o projeto no qual o gerente pode entrar em contato fisicamente com seus funcionários para verificar o andamento do projeto e resolver problemas de coordenação e controle (CARMEL, 1999). Segundo Freitas (2005), outro fator a ser considerado é sobre a infra-estrutura das organizações. A existência de infra-estrutura adequada em todas as organizações envolvidas no desenvolvimento pode ser difícil de ser obtida. Além de adequada, deve ser compatível entre as organizações no que diz respeito a sistemas operacionais, ambientes de desenvolvimento, etc. A autora cita também problemas estratégicos, gerados pela dificuldade em dividir entre as equipes o trabalho a ser realizado. Devem-se considerar diversos aspectos, como o grau de conhecimento sobre as tecnologias a serem utilizadas em cada atividade, a infra-estrutura e os recursos disponíveis. O ideal é que a divisão do trabalho permita a maior autonomia possível a cada equipe, para que ela não dependa das demais para a realização do trabalho sob sua responsabilidade. Se as informações sobre o processo não estiverem disponíveis e atualizadas, podem acarretar dificuldades aos gerentes de projeto para acompanhar o andamento do processo. A ausência de documentação compromete a integração entre as equipes, o que faz com que sejam feitos trabalhos repetitivos sem oportunidades de reuso no projeto. Em projetos distribuídos, cada equipe gerencia seu processo de *software*. Se houver problemas na sincronização dos processos, as equipes podem definir a mesma atividade de maneiras diferentes (FREITAS, 2005).

Além dos problemas referentes à gerência de processos, o gerenciamento de processos distribuídos apresenta alguns desafios decorrentes da distribuição do processo. A utilização de ambientes de suporte a processos distribuídos tende a facilitar bastante a solução de diversos desses desafios.

3.2.1 Desafios na gerência de processos distribuídos

Além dos problemas referentes ao processo de desenvolvimento, a gerência de processos distribuídos enfrenta os problemas causados por essa distribuição do desenvolvimento. A partir dos problemas decorrentes do DDS, podem-se traçar os principais desafios a serem considerados na gerência de processos distribuídos, segundo Freitas (2005): definição clara da estrutura de dependência entre as equipes; definição de critérios para atribuição de atividades às equipes; definição de estratégias para redução da distância cultural; estabelecimento de interfaces bem definidas para comunicação formal; criação de canais para comunicação informal; garantia de consistência e disponibilidade dos artefatos e documentos do projeto; acompanhamento do progresso das equipes.

Diversos trabalhos têm sido desenvolvidos a fim de propor soluções para estes problemas. A estrutura de dependências entre equipes pode ser explicitada pela definição de relações entre os subprocessos (FREITAS, 2005). Karolak (1998) sugere uma série de critérios para a divisão de atividades, dentre eles: relações comerciais, arquitetura do sistema, conhecimento das equipes e disponibilidade de ferramentas e recursos. Diversas estratégias têm sido propostas para diminuir a distância cultural: escolha estratégica de projetos, de modo a definir equipes com semelhanças culturais ou com experiência em trabalho conjunto; definição de equipes com 25% do trabalho realizado perto do cliente; utilização do papel do *liason*, um gerente de projeto que viaja freqüentemente entre as equipes; investimento no fortalecimento de uma língua comum entre as equipes (AGARWAL; CARMEL, 2001). A definição de interfaces de comunicação formal pode ser obtida por meio de modelos de processo bem definidos, com marcos (*milestones*) e métricas bem estabelecidas. Os canais de comunicação informal, por sua vez, podem abranger videoconferência, espaços de trabalho compartilhado, programas de troca de mensagens (*instant messengers*), etc. A consistência e a disponibilidade de artefatos e documentos podem ser obtidas pela manutenção de uma página do projeto na *WEB* que contenha descrição, métricas, planejamento e informações sobre as equipes do projeto e pela utilização de *software* de gerenciamento de configuração. O acompanhamento do andamento das equipes pode ser realizado pela monitoração da execução dos processos de *software*, pela realização de

reuniões (ou videoconferências) em intervalos regulares e pela definição da freqüência para sincronização do andamento dos processos locais (FREITAS, 2005).

4 *WORKFLOW*

Este capítulo apresenta os principais conceitos de *workflow*. Serão apresentados dados históricos relacionados a essa tecnologia, o surgimento do *workflow*, tipos, componentes e o modelo de referência, proposto pela *Workflow Management Coalition* - WfMC.

4.1 Histórico

Os primeiros trabalhos abordando a tecnologia de *workflow* surgiram no início da década de 70, quando começa a surgir também a idéia de uma organização de fluxo de documentos. Neste período, a intenção era a organização e o compartilhamento de documentos a fim de diminuir o acúmulo de documentos em papel nas organizações (RIZZI, 2001). Devido à tecnologia limitada e não acessível à maioria dos locais de trabalho, não se tinha acesso em tempo real aos recursos computacionais e nem informação distribuída e, por isto, estes conceitos foram, de certa forma, esquecidos por um período de tempo (TESSMANN, 2005). Além destes fatores, não existia mão-de-obra qualificada para trabalhar com este tipo de organização de trabalho (NICOLAU, 1998). Os primeiros sistemas de *workflow* tornaram-se inflexíveis, devido, principalmente, às inúmeras situações não previstas ocasionadas pelo fluxo de processo, fazendo com que os projetos na área não ganhassem força e fossem abandonados (ACOSTA; FAEDRICH, 2003). Na década de 80, voltaram-se as pesquisas para *Computer Supported Cooperative Work* - CSCW e *Groupware*. Tendo como foco central os diferentes tipos de *groupware* organizacionais (correio eletrônico, vídeo conferência, etc.) nos quais *workflow* não estava incluído (NICOLAO, 1998). Nos anos 90, sobressaíram-se as pesquisas sobre novos paradigmas de interação, baseados na exploração do potencial da *World Wide Web*, levando as pesquisas em *workflow* a um novo patamar voltado para a definição de arquiteturas distribuídas de execuções de processos (RIZZI, 2001). Conforme os recursos computacionais aumentaram e ficaram mais acessíveis comercialmente, a tecnologia de *workflow* foi sendo retomada com o intuito de

coordenar os processos durante o seu período de duração e prover ferramentas em tempo real para a comunicação e trabalho colaborativo entre pessoas distantes geograficamente (VIEIRA, 2005).

Hoje em dia, com a larga utilização de redes de computadores e da Internet, o paradigma de *workflow* volta-se a coordenar atividades distribuídas, facilitar o controle, auditoria e segurança dos processos, independente da localização geográfica dos participantes dos processos (VIEIRA, 2005). Um sistema de *workflow* já não serve apenas para distribuir tarefas, mas também, gerenciar de forma automatizada o processo como um todo, sendo ele próprio responsável pela sua correta execução (TRAMONTINA, 2004).

4.2 Conceitos iniciais

Existem diversas visões sobre esse tema, pois ainda não foi atingido um consenso entre aqueles que trabalham no seu desenvolvimento. O conceito de *workflow* está ligado diretamente com automação de processos, utilizando uma seqüência de passos, de acordo com um conjunto de regras pré-definidas, realizadas por pessoas e/ou máquinas. Segundo Tramontina (2004), pode-se entender *workflow* inicialmente como um meio de visualizar, analisar e melhorar os processos, sempre buscando a automação por ferramentas específicas. O autor destaca que a tecnologia *workflow* também trata da automação de processos onde documentos, informações ou tarefas são trocados entre os seus participantes, de acordo com um conjunto definido de regras e objetivos. Não existe um conceito único que defina exatamente do que trata a tecnologia de *workflow*, porém, a grande maioria dos pesquisadores enfoca o controle de processos e a sua melhora com o uso de *workflow*. Georgakopoulus (1995) define *workflow* como sendo um grupo de tarefas organizadas de maneira a realizar um processo de negócio. Sob o guarda-chuva do *workflow*, termo freqüentemente utilizado, aplica-se a consultas a um processo de negócio, a especificação de um processo, ao *software* que executa e automatiza um processo ou ao *software* que suporta simplesmente a coordenação e colaboração das pessoas que executam um processo. Os vários conceitos atribuídos ao termo *workflow*, segundo Georgakopoulus (1995), são apresentados na Fig. 13.

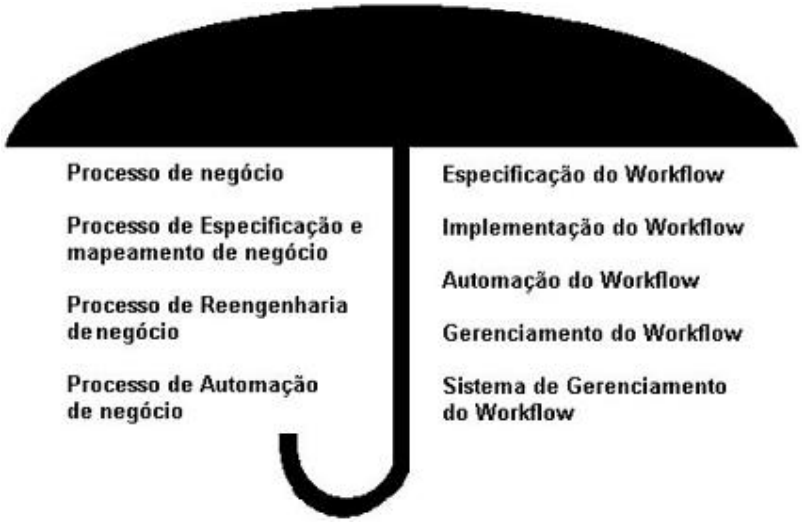


Figura 13: Guarda-chuva de *workflow*.
Fonte: VIEIRA, 2005, p. 14.

Segundo Usirono (2003) o que caracteriza o *workflow* é o fluxo de trabalho, às vezes repetitivo, que tem um início e um fim bem definidos. Portanto, baseado nestes conceitos, pode-se dizer que *workflow*, trata do gerenciamento e automação dos processos a fim de obter uma melhora da produtividade e integração dos participantes das respectivas atividades dentro dos processos.

4.2.1 Definição de processos

Um *workflow* visa à representação e otimização dos processos que ocorrem no mundo real. Para isso é necessário definir estes processos através de regras e de uma representação computacional, visual ou não. Segundo Usirono (2003), os processos têm por objetivo a produção de bens e serviços através da manipulação e processamento de matéria-prima através de um conjunto de atividades. Estes processos podem ainda ser divididos em subprocessos, atividades, procedimentos e tarefas. Para exemplificar o conceito de processos, é apresentada a Fig. 14, segundo Tramontina (2004). Nela encontra-se a especificação de um processo que uma empresa fictícia usa para lidar com os casos de reclamações de defeito em seus produtos, por parte dos clientes. Na figura os retângulos com laterais arredondadas representam as atividades do processo; os triângulos marcados com a

letra “o” representam pontos onde há uma separação de caminhos e o caso deve seguir por apenas uma das rotas que saem do triângulo. O fechamento desta diferenciação de caminhos, ou seja, o ponto onde a diferenciação termina, é também representado por este mesmo triângulo. Os retângulos pequenos marcados com a letra “e” representam tanto a abertura quanto o fechamento de atividades paralelas, que devem ser executadas ao mesmo tempo. E, por fim, os losangos marcados com a letra “l” representam iterações, onde há um ponto de retorno a uma atividade anterior, o que representa a formação de um laço ou *loop*. Essa representação gráfica é possível utilizando diversas ferramentas de *workflow*, sendo depois traduzidas para outros formatos digitais aceitos pelo WFMC, como definição de processos.

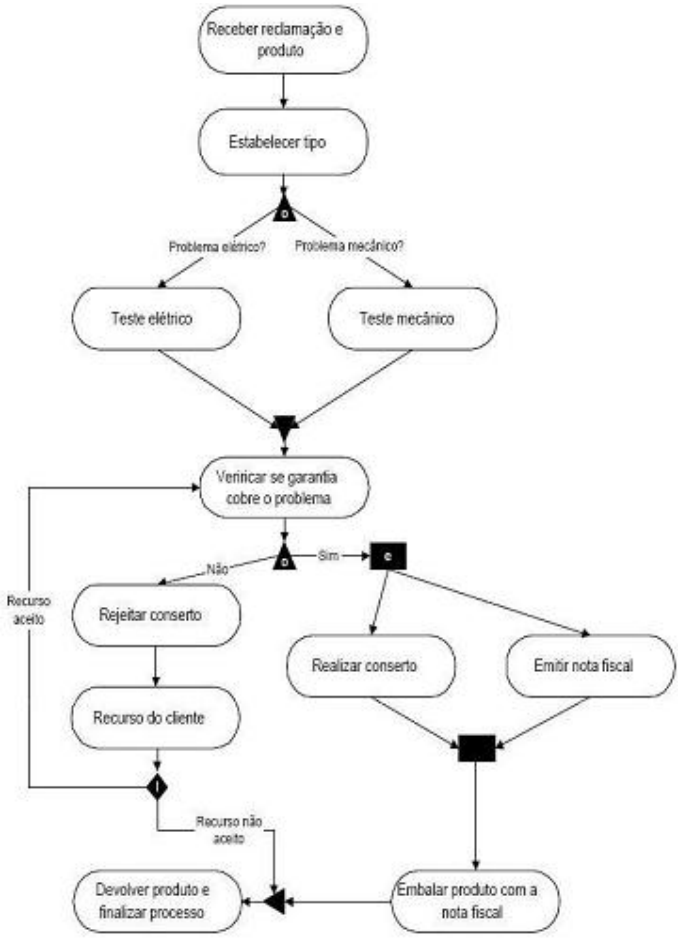


Figura 14: Exemplo de definição de processo

Fonte: TRAMONTINA, 2004, p. 7.

4.2.2 Definição de atividades

Atividades dentro de processos são cada uma das partes do trabalho realizado pelo processo como um todo. Segundo Tramontina (2004), dentro de um processo, cada atividade é uma etapa que deve ser realizada para que o processo se conclua. O autor acrescenta que uma atividade pode ser chamada de uma atividade especial quando ela é um subprocesso de um processo maior que a encapsula, sendo que esta é considerada especial, pois pode ser dividida enquanto as atividades ordinárias são passos indivisíveis do processo. Segundo Reijers (2002), uma atividade é a menor parte que se pode distinguir em um processo. Estas atividades que, segundo o autor também são muitas vezes referenciadas como passos, tarefas ou ações, definem a *"completa especificação de uma parte do trabalho a ser realizado"*. Acrescenta ainda que os limites de uma tarefa ou atividade são tipicamente determinados pelo escopo dos recursos disponíveis para a realização da atividade, podendo ser também determinados pelo tempo esperado para conclusão da atividade, pela localização do trabalho, pelo número e habilidades das pessoas envolvidas e todo tipo de regulamentação. Para Georgakopulus (1995) uma atividade define um trabalho a ser feito e que pode ser especificado por uma descrição textual, um formulário ou um programa de computador. Ele diz que um conjunto dessas tarefas ou atividades organizadas de forma a realizar um processo de negócio define um *workflow*, e acrescenta que essas atividades podem ser desenvolvidas por um ou mais programas de computador, um conjunto de pessoas ou a combinação deles. As atividades definem tarefas que são realizadas por uma ou mais pessoas ou recursos e que devem ser descritas com o objetivo de serem otimizadas dentro do contexto de um processo (TESSMANN, 2005).

4.2.3 Componentes de um *workflow*

Para definir um sistema de gerenciamento de *workflow*, é importante a definição de todos os componentes envolvidos. Araujo e Borges (2001) diferem os seguintes componentes de um *workflow*: atividades, papéis, rotas, regras, atores e dados, formulários ou documentos. Rizzi (2001) acrescenta alguns componentes a esta classificação como: eventos, processos, Itens de trabalho, listas de trabalho, instâncias, gatilho e o *workflow* propriamente dito, conforme descritos a seguir:

Atividades: é o elemento fundamental de trabalho do *workflow*. É um conjunto de eventos que executam um passo lógico que pode ser realizado por vários atores e está sob a responsabilidade de um ator; **ator**: participante do *workflow* que poderá assumir um papel e executar uma atividade durante a execução de uma instância do *workflow*, pode ser tanto uma pessoa quanto um sistema automatizado; **papel**: a cada atividade são associados tipos de usuários que podem executá-las ou serem responsáveis pelas mesmas. Este conceito surge porque, dentro das organizações, existe uma rotatividade natural dos executores das tarefas (atores), mas as características necessárias para a execução da tarefa permanecem. Isto torna o sistema mais flexível, não sendo necessária uma reorganização de todo o fluxo de trabalho cada vez que os executores são substituídos ou cada vez que um novo ator passa a fazer parte do sistema. Assim, a execução das atividades está ligada a um conjunto de características que define um papel dentro do sistema e não diretamente ao executor da atividade; **rotas**: definem quais atividades devem ser executadas umas após as outras, quais devem ser executadas em paralelo e quais dependem de uma condição para tomarem um caminho ou outro, ou seja, definem o fluxo das atividades; **regras**: são informações relativas à maneira de trabalhar da organização. São regras organizacionais administrativas que definem quais dados devem ser passados de uma atividade para a outra, quais atividades devem prosseguir e de que elas dependem para prosseguir. Essas regras não são partes do *workflow* em si, mas são elas que definem a maneira que os dados são controlados pelo *workflow*, para onde eles devem ir e como processá-los. A verificação e validação dessas regras se dão através de uma correta modelagem de atividades, rotas, papéis, etc.; **dados**: são todas as informações inerentes àquela atividade ou aquele processo. Podem ser documentos, formulários que podem ser preenchidos por um ou mais executores em uma ou mais atividades distintas, ou quaisquer outros tipos de dados que são relevantes àquele fluxo de trabalho; **eventos**: algum acontecimento observável. Difere-se de atividades pelo fato das atividades estarem associadas a um intervalo de tempo enquanto o evento ocorre em um determinado instante de tempo. Um evento pode disparar atividades assim como atividades podem disparar eventos. Esses eventos podem ser internos ou externos ao *workflow*; **processos**: são um conjunto de atividades que visam um mesmo objetivo e possuem uma ligação lógica dentro do *workflow*. Um processo pode ter vários subprocessos sendo que o próprio

workflow pode ser considerado um processo; **item de trabalho**: é a representação do trabalho a ser processado por um ator em uma instância do *workflow*; **listas de trabalho**: a lista de trabalho relativa a um ator indica quais itens de trabalho devem ser executados por ele; **instâncias ou casos**: é a representação de uma única ocorrência de um *workflow* em execução, incluindo seus dados. Essas instâncias podem ser executadas paralelamente, ou seja, várias instâncias sendo executadas ao mesmo tempo; **gatilhos ou *triggers***: aparecem quando a ocorrência de um evento dá início a uma atividade específica. É dito como sendo uma regra do *workflow* que, na ocorrência de um evento, verifica se as suas condições foram satisfeitas e dispara uma atividade. Quando o evento A dispara a atividade B pode-se dizer que o evento A é o *trigger* da atividade B; ***workflow***: Objetiva a automação e gerência de processos dentro de uma organização, através do controle e gerenciamento do fluxo de trabalho.

Esta representação dos elementos de um *workflow* é ilustrada na Fig. 15.

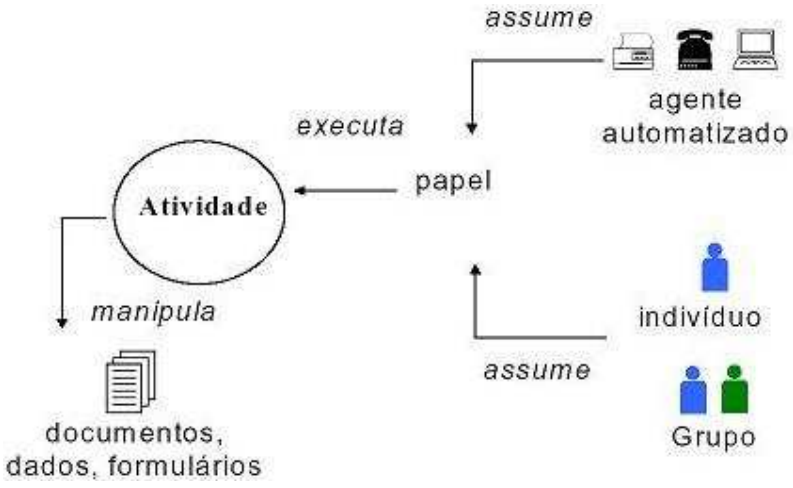


Figura 15: Elementos de um fluxo de trabalho.

Fonte: ARAUJO e BORGES, 2001, p. 5.

4.3 Execução de fluxos de trabalho

Segundo Araujo e Borges (2001), os fluxos de trabalho são executados através da ativação de instâncias de sua definição. Várias instâncias de um mesmo processo ou de um processo distinto podem estar em execução simultaneamente em um sistema de *workflow*. O sistema de *workflow* acompanha e coordena a execução de cada uma das instâncias ativas, conforme o fluxo definido no modelo do processo, e encaminha cada atividade para o ator ou atores correspondentes. O encaminhamento destas atividades aos respectivos executores provoca a inclusão de itens de trabalho (*workitens*) nas listas de trabalho (*worklists*) dos atores do processo. As *worklists* contêm as atividades que deverão ser executadas por um determinado usuário e podem conter, de forma simultânea, atividades de várias instâncias de processos diferentes que estão em execução. As atividades ocorrem no ambiente de trabalho de cada ator participante do *workflow*. A realização de uma determinada tarefa pode envolver, ainda segundo Araujo e Borges (2001), a manipulação dos documentos estipulados no fluxo de trabalho para análise de informações, tomada de decisões ou preenchimento de dados. Quando a atividade é finalizada o processo é reposto de volta ao fluxo, disparando novas atividades de acordo com os resultados gerados.

A Fig. 14 ilustra essa interação entre o usuário e o sistema de *workflow*. Primeiramente o usuário seleciona a tarefa na sua *worklist*. Então, ele faz a recuperação do documento, analisa e toma todas as decisões referentes à atividade designada para ele. Após a atualização do documento e a conclusão da tarefa, o processo é reposto de volta ao fluxo e o *status* da atividade é atualizado.

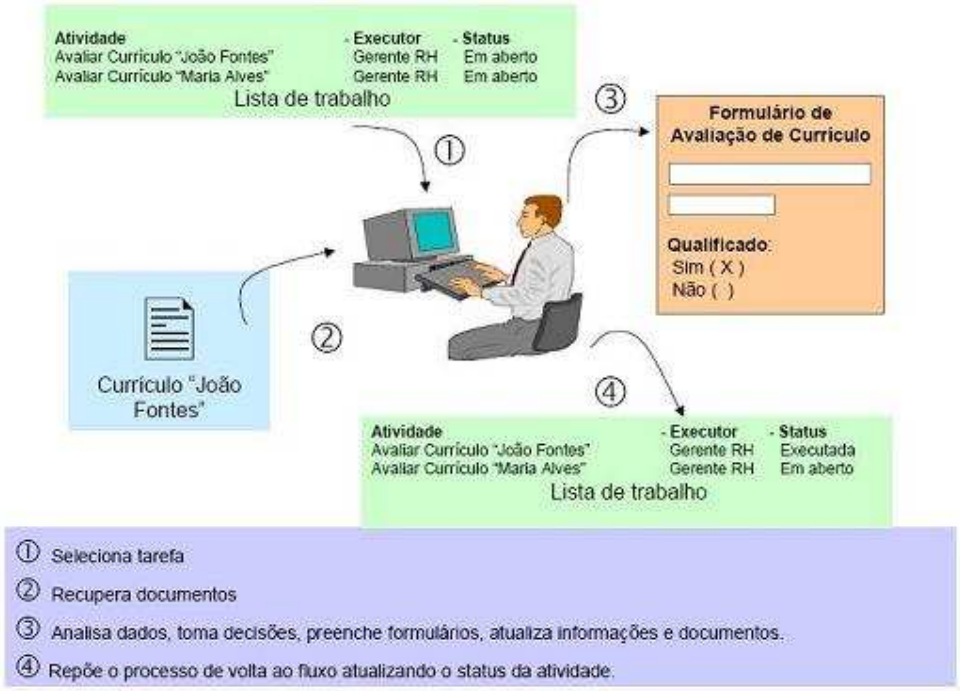


Figura 16: Interação entre usuários e o sistema de *workflow*
Fonte: ARAUJO e BORGES, 2001, p. 15.

Após todas estas definições dos conceitos inerentes a *workflow*, podemos conceituar um sistema de gerenciamento de *workflow* - WfMS, segundo a WfMC, como sendo um sistema que define, cria e gerencia a execução de *workflows* através do uso de software, rodando em um ou mais motores de *workflow*, que é capaz de interpretar a definição do processo, interagir com os participantes do *workflow* e, quando requisitado, invocar o uso de aplicações e ferramentas de tecnologia da informação. Mais sobre os WfMS será visto no item a seguir.

4.4 *Workflow management system* - WfMS

Um sistema de *workflow* pode ser organizado de maneira manual, porém, o que se busca é sempre a utilização de ferramentas computacionais que forneçam suporte automatizado à realização do trabalho. Apesar de existirem diversos produtos diferentes que se colocam como ferramentas de *workflows*, existem características comuns entre eles, no que diz respeito às suas funções principais (TRAMONTINA, 2004). A WfMC identifica que, em um nível mais alto, todos os

WfMS têm as seguintes funcionalidades: funções de **tempo de construção** (*build time functions*), que lidam com a definição e modelagem dos processos de *workflow* e suas atividades constituintes. O resultado final desta fase é a definição de processo acabada; função de **tempo de execução** (*run-time functions*), que gerenciam e executam os processos de *workflows* em um ambiente operacional, além de seqüenciar e alocar as atividades necessárias para a execução desses processos. Estas funções que criam uma instância de processo para lidar com um caso específico, escalonam suas atividades para os recursos e acompanham este caso até o término de sua execução; e as funções de **interação**, em tempo de execução, com os usuários e ferramentas computacionais para o processamento de diversos passos de cada atividade.

A relação entre estes itens é verificada na Fig. 17. O sentido das interações é indicado pelas setas.

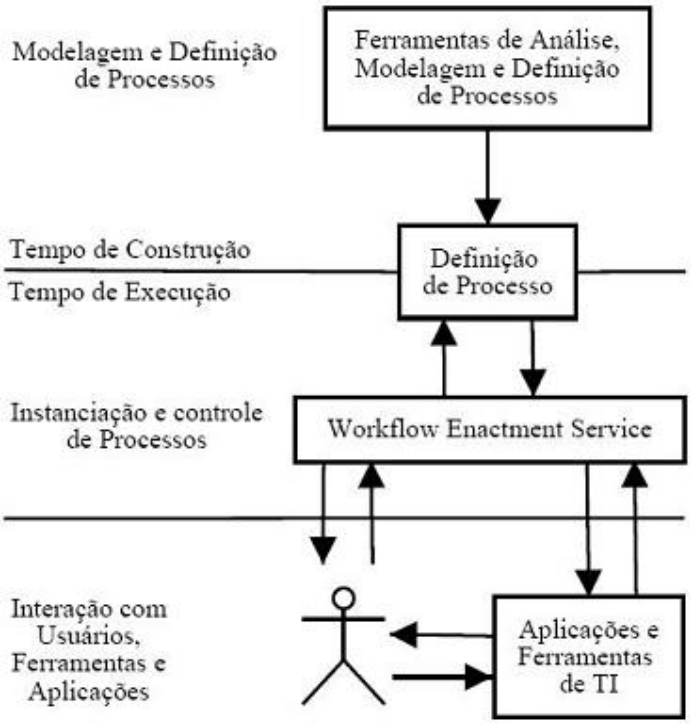


Figura 17: Relação entre as principais funções de um WfMS.

Fonte: TRAMONTINA, 2004, p. 9.

4.4.1 *Workflow engine* e *workflow enactment service*

Segundo a WfMC, uma *workflow engine*, um motor ou máquina de *workflow* é a parte responsável pelo andamento do *workflow*. É o módulo que gerencia e controla todos os componentes existentes, a execução dos processos, a seqüência de execução das atividades e a ativação de aplicativos externos. É o módulo da arquitetura responsável pela ligação dos outros módulos, ou seja, também funciona como uma interface entre diferentes módulos da arquitetura de um *workflow*. Uma ou mais *workflows engines* formam um domínio de *workflow*, ou seja, um ambiente homogêneo de execução dos processos. Este domínio é gerenciado pelo *workflow enactment service* - WES, que provê o suporte necessário para a execução das instâncias de processo pelo conjunto de *workflows engines* que os compõem. Assim sendo, um WES é formado por um ou mais *workflows engines*.

4.5 O modelo de referência da WfMC

Desde que a tecnologia do *workflow* começou a tomar forma muitas ferramentas surgiram. Porém, não existia a preocupação com a integração entre estas ferramentas, o que levava ao desenvolvimento de diversas soluções independentes entre si. Com o objetivo de padronizar o desenvolvimento de ferramentas de *workflow*, várias empresas desenvolvedoras dessa tecnologia se juntaram para criar a *Workflow Management Coalition*, ou WfMC. Em 1995, foi proposto por esta, um modelo de referência para sistemas de gerenciamento de *workflow*. O principal objetivo deste modelo de referência é promover a integração entre diferentes ferramentas de *workflow* de desenvolvedores diferentes, mas também, fornece uma ampla visão sobre as funcionalidades dos sistemas de *workflow* como um todo (TRAMONTINA, 2004). A Fig. 18 representa o modelo de WfMS proposto pela WfMC.

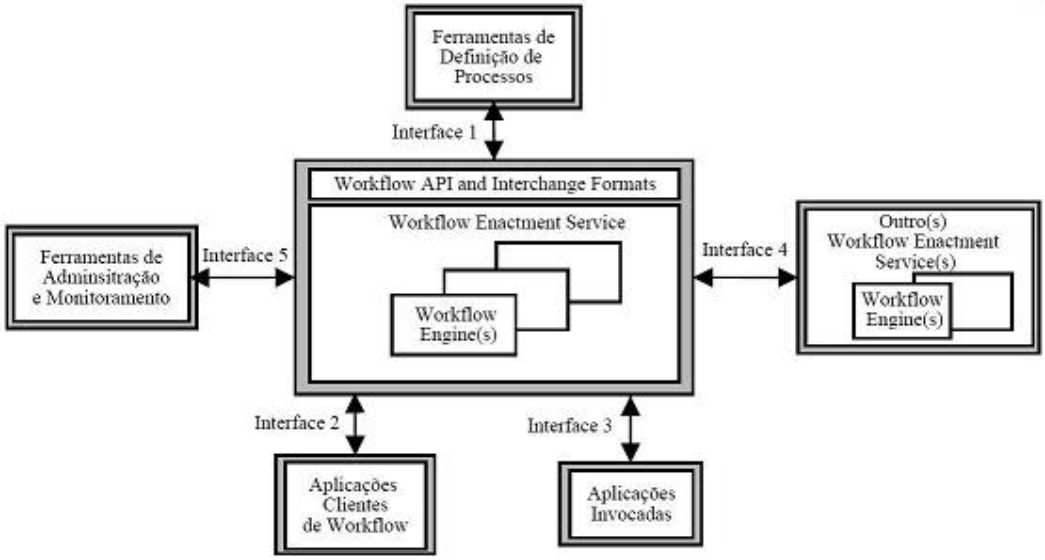


Figura 18: Modelo de referência para sistemas de *workflow* da WfMC
Fonte: WfMC 1995, p. 20.

A Fig. 18 demonstra o modelo da WfMC. Percebe-se que o centro do modelo é tomado pelas *workflow engines*, localizadas dentro do WES. Sendo este último encapsulado pela *Workflow Application Programming Interface* - WAPI, que possui cinco interfaces numeradas de um a cinco, cada uma cuidando da interação da WES com outros componentes do modelo. A WAPI representa a idéia da WfMC de um meio para a integração entre as ferramentas de *workflow* e consiste em um conjunto de funções, através das quais os componentes dos modelos interagem uns com os outros. Com isto, é possível que cada uma das partes do modelo seja projetada por diferentes desenvolvedores, mas cada uma possa agir juntas em único WfMS, através da WAPI. As principais partes da WAPI são as interfaces. Cada uma define um bloco lógico de interação entre as diferentes partes de um sistema de *workflow*. Através da interface 1 é feita a interação entre as ferramentas de definição de processos e o WES. A interface 2 permite o acesso de aplicações clientes às listas de itens de trabalho dos participantes do *workflow*. Na interface 3 é definida a relação entre o WES e as ferramentas computacionais por ele invocadas, sendo possível interagir com as aplicações e passar algum item de trabalho para elas, recebendo depois os resultados de volta. A interação de um WES com outros externos a este ocorre na interface 4, para a possível troca e/ou execução entre ferramentas de administração e monitoramento do sistema com WES.

4.6 Tipos de *workflow*

Assim como existem diversos conceitos de *workflow* e não apenas um conceito universal, existem também diversas caracterizações para *workflow*. As seções seguintes descrevem os tipos de *workflows* segundo a WfMC e classificações próprias de alguns autores, diferentemente do padrão da WfMC.

4.6.1 Tipos de *workflow* segundo a WfMC

Como visto na seção 4.5, a WfMC propôs um modelo de referência para estabelecer padrões para os tipos e caracterizações na área de *workflow*. Segundo a WfMC, os *workflows* podem ser distinguidos como: *workflow ad hoc, workflow* administrativo e *workflow* de produção. Cada um destes tipos de *workflow*, segundo a WfMC, serão vistos nos itens a seguir.

4.6.1.1 *Workflow ad hoc*

Workflows ad hoc executam processos de negócios, tais como documentação de produtos ou venda de produtos, onde não há um padrão pré-determinado de movimentação de informação entre pessoas. Tarefas do tipo *ad hoc* envolvem a coordenação humana. A ordenação e a coordenação de tarefas em um *workflow* do tipo *ad hoc* não são automatizadas, mas sim controladas por pessoas. Esta classe de *workflow* tipicamente envolve pequenos grupos de profissionais que tem a intenção de apoiar pequenas atividades que requerem uma solução rápida (GEORGAKOPOULOS; HORNICK, 1995). Tramontina (2004) apresenta um *workflow ad hoc* como sendo o tipo mais flexível de *workflow*. Segundo o autor, as definições dos processos podem ser facilmente modificadas mesmo durante o andamento do trabalho, quando necessário. Isto ocorre porque muitas vezes não se sabe ao certo como transcorrerá a atividade, antes de sua execução, nesse tipo de organização do trabalho. Este tipo de sistema de *workflow* é considerado pouco seguro por dar muita liberdade ao usuário do sistema. Araujo e Borges (2001) citam como exemplo de processo ad hoc a confecção de um relatório por uma equipe, ou então, a organização de um evento, ou também o projeto de um sistema.

4.6.1.2 *Workflow* administrativo

Um *workflow* administrativo envolve processos repetitivos com regras de coordenação de tarefas simples. A ordenação e coordenação de tarefas em *workflows* administrativos podem ser automatizadas. Esta classe de *workflow* não engloba um processamento complexo de informações e não requer acesso a sistemas de informação múltiplos usados para suportar produção ou serviços administrativos (GEORGAKOPOULOS; HORNICK, 1995). Segundo Usirono (2003), esses sistemas de *workflow* administrativos são compostos por atividades estruturadas que não fazem parte do principal objetivo da empresa e integram processos repetitivos e baseados em regras simples de direcionamento. Araujo e Borges (2001) dizem que este tipo de *workflow* geralmente apóia processos administrativos das organizações como: ordens de compra, pedidos de férias, admissão de funcionários, etc. e cita como exemplo de um processo administrativo, o processo de inscrição em disciplinas de uma universidade.

4.6.1.3 *Workflow* de produção

Um *workflow* de produção envolve processos de negócios repetitivos e previsíveis. Diferente dos administrativos, os de produção englobam um processamento de informações complexas envolvendo acesso a múltiplos sistemas de informação. A ordenação e coordenação de tarefas nestes tipos de *workflow* podem ser automatizadas. Contudo esta automatização é complexa, pois trabalha com processos de informações complexos e acesso a sistemas de informação múltiplos, para execução do trabalho e para a recuperação de dados (GEORGAKOPOULOS; HORNICK, 1995). Segundo Tramontina (2004), a única participação humana nesse tipo de *workflow* é para o tratamento de exceções que ocorrerem dentro do fluxo de trabalho. Assim, atividades repetitivas são organizadas de forma mais eficiente, tornando possível a realização de atividades bem mais complexas. Estes sistemas são muito mais utilizados integrando-se com outros sistemas já existentes dentro das organizações e, de acordo com Usirono (2003), as atividades deste tipo de *workflow* são de fundamental importância para a realização do trabalho da organização. Araujo e Borges (2001) consideram um exemplo deste tipo de *workflow*, o serviço de suporte de um sistema. Cada chamado ao suporte dá

início a um processo bem definido que começa com a análise do pedido, avaliação da gravidade do mesmo, encaminhamento para o setor específico, etc. até relatar ao usuário a solução do mesmo.

Uma importante observação, ainda segundo Araujo e Borges (2001), é que as categorias definidas nesta classificação não podem ser visualizadas de forma independente. Os autores dizem que este espectro deve ser observado de forma contínua e não como áreas mutuamente exclusivas. Ou seja, os processos não são totalmente *ad hoc* ou totalmente administrativos ou totalmente de produção. As diferenças relevantes entre este *workflow* de produção e o *ad hoc* ou administrativo são a interação do sistema de informação com os processos de negócio e o uso de executores de tarefas automatizadas (não humanos). A Fig. 19 representa esta visão do espectro de *workflow*.

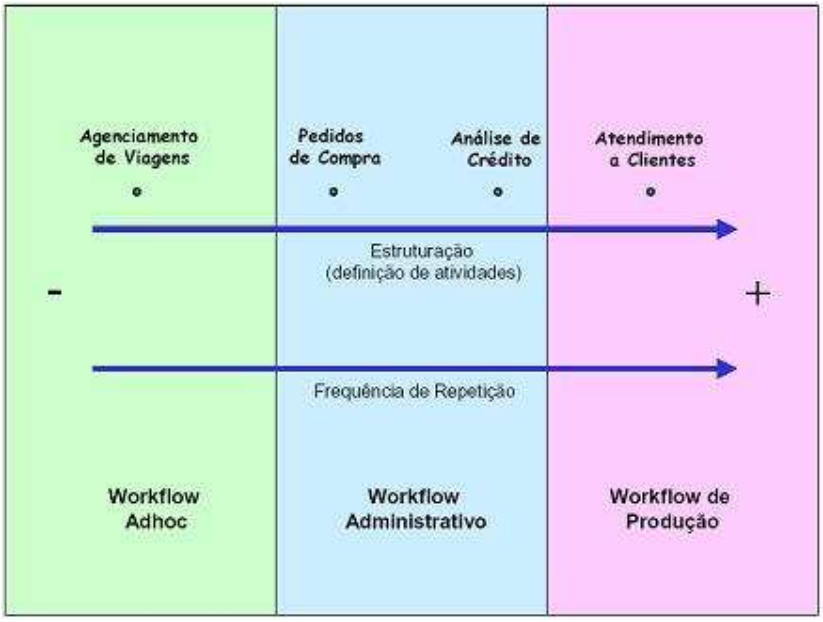


Figura 19: O espectro de *workflow*.

Fonte: ARAUJO e BORGES, 2001, p. 20.

4.6.2 Tipos de *workflow* segundo Georgakopoulus

Georgakopoulus no documento publicado em 1995 (WfMC), que traz novamente os estudos sobre *workflow*, descreve dois tipos de classificação: classificação comercial e caracterização quanto à orientação, conforme descritas a seguir**:**

4.6.2.1 Classificação comercial

A classificação comercial, segundo o autor, pouco difere da classificação já vista. A principal diferença é que o autor extingue os *workflows* colaborativos e coloca as atividades de *groupware* como parte integrante dos *workflows ad hoc*. Esta diferenciação não chega a ter tanta representação, pois, o principal objetivo dos *workflows* colaborativos é o trabalho em equipe, sem rigidez de regras na execução das atividades. Na Fig. 20, o gráfico apresentado por Georgakopoulos e Hornick (1995) representa a estruturação das atividades e suas respectivas complexidades. Comparando a complexidade em relação à estrutura do *framework* da figura, nota-se que estes *workflows* têm exigências de estrutura e complexidade maiores do que as encontradas em *workflows* de produção.

Figura 20: Comparação dos tipos de *workflows* comerciais

Fonte: VIEIRA, 2005, p. 25.

4.6.2.2 Caracterização quanto à orientação

Conforme Georgakopoulos e Hornick (1995) o Workflow pode ser caracterizado dentro de dois aspectos: orientado a pessoas e orientado a sistemas. Em *workflows* orientado a pessoas, as tarefas ou atividades a serem realizadas no fluxo de trabalho são tarefas não automatizadas, ou seja, devem ser executadas por seres humanos. O sistema de *workflow* para esse tipo de orientação deve auxiliar na coordenação e controle destas tarefas, além de alocar pessoal especializado para cada tarefa. As principais questões de orientação neste tipo de *workflow* incluem interação humano computador e a combinação de habilidades humanas para suportar as tarefas necessárias. Já os orientados a sistemas realizam tarefas especializadas, interagem com outros *softwares* e organizam o trabalho automatizado. Um sistema de *workflow* para este tipo de orientação deve coordenar e controlar tarefas de *software* com pouca ou nenhuma participação humana. Em um *workflow* orientado a sistema, as principais questões de orientação incluem: combinar as necessidades dos processos de negócios para a funcionalidade do sistema e providenciar dados a partir dos sistemas de informação existentes; interoperabilidade entre sistemas distribuídos (heterogêneo, assíncrono, distribuído); procurar *softwares* adequados para executar tarefas de *workflow*; determinar novas necessidades de *software* de forma a permitir automação dos processos de negócios; assegurar a execução correta e segura dos sistemas.

5 MODELAGEM DO SISTEMA

Inicialmente foram definidos os artefatos e os papéis dos participantes envolvidos neste processo de modelagem do fluxo de trabalho no ambiente de desenvolvimento distribuído proposto. Os papéis descrevem como as pessoas se comportam no negócio e quais são as suas responsabilidades. Normalmente os papéis podem ser desempenhados por uma pessoa ou um grupo de pessoas em um ambiente de desenvolvimento. Toda a modelagem do *workflow* que ocorre no ambiente de desenvolvimento distribuído proposto foi baseada nos seguintes processos: o Processo Orientado a Reuso, visto na seção 2.3.1, devido o reuso de código ser um importante artefato para a diminuição de código a ser escrito, diminuindo assim o tempo gasto com a implementação; e no RUP, detalhado na seção 2.4, por ser um dos processos de desenvolvimento de *software* mais abrangente e completo, embora não seja voltado especificamente para o desenvolvimento distribuído. Como visto no capítulo 2, é muito importante que em um ambiente distribuído se utilize um processo bem definido, adaptando-o para a realidade da organização e para seus reais objetivos.

Baseado nos referidos processos, os papéis existentes na equipe de desenvolvimento distribuído foram definidos como segue: os **analistas**, que são responsáveis pela liderança, coordenação e identificação de requisitos e a modelagem de casos de uso, delimitando o sistema e definindo sua funcionalidade; os **desenvolvedores** que tem como função organizar os papéis envolvidos principalmente no design e desenvolvimento de *software*; os **testadores**, que são responsáveis pelas atividades centrais do esforço de teste, que envolve conduzir os testes necessários e registrar os resultados destes testes; e os **gerentes** que organizam os papéis envolvidos principalmente no gerenciamento e na configuração do processo de engenharia de *software*. Existem também outros papéis envolvidos, como o **integrador** que fará a integração dos sistemas e subsistemas depois de desenvolvidos, e o **cliente** a quem o *software* será entregue.

Os artefatos considerados na modelagem do processo de desenvolvimento são: o **documento de arquitetura de *software***, o **modelo de design**, o **modelo de**

implementação, o **plano de integração** e o **plano de trabalho**. Cada um destes artefatos será explicado nas seções que seguem, conforme detalhado seu uso.

No processo de desenvolvimento de *software* escolhido são definidas três fases importantes para o desenvolvimento: a fase de **iniciação**, a fase de **elaboração** e a fase de **construção**. Existe ainda uma quarta fase, denominada de fase de **transição**, que não será detalhada neste trabalho, pois é responsável pela implantação do *software* junto à comunidade de usuários, ou seja, a etapa posterior à de desenvolvimento do *software*.

Workflows são comumente representados através de diagramas de atividades. Por isso, para a modelagem do sistema, foi utilizado como ferramenta de apoio o JUDE UML *Modeling Tool* [1], versão 3.2.1. Os principais elementos envolvidos na modelagem são: início e fim das execuções, atividades, conexões entre os elementos, construções em paralelo, elementos de junção e elementos de separação. Estes elementos, utilizados na modelagem, estão representados no modelo conforme a Fig. 21.

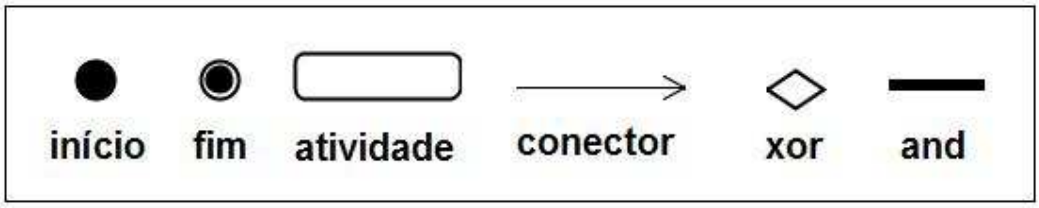


Figura 21: Elementos da modelagem do sistema

Nos itens a seguir será detalhada cada uma das fases de desenvolvimento, conforme o processo escolhido.

5.1 Fase de iniciação

Na fase de iniciação deve-se atingir o consenso entre todos os envolvidos sobre os objetivos do ciclo de vida do projeto. Esta é uma fase muito importante principalmente por que há muitos riscos de negócios e de requisitos que precisam ser tratados para o seguimento do projeto. Os principais objetivos da fase de

[1] O JUDE é um software para modelagem, livre e multiplataforma, e pode ser obtido em http://www.jude.change-vision.com

iniciação incluem: estabelecer o escopo do *software* do projeto; verificar os critérios de aceitação e o que deve ou não estar no produto; estimar o custo geral e a programação para o projeto, assim como as estimativas detalhadas para a próxima fase de elaboração. O fluxo de trabalho do desenvolvimento de *software* tem início no cliente, que primeiramente propõe o trabalho a ser analisado. O analista faz a proposta e entra em contato com o restante da equipe. Se a equipe aprovar, o analista envia o projeto para o cliente, caso contrário, a equipe retorna para o analista até que haja um consenso entre todos. Após todos os participantes concordarem, o projeto é emitido ao cliente para análise. Após a análise do cliente, novas modificações podem ser enviadas para o analista, ou então, caso o cliente aprove a proposta, conclui-se esta primeira fase e é dada seqüência a fase de elaboração. A Fig. 22 ilustra o fluxo de trabalho resumido da fase de iniciação.

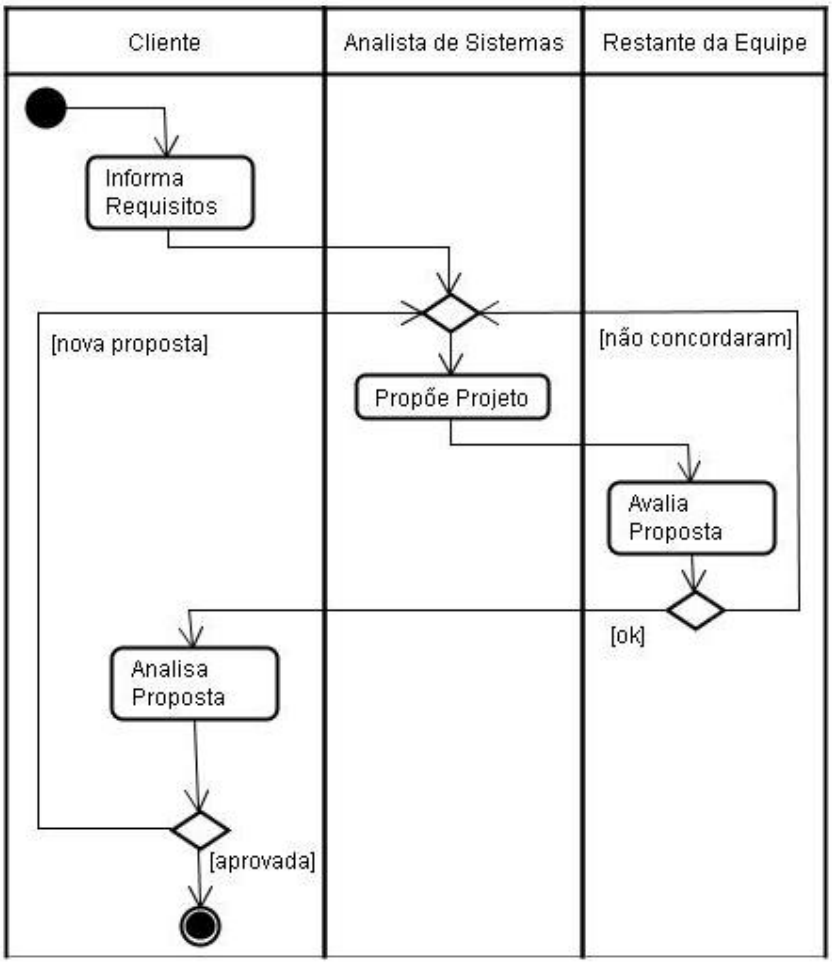


Figura 22: Fase de iniciação – Fluxo de trabalho resumido

Na Fig. 22 a fase de iniciação do projeto é apresentada de uma maneira bastante compacta, omitindo os detalhes do processo de desenvolvimento. Optou-se por esta representação apenas para uma melhor compreensão do funcionamento inicial. Nas seções a seguir serão apresentados os detalhes desta primeira fase de desenvolvimento, assim como todo o fluxo de trabalho dos atores envolvidos.

5.1.1 Seleção da equipe

Na fase de seleção da equipe é preciso selecionar uma equipe competente que se adapte aos requisitos necessários do *software* a ser desenvolvido. Sendo assim, o fluxo de trabalho inicia com a identificação de um gerente de projeto com a experiência e as habilidades apropriadas que será aprovado pelo comitê do projeto. É este gerente o responsável pela seleção dos demais membros da equipe. Por se tratar de desenvolvimento distribuído, muitas vezes o responsável por esta tarefa pode não possuir o devido conhecimento sobre todos os membros da equipe. É necessário entrar em contato com responsáveis por equipes que estão localizadas em regiões geograficamente distintas, ou consultar a base de dados que contém as informações referentes aos membros da organização, a fim de selecionar o pessoal que desempenhará as funções no desenvolvimento. Nesta fase, o grupo inicial de membros da equipe é formado pela equipe de planejamento do projeto. Este grupo, ainda pequeno, é identificado, aprovado e designado pelo gerente do projeto e, em geral, pode ser formado pelo gerente de projeto, arquiteto de *software*, analista de sistemas, chefe de desenvolvimento e chefe de teste. O processo da seleção de membros da equipe é ilustrado na Fig. 23.

Após o gerente selecionar um membro para a equipe o sistema verificará se ele está disponível, ou seja, não está trabalhando em muitos projetos simultaneamente. Se estiver disponível, é enviado um convite com as informações referentes ao projeto e o papel a ser desempenhado pelo participante. Caso não esteja disponível, o sistema avisará o gerente e recomendará que seja escolhida outra pessoa para desempenhar aquele papel. O gerente poderá então, retornar e escolher um novo participante, ou encerrar a equipe. Quando finalmente a equipe estiver completa, o projeto passa a fase seguinte.

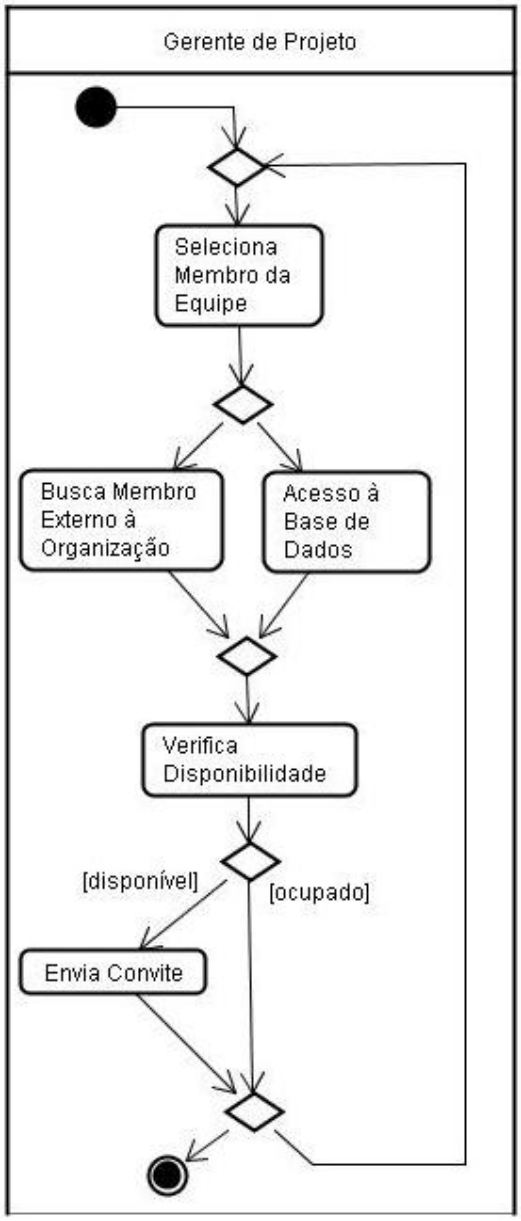


Figura 23: Seleção da equipe

Após a equipe inicial ter sido selecionada, o gerente de projeto designará as tarefas a cada membro escolhido para compor a equipe. Deverão ser discutidas junto ao cliente as necessidades básicas do sistema, estabelecer o escopo do *software* e o que deve ou não estar no produto. Como mencionado na seção 5.1, deve haver um consenso entre todos os participantes para a continuidade do projeto. Nesta fase inicial pode ainda ser desenvolvido um modelo, simulando o que é exigido, ou ainda um protótipo inicial que explora as áreas consideradas de alto

risco. O desenvolvimento do protótipo nesta fase de iniciação deve se limitar a ganhar confiança, visando a possibilidade de uma solução. A solução será executada durante as fases de elaboração e construção.

5.2 Fase de elaboração

Esta fase do processo de desenvolvimento deve ser focada nos riscos técnicos e arquiteturais. Deve ser feita uma revisão no escopo e os requisitos devem estar bem mais compreendidos. Deve também ser fornecida uma base estável para o esforço da fase de construção. Após um exame dos requisitos mais significativos, ou seja, aqueles que têm um maior impacto na arquitetura do sistema, e uma avaliação de risco, é desenvolvida a arquitetura inicial. Através de um ou mais protótipos é analisada a estabilidade da arquitetura. Para a construção do protótipo é dada uma atenção maior a parte arquitetural do sistema, ou seja, o desenvolvimento é direcionado às atividades de design, sendo pouca atenção dada aos detalhamentos das classes e seus atributos. Durante esta etapa, o esforço maior é da equipe de arquitetura e de uma equipe designada para a criação do protótipo, geralmente composta pelos programadores mais experientes. Neste ponto têm-se uma pequena equipe de design utilizando-se de mecanismos e tecnologias genéricos. O grupo de teste concentra-se na criação do ambiente de teste e no teste dos casos de uso iniciais. Um artefato bastante importante nesta fase de elaboração é o documento de arquitetura de *software*, desenvolvido pelo arquiteto de *software* que captura as decisões mais importantes do projeto. Este documento serve como um meio de comunicação entre o arquiteto de *software* e os demais membros da equipe. Nele constam todas as decisões tomadas a respeito do projeto.

Todo o fluxo de trabalho do desenvolvimento do protótipo é ilustrado na Fig. 24 e detalhadamente explicado na seção 5.2.1.

5.2.1 Criação do protótipo

Após a análise dos requisitos do sistema e a seleção da equipe de desenvolvimento inicial desta fase de elaboração, dá-se início à criação do protótipo. O fluxo de trabalho da criação do protótipo tem início na equipe de arquitetura que primeiramente desenvolve uma arquitetura inicial e, após analisada pelo arquiteto de

s*oftware*, é devidamente registrada no documento de arquitetura de s*oftware*. Neste documento constam todas as decisões tomadas a respeito do projeto, bem como os detalhes do desenvolvimento. Após a equipe de arquitetura concluir sua tarefa, é enviada uma atividade para a lista de trabalho (detalhada na seção 6.1) da equipe responsável pelo protótipo, informando que a primeira equipe finalizou seu trabalho e liberando os desenvolvedores do protótipo a começarem o desenvolvimento. Quando o protótipo estiver pronto, é encaminhado para a equipe de teste que efetuará todos os testes necessários. Se a equipe de teste, após concluir seus testes iniciais, detectar erros ou falhas, encaminhará para a equipe que desenvolveu o protótipo, a fim de que sejam resolvidos os problemas. Então um novo ciclo de desenvolvimento inicia até que não haja mais erros. Quando o protótipo finalmente estiver pronto, sem erros detectados, será encaminhado para o cliente testar e verificar se está de acordo com o projeto inicial. Se for solicitada alguma modificação no protótipo, será encaminhada à equipe de desenvolvimento. Quando o projeto finalmente estiver de acordo com o propósito inicial, ou seja, o protótipo estiver funcionando, testado, sem erros e conforme o cliente solicitou, encerra-se esta fase de elaboração.

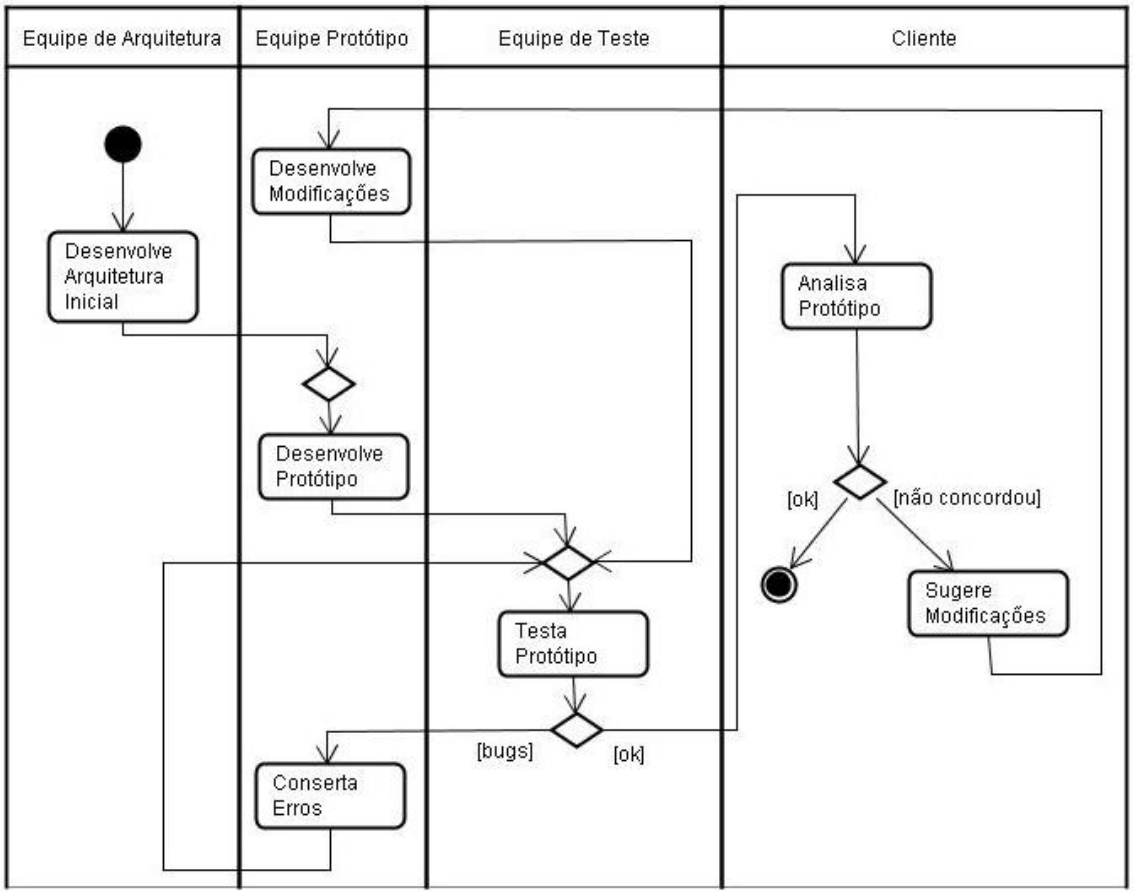


Figura 24: Desenvolvimento do protótipo

Considerando-se o desenvolvimento distribuído, cada uma destas equipes pode estar localizada em diferentes espaços físicos, ou até mesmo em países diferentes. Por esta razão o controle das equipes e da distribuição do trabalho através do *workflow* torna-se uma importante ferramenta para o controle sobre cada uma das equipes, mantendo o gerente e demais membros da organização sempre informados sobre o andamento do projeto.

5.3 Fase de construção

A meta desta fase de construção é esclarecer os requisitos restantes e concluir o desenvolvimento do sistema, baseado na arquitetura desenvolvida na fase de elaboração (seção 5.2). Nesta fase a ênfase está no gerenciamento de recursos e controle das operações, a fim de otimizar custos, programações e qualidade.

Neste sentido o gerenciamento passa por uma transição do desenvolvimento ocorrido durante as fases de iniciação e de elaboração, para o desenvolvimento dos produtos que podem ser implantados durante a construção e transição. É na fase de construção que a maior parte do trabalho será realizada. Geralmente uma equipe de construção responsabiliza-se por um ou mais subsistemas, sendo que a inclusão de novos subsistemas provoca mudanças arquiteturais e necessitam ser cuidadosamente consideradas. Subsistemas podem ser definidos como um conjunto de componentes de outros subsistemas de implementação, usado para estruturar o modelo de implementação, dividindo-o em partes menores. No subsistema uma equipe pode ter uma liberdade relativa para projetar e implementar quando considerar necessário. No entanto, é fundamental a comunicação entre equipes a fim de garantir que elas não estejam criando os mesmos mecanismos de implementação de forma paralela.

5.3.1 Etapas da fase de construção

Esta fase de construção pode ser dividida em cinco etapas: a estrutura de um **modelo de implementação**, o **planejamento da integração**, a **implementação dos componentes**, a **integração dos componentes** e por fim a **integração do sistema**. Nos itens a seguir será detalhada cada uma destas etapas.

5.3.1.1 Modelo de implementação

Primeiramente nesta fase de construção é estruturado, pelo arquiteto de *software*, o modelo de implementação, que estabelece a estrutura da implementação e atribui responsabilidades para subsistemas de implementação e seu conteúdo. Este modelo é um conjunto dos componentes e dos subsistemas de implementação que os contém. Estes componentes incluem componentes de produtos liberados, como executáveis, e componentes a partir dos quais estes produtos são criados, como arquivos de código-fonte. A transição do espaço de design para o espaço de implementação começa com o espelhamento da estrutura do modelo de design no modelo de implementação. O modelo de design é uma abstração da implementação do sistema, também desenvolvido pelo arquiteto de *software*. É um artefato composto e abrangente que envolve todas as classes de design, subsistemas,

pacotes, colaborações e os relacionamentos entre eles, sendo usado para conceber e documentar o design do sistema de *software*. A Fig. 25 ilustra como é desenvolvido o modelo de implementação.

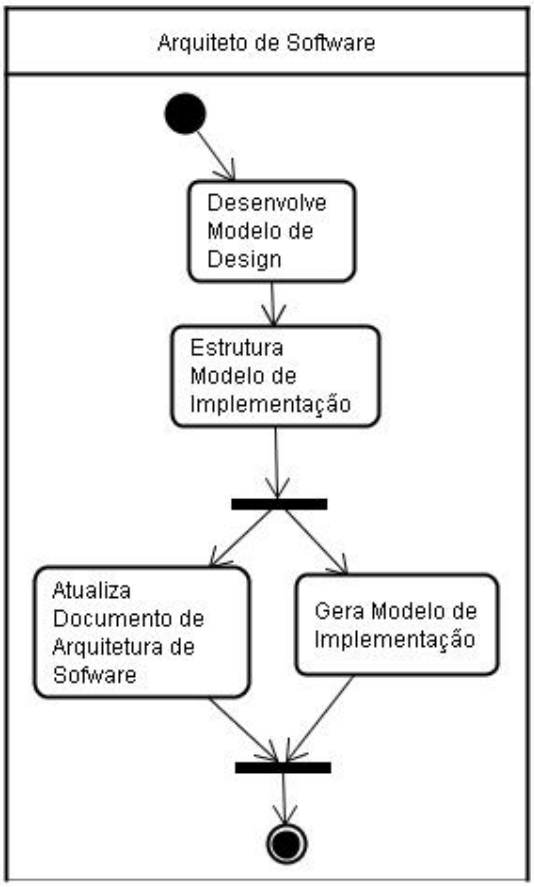


Figura 25: Modelo de implementação

Inicialmente o arquiteto de *software* estrutura o modelo de implementação baseado no modelo de design previamente desenvolvido. No modelo de implementação é estabelecida a estrutura em que a implementação residirá e atribuídas as responsabilidades para subsistemas de implementação e seu conteúdo. Após a estrutura inicial, o modelo de implementação finalmente é desenvolvido e o documento de arquitetura de *software* é atualizado.

5.3.1.2 Planejamento da integração do sistema

Após estruturado o modelo de implementação, deve-se planejar a integração do sistema, que tem por finalidade planejar os subsistemas que devem ser implementados e a ordem em que devem ser integrados na iteração atual. Deve ser verificado se a ordem da integração facilita a localização de erros e se está associada à ordem em que os componentes são desenvolvidos. Em sistemas grandes, onde podem ocorrer centenas de subsistemas de implementação, planejar a integração é uma tarefa bastante complexa. A Fig. 26 ilustra o processo de planejamento de integração do sistema.

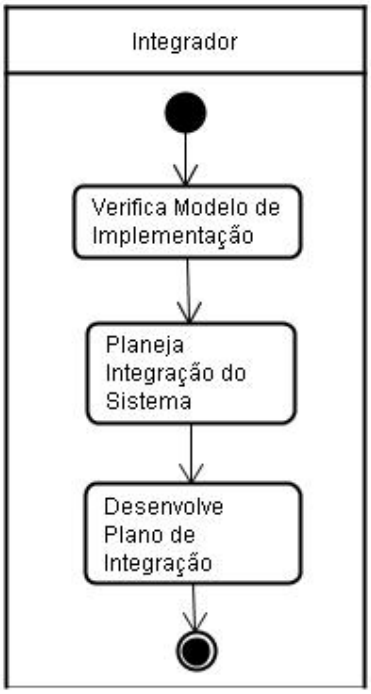


Figura 26: Planejamento da integração do sistema

O integrador verifica o modelo de implementação já desenvolvido, explicado no item 5.3.1.1, e faz o planejamento da integração do sistema. Todo o planejamento é registrado no artefato plano de integração que tem por finalidade definir a ordem em que os componentes e os subsistemas devem ser implementados. Quando for verificado que a ordem da integração facilita a

localização de erros e esteja associada à ordem em que os componentes sejam desenvolvidos o processo está concluído.

5.3.1.3 Implementar componentes

Esta atividade tem como finalidade principal produzir o código-fonte de acordo com o modelo de design proposto. Os implementadores escrevem o código-fonte, adaptam os componentes existentes, compilam, etc. Os implementadores também consertam defeitos de código e realizam testes unitários para verificar as mudanças. Em seguida o código é revisado para avaliar a qualidade e a compatibilidade com o modelo proposto. O implementador pode utilizar-se de um repositório de dados que contêm componentes reusáveis, com o propósito de diminuir a quantidade de código a ser desenvolvida. O reuso de código diminui também os custos e o tempo de desenvolvimento, pois, supõe-se que o trecho de código a ser utilizado já foi devidamente testado e está funcionando, necessitando apenas de uma adaptação conforme a necessidade do implementador. Quando o componente estiver pronto serão feitos testes unitários, a fim de verificar os erros de desenvolvimento. Os testes serão executados até que não existam mais erros no código. Quando o componente estiver funcionando, o código-fonte é analisado e verificado se está de acordo com o modelo proposto. Caso necessite alguma modificação no código, é feita uma reestruturação e um novo ciclo de testes é iniciado. Quando o componente estiver pronto, testado e funcionando será enviado para uma área de trabalho compartilhada, ficando disponível para que outros implementadores utilizem-no, caso haja uma dependência. Por fim todos os subsistemas serão depositados nesta área de trabalho para que possam ser integrados para compor o sistema completo. A Fig. 27 ilustra o processo de implementação de componentes.

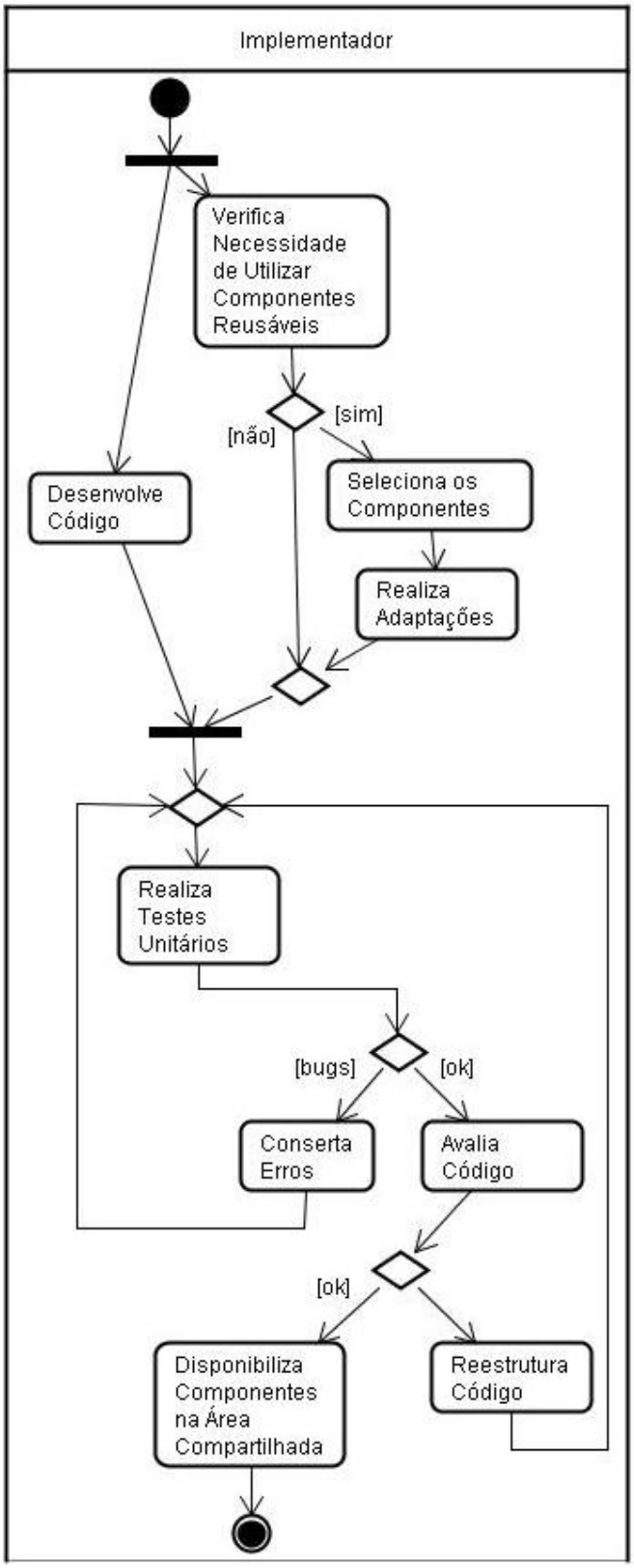


Figura 27: Implementação dos componentes

5.3.1.4 Integração dos componentes

Nesta etapa é feita a integração de cada componente do subsistema, ou seja, os componentes desenvolvidos por cada implementador, ou equipe de implementadores são integrados e formam o subsistema. Após a execução dos testes, o subsistema de implementação é liberado no espaço de trabalho de integração do sistema. Se uma equipe com diversos membros estiver trabalhando de forma paralela no mesmo subsistema, é importante o compartilhamento freqüente dos resultados obtidos, a fim de estarem sempre informados sobre o andamento do trabalho. A Fig. 28 ilustra o processo de integração dos componentes.

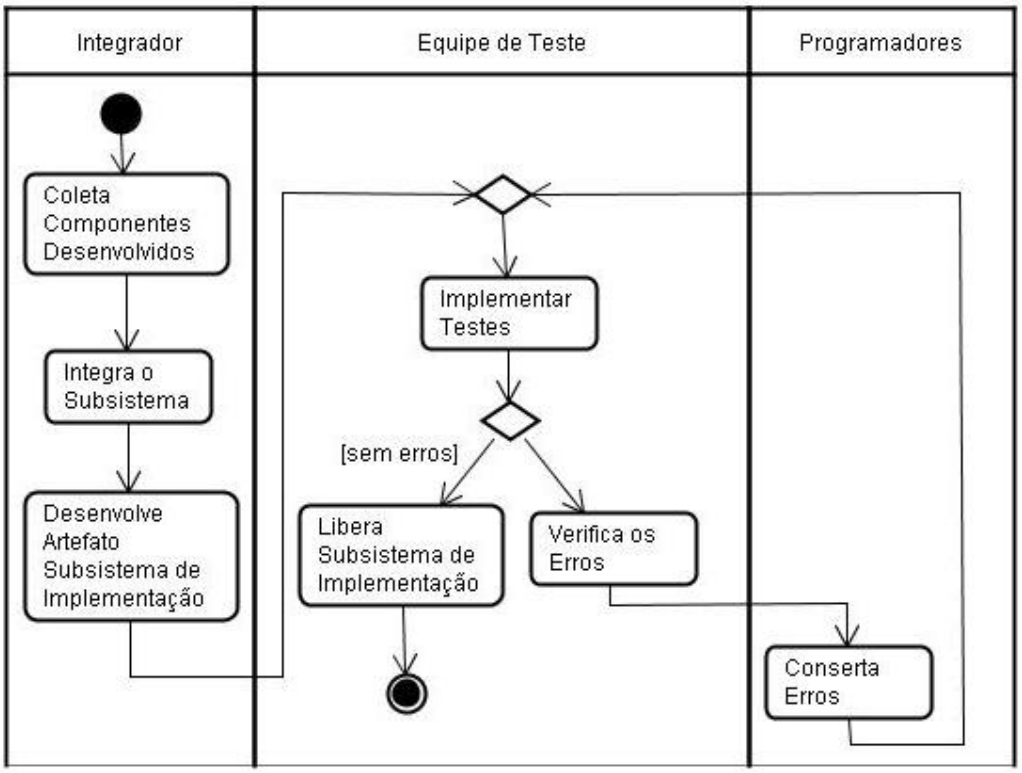


Figura 28: Integração de componentes

O integrador utiliza-se dos componentes liberados pelos implementadores e realiza a integração do subsistema. Após a integração, é desenvolvido o artefato subsistema de implementação e enviado para a equipe de testes. Os testes são implementados e, se não forem detectados erros, o subsistema de implementação é liberado no espaço de integração dos sistemas para que possa juntar-se com outros

subsistemas liberados por outras equipes. Por fim, quando todos os subsistemas estiverem liberados na área compartilhada, poderão ser integrados, formando o sistema completo.

5.3.1.5 Integração do sistema

Esta etapa tem por finalidade a integração dos subsistemas de implementação por partes. Os subsistemas, liberados no espaço de trabalho, são adicionados e, cada integração é testada por uma equipe de teste. Ao final, após o último incremento, o sistema pode ser completamente testado. A Fig. 29 ilustra o processo de integração de sistemas.

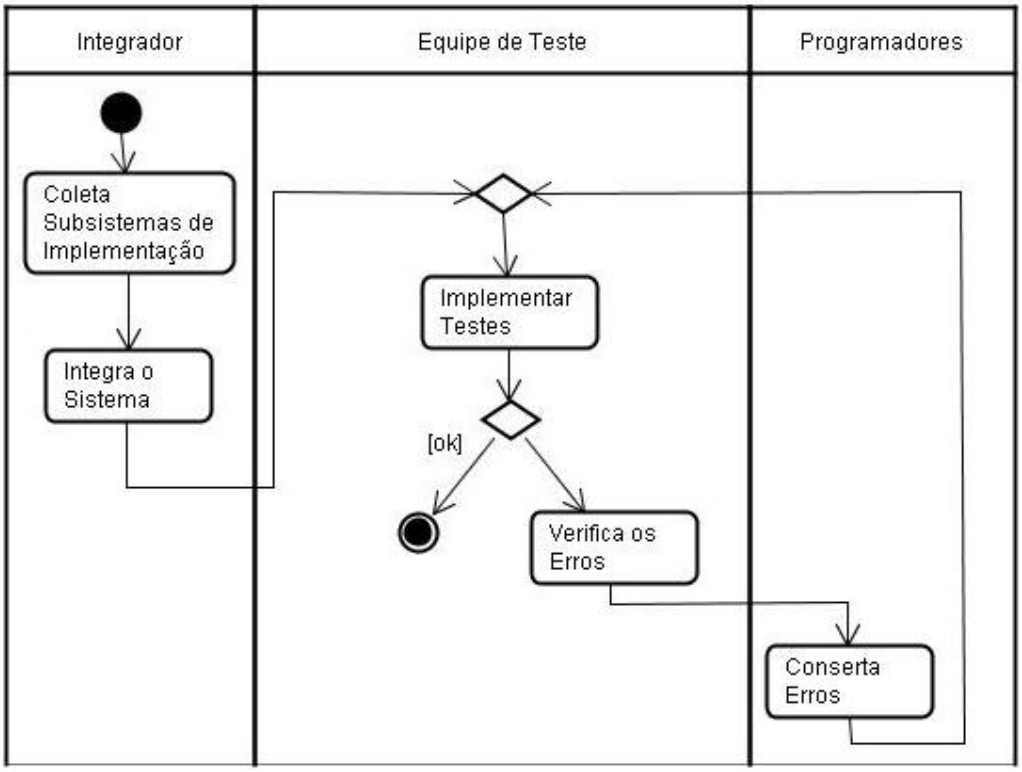


Figura 29: Integração do sistema

O integrador faz a integração dos sistemas utilizando-se dos subsistemas liberados previamente pelas equipes de desenvolvimento. Com o sistema integrado a equipe de testes começa a implementar todos os testes necessários até que o *software* esteja pronto. Se forem encontrados erros durante esta etapa de testes, a

equipe testadora entra em contato com a equipe responsável por este determinado erro, a fim de consertar o problema. Quando a equipe de teste não encontrar mais falhas, o *software* está pronto para a fase de implantação, onde será entregue ao cliente.

6 AMBIENTE DE DESENVOLVIMENTO DISTRIBUÍDO

No ambiente de desenvolvimento distribuído proposto existe uma área de desenvolvimento privada, ou seja, cada membro da equipe pode efetuar mudanças em artefatos, sem que elas se tornem imediatamente visíveis para os demais membros da equipe. Quando um desenvolvedor ou uma equipe concluir sua parte no trabalho, disponibilizará o que foi feito em uma área de trabalho compartilhada, para que outras equipes possam utilizar-se dos componentes do sistema que estão disponíveis. Um componente pode ser considerado um artefato e representa um trecho de código de *software* ou um arquivo contendo informações. Pode também ser uma agregação de outros componentes, como por exemplo, um aplicativo composto de vários executáveis. Existem restrições sobre quem pode visualizar ou modificar determinado artefato, definidas pela política do projeto. Este processo de dependência entre componentes deve ser tratado com muita atenção, principalmente na fase de construção. Por exemplo, uma dependência de um componente A em relação a um componente B indica que o componente A possui uma dependência de compilação ou de tempo de execução em relação a B. No desenvolvimento distribuído, com as equipes distantes fisicamente, este problema torna-se um agravante ainda maior.

Um artefato muito importante na fase de construção, responsável pela organização destas dependências entre componentes e atividades, é a ordem de trabalho. A ordem de trabalho é o meio pelo qual o gerente de projeto comunica à equipe responsável o que deve ser feito e quando deve ser executada determinada tarefa. A ordem passa a ser um contrato interno entre o gerente de projeto e os demais membros da equipe. É através dela que o gerente distribui as funções e atividades que deverão ser executadas por cada equipe, bem como a ordem das tarefas, cronograma com início e término estimados, horas de trabalho da equipe e todos os detalhes do que deve ser executado. Em projetos pequenos com equipes localizadas no mesmo espaço físico, esta seria uma tarefa bastante simples, pois o projeto poderia ser discutido entre todos os membros, com as tarefas entregues pessoalmente para cada desenvolvedor. Quando o desenvolvimento envolve

equipes que trabalham de forma distribuída, existe uma intensa necessidade de automatizar este processo, a fim de facilitar a distribuição das tarefas.

Baseado nisso, foi planejada uma área no ambiente proposto responsável pela automação deste processo. A área privada do ambiente de desenvolvimento distribuído proposto, quando acessada por um participante, contém uma lista, composta pelas atividades que ele deve executar em seqüência. Se houver dependências, quando uma atividade que está sendo executada por uma equipe A é finalizada, será enviada para a lista dos participantes da equipe B, até que ela seja executada. Estas atividades, conforme executadas, são automaticamente eliminadas. O funcionamento desta parte do ambiente será explicado no item a seguir.

6.1 Uso das listas de trabalho

Primeiramente ocorre a distribuição do trabalho por parte do gerente, registrando todo este processo no artefato ordem de trabalho. Então, com todas as tarefas distribuídas, dá-se início ao desenvolvimento. A Fig. 30 ilustra como cada participante desenvolve suas atividades de forma seqüencial e a maneira como é distribuída cada atividade de dependência. Nela está representado o processo de execução das tarefas na forma de um diagrama de atividades, ou seja, a Fig. 30 não representa um *workflow*. De acordo com a figura, no momento do acesso, o membro da equipe A verifica sua lista de atividades que contém as atividades a serem desenvolvidas. Conforme cada atividade é desenvolvida, vai sendo liberada no sistema e, se houver alguma dependência por parte de outra equipe de desenvolvimento, o sistema enviará a tarefa para a lista de atividades desta outra equipe, para que possa ser executada. As tarefas são executadas, liberadas no sistema e enviadas para as respectivas equipes que necessitam da liberação de uma atividade para que possam continuar o trabalho.

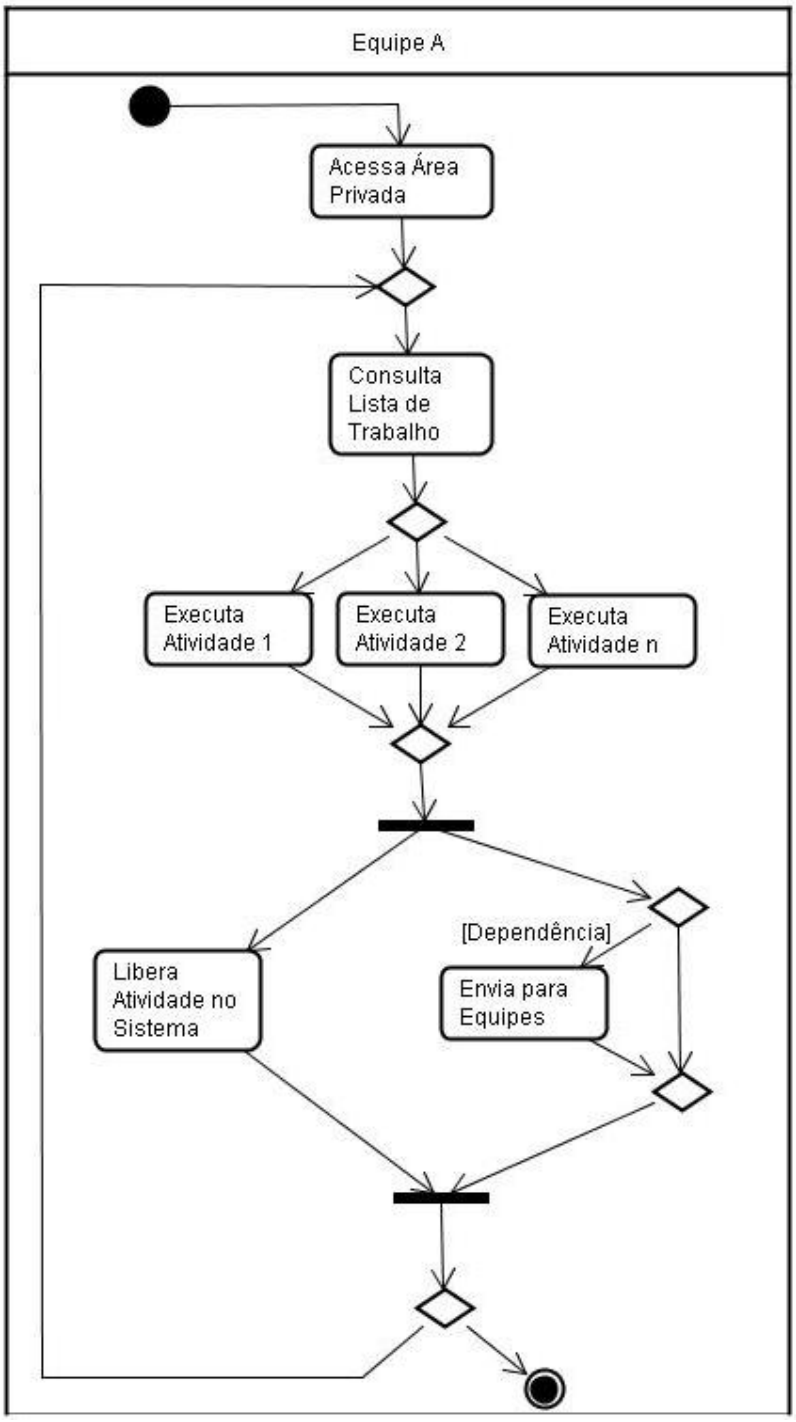


Figura 30: Diagrama de atividade – Listas de atividades

6.2 Interface do sistema

Este item tem como finalidade principal mostrar algumas das interfaces do ambiente de desenvolvimento distribuído e as maneiras como os usuários podem interagir através delas. Para o desenvolvimento das interfaces optou-se por utilizar a formatação de textos do tipo *HyperText Markup Language* - HTML, sendo, desta forma, facilmente disponibilizada na WEB, facilitando o acesso de qualquer localidade. Além disso, esta escolha torna fácil a inserção de *scripts* que acessam e manipulam bases de dados. Optou-se também por uma interface bem simples, minimizando assim perdas de rendimento ao ser carregada. A idéia principal deste item é mostrar como seria a interação do usuário com o ambiente através da modelagem desenvolvida no capítulo 5, para uma melhor compreensão do modelo proposto. A Fig. 31 mostra a interface de acesso, ou seja, a tela inicial do sistema.

Figura 31: Tela de acesso ao sistema

O usuário acessa o sistema através do *login* e senha previamente cadastrados em uma base de dados e é automaticamente reconhecido no momento do acesso. Alguns usuários têm privilégios de acesso, como o gerente, por exemplo, que pode convocar uma equipe ou acessar informações na base de dados que não estão disponíveis a todos os membros da organização. Desta forma, será mostrada

na Fig. 32 a interface de acesso com alguns dos recursos disponíveis, considerando que o gerente de projetos acessou o sistema.

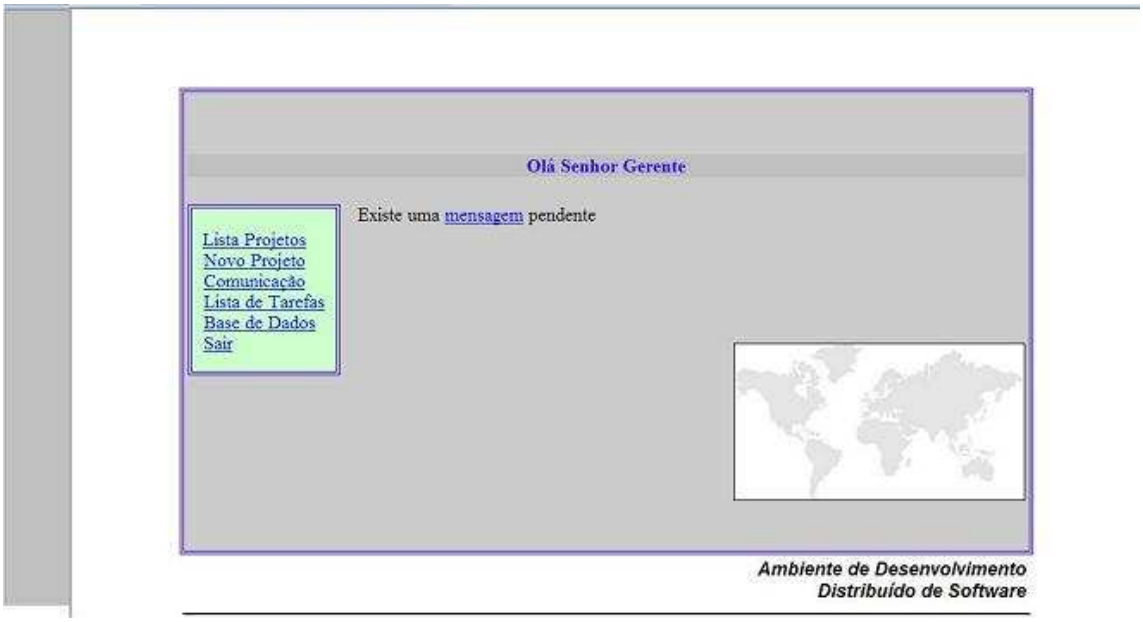


Figura 32: Tela de opções do gerente de projetos

Quando o gerente acessa o sistema pode listar os projetos em que está envolvido, iniciar um novo projeto, acessar uma área de comunicação ou ainda acessar a base de dados. É possível também receber as mensagens enviadas por outros participantes, através do ambiente. Na tela de acesso o gerente recebe o alerta de que uma nova mensagem foi recebida. Ao selecionar a mensagem, será carregada uma tela com a mensagem e as opções de responder ou excluir, conforme a Fig. 33.

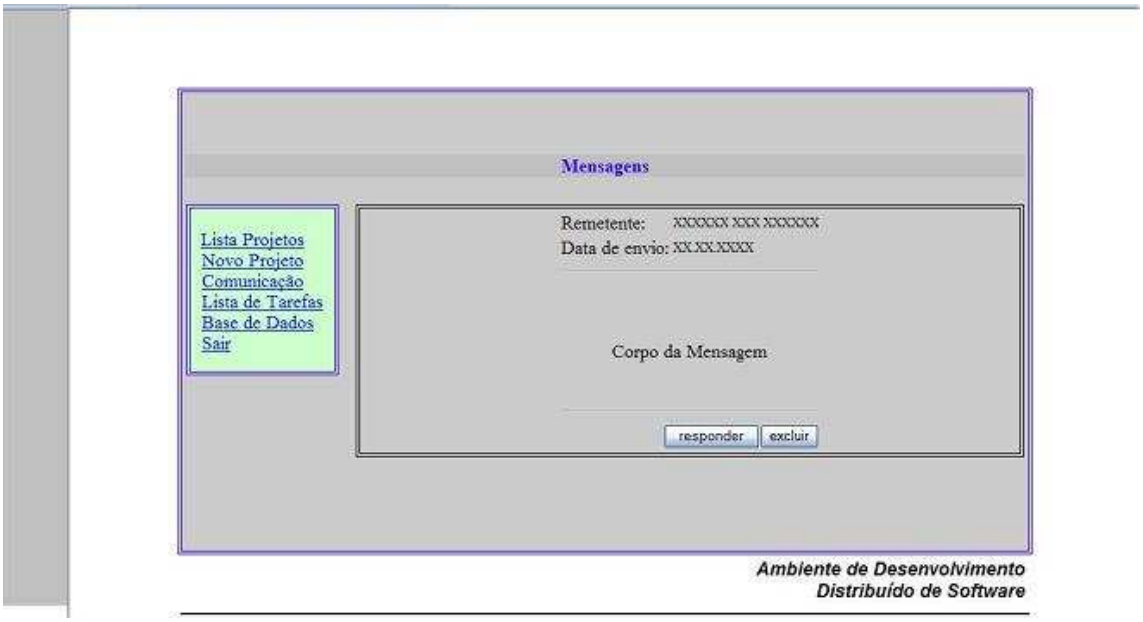


Figura 33: Tela de mensagens

Quando a opção Lista Projetos, no menu à esquerda, é escolhida, todos os projetos que o gerente estiver trabalhando serão listados conforme mostra a Fig. 34.

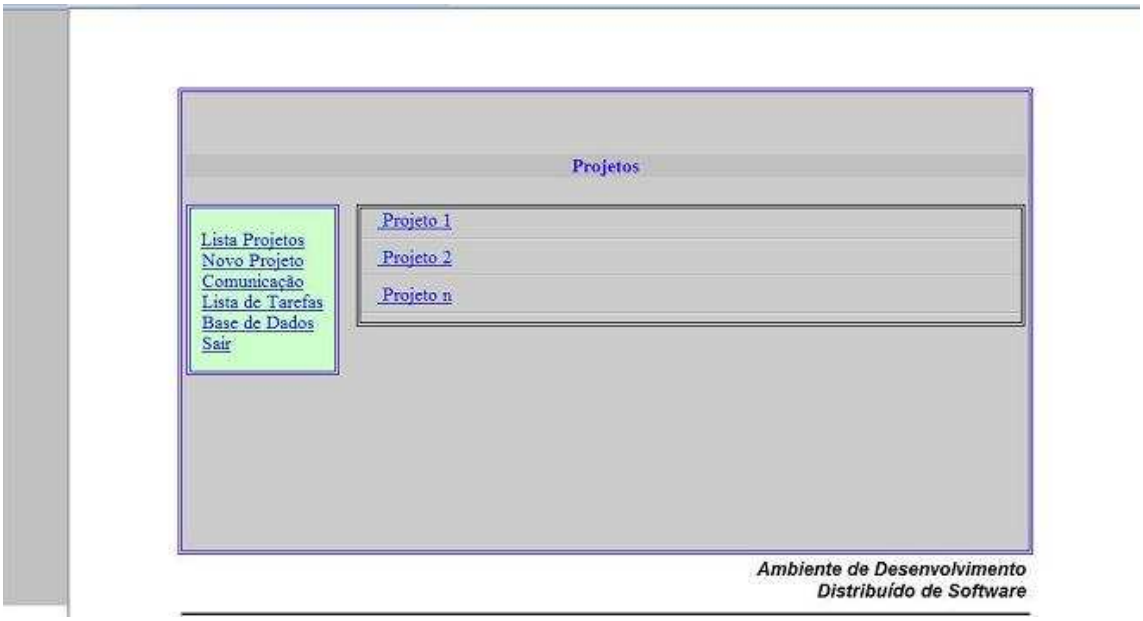


Figura 34: Tela de listagem dos projetos

Ao escolher um projeto, todas as informações referentes ao projeto escolhido estarão disponíveis para visualização, como: detalhes do projeto, prazo de entrega, andamento do trabalho e a equipe responsável, conforme mostra a Fig. 35. Existe também uma área de gerência, onde é possível entrar em contato com membros da equipe ou verificar o andamento do trabalho, por exemplo.

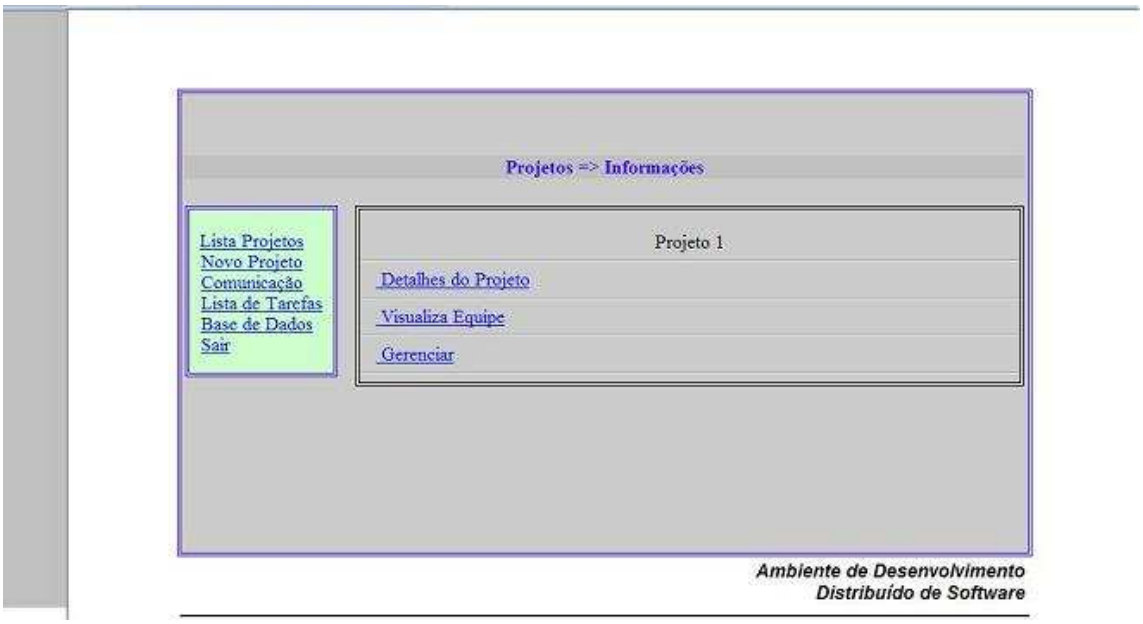


Figura 35: Tela informações do projeto

Se a opção escolhida for Novo Projeto, o gerente poderá iniciar um novo projeto, escolhendo a equipe e cadastrando todos os detalhes. A Fig. 36 mostra a interface desta parte do sistema.

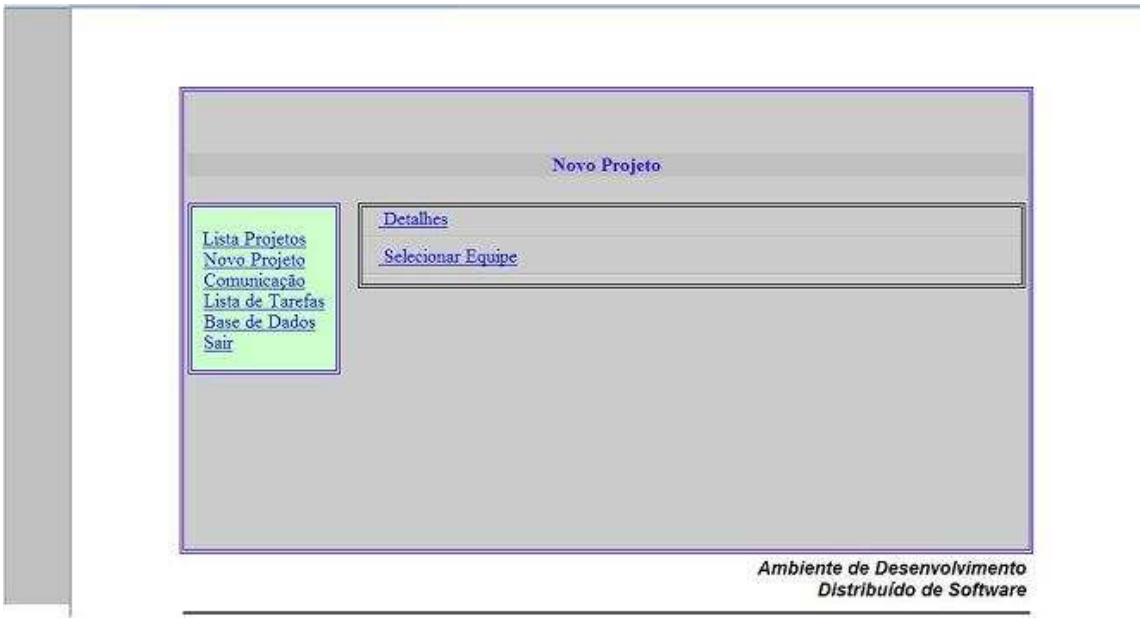


Figura 36: Tela novo projeto

Ao selecionar a equipe para um novo projeto o gerente pode utilizar a base de dados da organização, a fim de encontrar desenvolvedores que preencham os requisitos para o desenvolvimento do sistema, conforme visto no item 5.1.1. A Fig. 37 mostra a tela de seleção da equipe por parte do gerente de projetos. Como exemplo a figura mostra que foi selecionado a área de atuação do desenvolvedor, ou seja, serão listados apenas aqueles participantes que trabalham na área de banco de dados. E, após a listagem dos participantes encontrados, o gerente poderá selecionar um membro para a equipe.

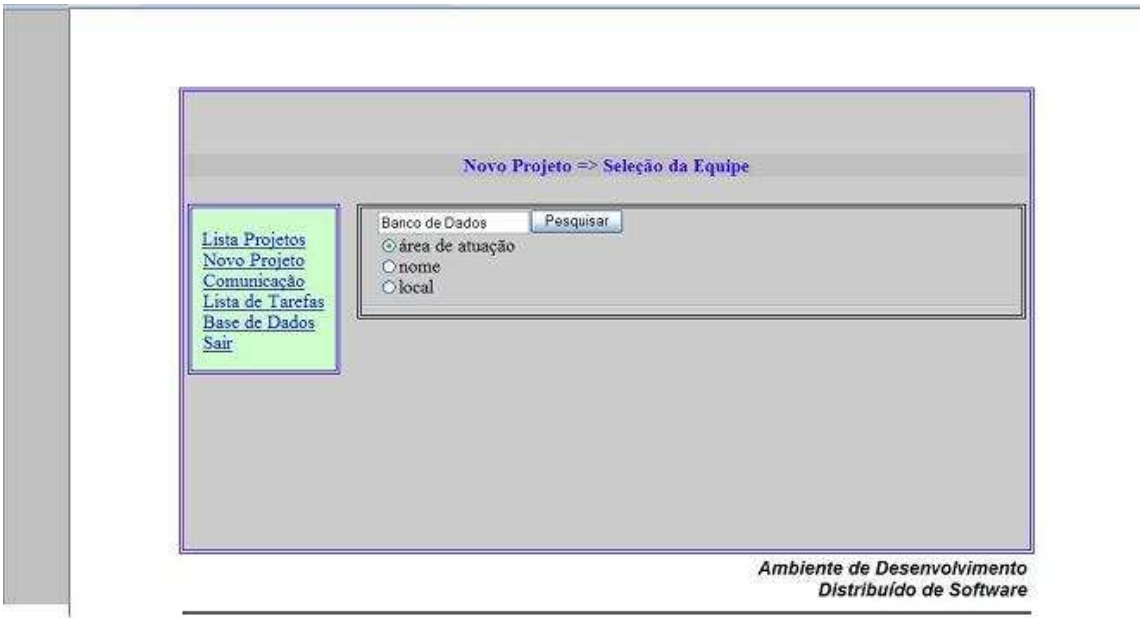


Figura 37: Tela seleção da equipe

Se for escolhido um desenvolvedor que já estiver trabalhando em um projeto, será identificado pelo sistema, pois todas as informações devem ficar armazenadas na base de dados. O gerente será informado e recomendando a selecionar outro membro, conforme mostra a Fig. 38.

Figura 38: Tela resultado da busca

Conforme o gerente vai selecionando a equipe, vai atribuindo a cada participante o seu papel dentro do projeto. O gerente é o responsável pela distribuição das tarefas a cada membro da equipe. Com a equipe selecionada ele pode fazer a atribuição de cada tarefa ligada a cada papel. Conforme as tarefas vão sendo desenvolvidas, os responsáveis pelos outros papéis também podem enviar uma atividade para outra equipe. Por exemplo: o gerente envia uma atividade para uma equipe de programadores. Quando esta equipe concluir seu trabalho, é enviada para a lista de trabalho da equipe responsável pelos testes, uma atividade, informando que foi concluída a etapa de desenvolvimento. Inicia-se então a fase de testes. E, se houver problemas, o chefe da equipe de teste enviará uma tarefa de volta a equipe de programadores para que sejam consertados os *bugs*. A Fig. 39 mostra a tela de definição de uma atividade.

Figura 39: Tela de definição de atividades e associação a seu executor

Quando o usuário que está trabalhando em sua área de trabalho, no ambiente de desenvolvimento, acessa o *link* Lista de Tarefas no menu à esquerda, tem acesso a todas as atividades que foram dirigidas a ele, conforme mostra a Fig. 40. Este usuário pode, então, selecionar uma dentre as atividades na sua lista de trabalho e tomar conhecimento sobre as instruções para sua realização. Ao terminar a tarefa, o usuário deve informar ao sistema sobre sua realização.

Figura 40: Tela listas de trabalho

Existe também, nas opções do menu à esquerda, um canal de comunicação. A comunicação entre as equipes é uma das dificuldades encontradas no desenvolvimento de *software*. No desenvolvimento distribuído essa comunicação torna-se uma dificuldade ainda maior, devido a diversos fatores mencionados no capítulo 3. Portanto é muito importante uma comunicação bem feita entre os membros das equipes, principalmente entre os gerentes, para um maior controle e gerencia das atividades. Acessando esta opção de comunicação, o gerente pode comunicar-se de maneira síncrona com outros membros que estejam acessando o sistema, ou seja, é possível selecionar um participante que esteja acessando o sistema ao mesmo tempo e manter uma conversa em tempo real. Porém, como as equipes podem estar em localidades com diferentes fusos horário, muitas vezes esta comunicação pode não ocorrer de maneira síncrona. É possível então, selecionar uma pessoa e enviar uma mensagem, que será mostrada na tela inicial no instante em que esta pessoa acessar o sistema, conforme visto na Fig. 32. Inicialmente é feita uma busca pelo nome da pessoa e, após selecionar para quem deve ser enviada a mensagem, será carregado um formulário, onde a mensagem será redigida e enviada ao destinatário conforme a Fig. 41.

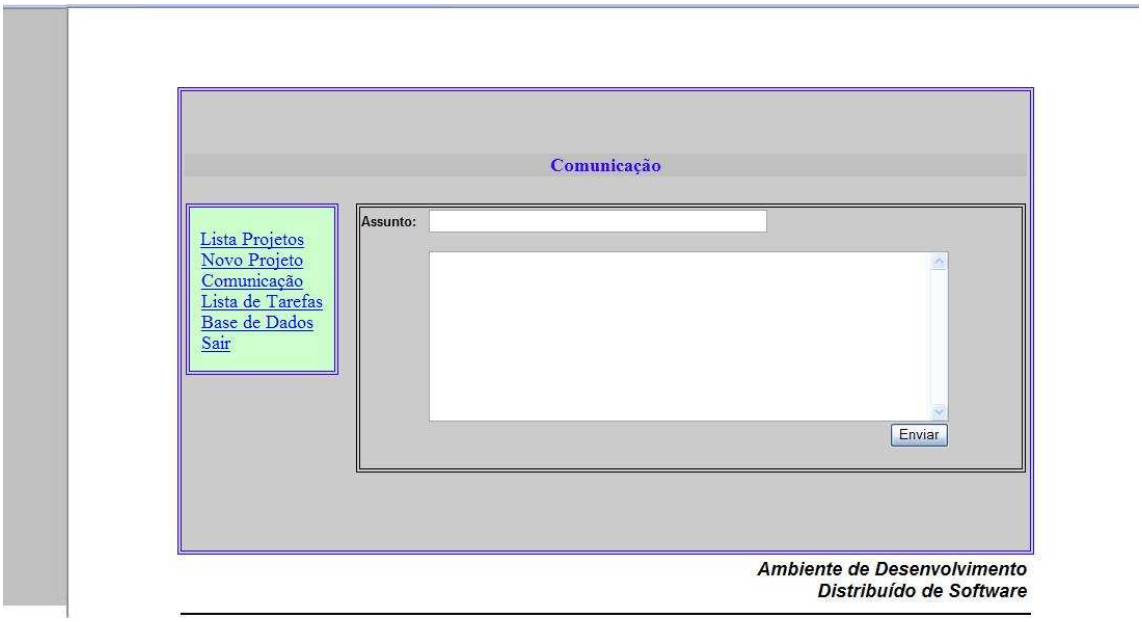


Figura 41: Tela de envio de mensagens.

Neste capítulo foi demonstrado, através do modelo das interfaces, o funcionamento do modelo proposto de um ambiente de desenvolvimento de *software* com distribuição assíncrona de atividades, a fim de validar os diagramas de *workflows* propostos. No capítulo 7 serão apresentadas as conclusões sobre este trabalho, bem como as dificuldades encontradas e os trabalhos futuros que podem ser desenvolvidos a partir deste.

7 CONCLUSÃO

Percebe-se com este trabalho que uma modelagem dos processos de *workflow* em um ambiente de desenvolvimento distribuído de *software*, não só facilita a distribuição das tarefas por parte do gerente como melhora a comunicação e a interação entre as equipes, mantendo os processos que necessitam de um grande índice de repetições, de forma automatizada. Percebeu-se também a importância de um processo de desenvolvimento de software - PDS bem definido, e a necessidade de um processo padrão dentro de uma empresa de desenvolvimento.

Verificou-se que não existe um único PDS que satisfaça todos os requisitos de um sistema, ou um único processo para qualquer tipo de sistema. Porém, é importante que haja uma padronização para que o desenvolvimento ocorra de uma maneira cada vez mais sólida, necessitando apenas adaptar o processo padrão as reais necessidades do projeto.

Foram apresentadas as principais vantagens do desenvolvimento distribuído de *software* – DDS e as motivações que levam uma empresa de desenvolvimento a distribuir suas equipes em busca dos melhores recursos globais. Mas, nota-se que ainda existem muitas lacunas a serem preenchidas e existem também algumas dificuldades a serem superadas. Foram apresentadas também as principais dificuldades encontradas pelos gerentes de projetos em acompanhar o andamento do trabalho, em definir a equipe e distribuir as tarefas para cada equipe distribuída.

Além disto, foram apresentados alguns conceitos relacionados a *workflow*, como: os diferentes tipos e os principais componentes de um *workflow*; papéis, atividades, artefatos e listas de trabalho; o funcionamento de um sistema de gerenciamento de *workflow* – WfMS, segundo o modelo de referência da *Workflow Management Coalition* – WfMC; o uso das *worklists* como forma de auxílio na distribuição das tarefas, visto que os PDS possuem uma intensa distribuição de atividades durante todo o ciclo de vida do *software*. O uso das *worklists* facilita o desenvolvimento das atividades, mantendo o trabalho sempre de forma ordenada e com um melhor controle sobre as tarefas. Qualquer participante do *workflow* tem um

completo acompanhamento sobre quais atividades já foram executadas e quais estão sendo executadas no momento.

Percebe-se que o momento de seleção da equipe é muito importante para a continuidade do projeto, pois em um ambiente de desenvolvimento distribuído de *software*, as dependências entre as equipes podem acarretar graves problemas de continuidade. É importante diminuir ao máximo as dependências entre equipes que estão em localidades diferentes, evitando que equipes desenvolvam o mesmo componente duas vezes, ou acabem utilizando versões de componentes não atualizadas. A comunicação é um dos fatores mais prejudicados no DDS, por isso é importante um canal de comunicação bem estruturado entre as equipes através do ambiente, mantendo todos os participantes informados sobre o andamento do projeto.

Devido a diversos fatores, citados durante o trabalho, verificou-se que o uso da tecnologia de *workflow* no controle dos PDS, em um ambiente de desenvolvimento, é uma importante ferramenta de apoio as equipes que trabalham de maneira distribuída.

Foram encontradas algumas dificuldades durante a realização deste trabalho. A principal dificuldade encontrada foi a falta de um processo padrão, voltado para o desenvolvimento distribuído, visto que ainda não foi definido um padrão de desenvolvimento de *software* a ser adotado por empresas que desenvolvem *software* comercial de maneira distribuída.

Alguns trabalhos podem ser desenvolvidos futuramente, a partir deste trabalho de conclusão de curso, como: a implementação de uma ferramenta baseada no modelo de *workflow* proposto, para que possa ser utilizada nas organizações que trabalham com desenvolvimento distribuído, a fim de facilitar o trabalho por parte das equipes; e é interessante também que uma padronização nos processos de desenvolvimento distribuído de *software* seja criada, não apenas para facilitar o desenvolvimento de *software*, mas para que mais trabalhos como este possam ser desenvolvidos, visando enriquecer cada vez mais a quantidade de material referente a este assunto.

REFERÊNCIAS

ACOSTA, L.; FAEDRICH, A. **Modelagem de sistemas utilizando a tecnologia *workflow***. UCPel, Pelotas, Rio Grande do Sul, Brasil - 2003.

AGARWAL, R.; CARMEL, E. ***Tactical approaches for alleviating distance in global software development***. IEEE *Software*, [S.l.], p. 22-29, Mar. 2001.

ARAUJO, R.; BORGES, M. **Sistemas de *workflow***. In: XX Jornada de Atualização em Informática - Congresso da SBC, Fortaleza, Ceará, Brasil - 2001.

AUDY, J.; PRIKLADNICKI, R. **MuNDDoS – Um modelo de referência para desenvolvimento distribuído de software**. In: XVIII Simpósio Brasileiro Engenharia de *Software* - 2004 - Brasília, DF, Brasil. Anais. pp. 289-304

BOOCH, G. ***Object-oriented analysis and design with applications***. 2nd ed. California: *Addison-Wesley*, 1998.

BOOCH, G.; JACOBSON, I., RUMBAUGH J. ***The unified software development process***, *Upper Saddle River*, Addison Wesley, 2001.

BORGES, L. M. S., FALBO R. A. **Uma ferramenta de apoio à instanciação de processos de *software* com gerência de conhecimento**, I Concurso de Teses e Dissertações em Qualidade de *Software*, Anais do I Simpósio Brasileiro de Qualidade de *Software*, 2002.

CARMEL, E. ***Global software teams – collaborating across borders and time-zones***. *Prentice Hall*, EUA, 1999, 269p.

CYRILLO, L. C.; HIRAMA, K. **Problemas, desafios e boas práticas para o processo de gestão em desenvolvimento distribuído de *software***.In: observatório – sessões técnicas de informática d Unochapecó, Chapecó, 2005. Anais. 2005.

DERNIAME, J. C., KABA B. A., WARBOYS B. **Software process: principles, methodology, and technology**, *Lecture Notes in Computer Science* 1500, Springer, 1999.

DROUIN, J., EMAM, K. E., MELO W., SPICE – ***The theory and practice of software process improvement and capability determination***, IEEE *Computer Society Press*,1998.

EVARISTO, R.; FENEMA, P. C. V. ***A typology of project management emergence and evolution of new forms****. International Journal of Project Management*, v. 17, n. 5, p. 275. 1999.

FITZGERALD B., O'KANE T; RUSSO N. L. ***Software development method tailoring at motorola****, In: Communications of the* ACM, April 2003, pp.65-70.

FREITAS, A.. ***APSEE-Global*****: um modelo de gerência de processos distribuídos de *software*.** UFRGS, Porto Alegre, 2005

FUGGETTA, A. ***Software process: a roadmap***, In: *Future of Software engineering Limerick Ireland*, ACM, 2000.

GEORGAKOPOULOS, D.; HORNICK, M. ***An overview of workflow management: from process modeling to workflow automation infrastructure***. *Distributed and Parallel Databases*, 3, 119-153. 1995.

GINSBERG, M. P., QUINN, L. H. ***Process tailoring and the software capability maturity model****. Technical Report, November* 1995.

GUTWIN C.; PENNER R.; SCHNEIDER K. ***Group awareness in distributed software development***. *In: Computer Supported Cooperative Work Conference Proceedings*. 2004: 72-81.

HERBSLEB, J. D.; MOITRA, D. ***Global software development***, IEEE *Software Magazin*,, IEEE *Computer Society*, EUA, Mar/Abr 2001.

HORVATH, L.; MARQUARDT, M. J. ***Global teams: how top multinationals span boundaries and cultures with high-speed teamwork***. Davies-Black. Palo Alto, EUA, 2001.

ISO/IEC 12207, ***Information technology – software life-cycle processes*****, *technical report***, 1995.

ISO/IEC TR 15504, ***Information technology – software process assessment, technical report,*** 1998.

KAROLAK, D. W. ***Global software development – managing virtual teams and environments***, IEEE *Computer Society*, EUA, 1998.

KELLNER, M. I. ***Connecting reusable software process elements and components****, In Proc. International Software Process Workshop*, 1996, pp.8-11.

KIEL, L. ***Experiences in distributed development: a case study****, In: Workshop on Global Software Development at* ICSE 2003", Oregon, EUA, 2003.

KRUCHTEN, P. ***The rational unified process: an introduction***. *Upper Saddle River, NJ: Addison-Wesley*, 2000.

MACHADO, L.F.C., **Modelo para definição de processos de *software* na estação TABA**, Dissertação de Mestrado, COPPE/UFRJ, 2000.

MUELLEITNER, G.; SALHOFER, P.; SCHSTER, G.; ZECHNER, M.; ZESAR, K. D. ***Performance and quality aspects of virtual software enterprises***. *In: EUROMICRO CONFERENCE* (EUROMICRO 1998), 24, Sweden 1998. *Proceedings*. IEEE *Computer Society*, 1998. p. 20824-20829.

NICOLAO, M. **Modelagem de *workflow* utilizando um modelo de dados temporal orientado a objetos com papéis**. Dissertação de Mestrado. UFRGS, Porto Alegre, Rio Grande do Sul, Brasil - 1998.

OLIVEIRA, K. **Modelo para construção de ambientes de desenvolvimento de *software* orientados a domínio**, Tese de D. Sc. COPPE/UFRJ, Rio de Janeiro, Brasil, 1999.

OPPENHEIMER, H. L. ***Project management issues in globally distributed development***. *In: Workshop on Global Software Development* (ICSE 2002), 24. Florida, 2002.

PEREIRA, Eliana B., **Uma proposta para adaptação de processos de desenvolvimento de *software* baseados no *rational unified process.*** Dissertação de Mestrado. PUC, Porto Alegre, Rio Grande do Sul – 2005.

PRESSMAN, R. S. ***Software enginnering: a practitioner's approach***." Makron Books, 2001.

REIJERS, Hajo A. ***Design and control of workflow process*.** *Business Process Management for the Service Industry.* (2002).

RIZZI, A. **Validação de *workflow* de autoria na implementação de um curso de ensino a distância**. UFRGS, Porto Alegre, Rio Grande do Sul, Brasil – 2001.

Rocha, A.R.C.; Travassos, G.H.; Werneck, V.M.; Werner, C.M.L. **Memphis: um ambiente para desenvolvimento de *software* baseado em reutilização**. Relatório Técnico, COPPE/UFRJ, 1996.

SOMMERVILLE, I. ***Software engineering***. 6. ed. *Addison Wesley*, 2003.

TESSMANN, L. **Modelo de *workflow* para gerência de processos em empresas prestadoras de serviços**. UFPEL, Pelotas, Rio Grande do Sul, Brasil – 2005.

TRAMONTINA, G. B. **Análise de problemas de escalonamento de processos em *workflow***. Dissertação de Mestrado. Instituto de Computação (IC) - Universidade Estadual de Campinas (UNICAMP), 2004.

USIRONO, C. **Tecnologia *workflow*: o impacto de sua utilização nos processos de negócio. Um estudo de casos múltiplos**. USP, São Paulo, São Paulo, Brasil - 2003.

VIEIRA, H. **Modelagem de uma aplicação de *workflow* na *WEB* para a integração de grupos de pesquisa**. UFPEL, Pelotas, Rio Grande do Sul, Brasil – 2005.

WfMC - *Workflow Management Coalition.* ***The Workflow Reference Model***. Disponível em: <http://www.wfmc.org>. Acesso em: 20 jul. 2007.

Dados de catalogação na fonte:
Ubirajara Buddin Cruz – CRB-10/901
Biblioteca de Ciência & Tecnologia - UFPel

T266m Teixeira, Juliano Machado

Modelagem de um ambiente de gerenciamento de desenvolvimento distribuído de software baseado em workflow / Juliano Machado Teixeira ; orientador Flávia Braga de Azambuja ; co-orientador Hugo Vares Vieira. – Pelotas, 2007. – 88f. : il. - Monografia (Conclusão de curso). Curso de Bacharelado em Ciência da Computação. Departamento de Informática. Instituto de Física e Matemática. Universidade Federal de Pelotas. Pelotas, 2007.

1.Informática. 2.Engenharia de software. 3.Workflow. 4.Desenvolvimento distribuído de software. 5.Processo de desenvolvimento de software. I.Azambuja, Flávia Braga de . II.Vieira, Hugo Vares. III.Título.

CDD: 005.1