CENTRO UNIVERSITÁRIO DA BAHIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO E TECNOLOGIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ISAC VELOZO DE C. AGUIAR

MAX-FLOW
Um estudo de interface e usabilidade de uma ferramenta de editoração de processo

Salvador
2007

ISAC VELOZO DE C. AGUIAR

MAX-FLOW
Um estudo de interface e usabilidade de uma ferramenta de editoração de processo

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Faculdade de Ciência da Computação e Tecnologia, Centro Universitário da Bahia, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Eduardo Jorge

Salvador
2007

TERMO DE APROVAÇÃO

ISAC VELOZO DE C. AGUIAR

MAX-FLOW
Editor de Processos

Monografia aprovada como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação, Centro Universitário da Bahia, pela seguinte banca examinadora:

Orientador:

Eduardo Manuel de Freitas Jorge
Mestre em Ciência da Computação
Centro Universitário da Bahia

Examinador 1:

Roberto Luiz Monteiro
Mestre em Modelagem Computacional
Centro Universitário da Bahia

Examinador 2:

Carlos de Azevedo Pinto Soares Neto
Centro Universitário da Bahia

Salvador, 28 de maio de 2007

DEDICATÓRIA

Dedico a meus pais, minha esposa e meus filhos: Ian e Júlia, pelo apoio, incentivo, conselhos e participação desde o princípio até a realização deste trabalho.

AGRADECIMENTOS

Agradeço principalmente ao Prof. Eduardo Jorge, uma pessoa simples e um exemplo sério. Por todo apoio que me deu, pelas demonstrações; por entender e com isso ter sabido orientar-me com excelência.

Aos professores da FIB que contribuíram para o acúmulo de conhecimento ao longo do curso, proporcionando um amadurecimento adequado para a conclusão do curso com este trabalho.
A meus colegas de curso, os quais compartilharam conhecimentos e emoções.

A meus amigos que me incentivaram nos momentos difíceis.

E enfim a todos que direta ou indiretamente me auxiliaram durante todo este período, que considero uma grande conquista.

RESUMO

Questões de interface e usabilidade são os elementos essenciais para a concepção de aplicações de excelência que envolve interação homem-máquina. Nesse contexto o trabalho apresenta um estudo de interface e usabilidade de uma ferramenta de editoração de processo denominada E-Flow: Uma solução de *workflow* para integração da gestão de processos com a documentação da qualidade usando xml (Costa, 2004). Os princípios que nortearam o estudo abordam fundamentos da Engenharia Cognitiva e Engenharia Semiótica, visando desta forma atender os seguintes aspectos: facilidade de aprendizado do sistema; facilidade de uso, no processo de execução do sistema; Satisfação do usuário; modificação das funções e ambientes iniciais do sistema; e maior produtividade. Como resultado deste estudo gerou-se uma especificação e implementação de uma nova interface visual para o E-Flow agora chamado de Max-Flow.

Palavras-chaves Interação Homem-Máquina, Análise e Projeto Orientado a Objeto, Sistemas de *Workflow*

LISTA DE FIGURAS

FIGURA 1. MODELO DE INTERAÇÃO DA ENG. COGNITIVA. 14
FIGURA 2. AÇÃO DO USUÁRIO DURANTE A INTERAÇÃO COM O SISTEMA. 15
FIGURA 3. PROCESSO DE COMUNICAÇÃO ENTRE DUAS PESSOAS. 18
FIGURA 4. PROCESSO DE COMUNICAÇÃO ENTRE O DESIGNER E O USUÁRIO, NA ENG. SEMIÓTICA. 19
FIGURA 5. EXEMPLO DE DIFERENTES MENSAGENS PARA UMA TAREFA DE CONSULTAS. 20
FIGURA 6. EXEMPLO DE DIFERENTES MENSAGENS PARA UMA TAREFA DE CONSULTAS. 21
FIGURA 7. EXEMPLO DE JANELA DE APLICATIVO. 24
FIGURA 8. EXEMPLO DE JANELA DE AJUDA, CONHECIDA COMO "HELP". 24
FIGURA 9. EXEMPLO DE CAIXAS DE DIÁLOGO E MENSAGENS. 25
FIGURA 10. EXEMPLO DE IMAGENS, BITMAPS E DESENHOS. 25
FIGURA 11. EXEMPLO DE MENU *DROP-DOWN*. 26
FIGURA 12. EXEMPLO DE BARRA DE FERRAMENTAS. 26
FIGURA 13. EXEMPLO DE ALGUNS DISPOSITIVOS DE ENTRADA DE DADOS. 27
FIGURA 14. PROCESSO DE DESENVOLVIMENTO DE PROTÓTIPO. 28
FIGURA 15. REPRESENTAÇÃO DE UM GRÁFICO SIMPLES, UTILIZANDO O JGRAPH. 31
FIGURA 16. EXEMPLO DE INTERAÇÃO NO JGRAPH 31
FIGURA 17. A ORIENTAÇÃO A OBJETOS ENFATIZA A REPRESENTAÇÃO DE OBJETOS. 33
FIGURA 18. ABORDAGENS ALTERNATIVAS UML PARA MOSTRAR ANINHAMENTO E PACOTES USANDO PACOTES EMBUTIDOS, NOME UML PLENAMENTE QUALIFICADOS E O SÍMBOLO CÍCULO-CRUZ. 34
FIGURA 19. NOTAÇÃO COMUM DE DIAGRAMA DE CLASSE UML. 35
FIGURA 20. EXEMPLO DE FLUXOGRAMA DE UM PROCESSO. 37
FIGURA 21. COMUNICAÇÃO ENTRE OS COMPONENTES E INTERFACES DO *WORKFLOW*. 40
FIGURA 22. ARQUITETURA GENÉRICA DO E-FLOW. 42
FIGURA 23. PROCESSO DE DESIGN DE INTERFACES. 44
FIGURA 24. COMPARATIVO ENTRE AS TELAS DE LOGIN, DA ANTIGA VERSÃO E DA VERSÃO ATUAL RESPECTIVAMENTE. 46
FIGURA 25. TELAS DO APLICATIVO: VERSÃO 2.02.00 E VERSÃO 3.00 RESPECTIVAMENTE. 48
FIGURA 26. ESTILOS DE *LOOK AND FEEL*, QUE PODEM SER ALTERADOS PELOS USUÁRIOS. 50
FIGURA 27. COMPARATIVO DAS OPÇÕES DE MENU E FUNÇÕES EXISTENTES, ENTRE A VERSÃO ANTERIOR E A ATUAL, RESPECTIVAMENTE. 50
FIGURA 28. ILUSTRAÇÃO DE UM MENU EM CASCATA. 51
FIGURA 29. BARRAS DE FERRAMENTAS DA VERSÃO ANTERIOR E ATUAL, RESPECTIVAMENTE. 52
FIGURA 30. COMPARAÇÃO ENTRE OS ELEMENTOS GRÁFICOS DISPONÍVEIS NA VERSÃO ANTERIOR E NA ATUAL RESPECTIVAMENTE. 53
FIGURA 31. DIAGRAMA DE CLASSE QUE REPRESENTA AS NOVAS CLASSES ADICIONADAS AO PROJETO. 62
FIGURA 32. DIAGRAMA DE CLASSE QUE REPRESENTA A ARQUITETURA LÓGICA DA CLASSE EFLOWLOGIN, CONTENDO OS MÉTODOS QUE FORAM ADICIONADOS E ALTERADOS. 63
FIGURA 33. DIAGRAMA DE CLASSE QUE REPRESENTA A ARQUITETURA LÓGICA DA CLASSE CONFIGURATION, CONTENDO OS MÉTODOS QUE FORAM ADICIONADOS E ALTERADOS. 64
FIGURA 34. DIAGRAMA DE CLASSE QUE REPRESENTA A ARQUITETURA LÓGICA DA CLASSE PROCESSDESIGNER, CONTENDO OS MÉTODOS QUE FORAM ADICIONADOS E ALTERADOS. 64

FIGURA 35. DIAGRAMA DE CLASSE QUE REPRESENTA A ARQUITETURA LÓGICA DO PACOTE PANEL, CONTENDO AS CLASSES SEUS RESPECTIVOS MÉTODOS QUE FORAM ADICIONADOS E ALTERADOS. 65
FIGURA 36. DIAGRAMA DE CLASSE QUE REPRESENTA A ARQUITETURA LÓGICA DO PACOTE DETAILPANEL, CONTENDO AS CLASSES SEUS RESPECTIVOS MÉTODOS QUE FORAM ADICIONADOS E ALTERADOS. 66
FIGURA 37. DIAGRAMA DE CLASSE QUE REPRESENTA A ARQUITETURA LÓGICA DO PACOTE RENDERER, CONTENDO A CLASSE SEU MÉTODO QUE FOI ALTERADO. 66
FIGURA 38. DIAGRAMA DE CLASSE QUE REPRESENTA OS ELEMENTOS GRÁFICOS. 66
FIGURA 39. DIAGRAMA DE CLASSE QUE REPRESENTA AS PROPRIEDADES DOS ELEMENTOS GRÁFICOS. 66

LISTA DE TABELAS

TABELA 1:ETAPAS DE INTERAÇÃO USUÁRIO SISTEMA 16

SUMÁRIO

INTRODUÇÃO 11

1. INTERAÇÃO HOMEM-MÁQUINA 13

1.1 Contexto IHC 13

1.2 Teoria da Interação 13

1.3 Engenharia Cognitiva 14

1.4 Engenharia Semiótica 17

1.5 Engenharia Semiótica e Engenharia Cognitiva 20

1.6 Interface Gráfica com o usuário 21

1.7 Usabilidade 22

1.8 Componentes GUI 23
1.8.1 Janelas 23
1.8.2 Ícones, Bitmaps e Desenhos 25
1.8.3 Menus 25
1.8.4 Entrada de Dados 26
1.8.5 Cores 27

1.9 Definindo um aplicativo GUI 28
1.9.1 Protótipos 28
1.9.2 Java, Componentes de Interface Gráfica 29
1.9.3 JGraph 30

2 ANÁLISE E PROJETO ORIENTADO A OBJETO 32

2.1 Análise e Projeto 32

2.2 Análise e projeto orientados a objetos 32

2.3 UML 33
2.3.1 Aplicação de UML: diagramas de pacotes 33
2.3.2 Aplicação de UML: diagrama de classe 34

2.4 REFATORAÇÃO 35

3 SISTEMAS DE WORKFLOW 37

3.1 Processo de Negócio 37

3.2 Arquitetura dos Sistemas de Workflow 39
3.2.1 A arquitetura da WfMC 39

3.3 JBPM 40

3.4 E-Flow 41

4 MAX-FLOW: UM ESTUDO DE INTERFACE, USABILIDADE E REFATORAMENTO 43

4.1 CONTEXTO DO PROBLEMA 44

4.2 Estratégias e Mudanças Efetuadas na Interface Gráfica 45
4.2.1 Tela de Login 45
4.2.2 Tela do Aplicativo 47
4.2.3 Menu 50
4.2.4 Barra de Ferramentas 51
4.2.5 Área de Modelagem/Elementos Gráficos 53

4.3 Desenvolvimento/Implementação 55
4.3.1 Especificação da funcionalidade e modelo de interação 55
4.3.2 Protótipo 55
4.3.3 Estabelecimento dos objetivos do protótipo 56
4.3.4 Desenvolvimento 56

CONSIDERAÇÕES 58

REFERÊNCIAS BIBLIOGRÁFICAS 60

ANEXO A – CLASSES QUE FORAM ALTERADAS E INSERIDAS NO PROJETO 62

INTRODUÇÃO

Nas últimas décadas, tem sido dada cada vez maior importância à interface de aplicações computacionais. A interface de uma aplicação computacional envolve todos os aspectos de um sistema com o qual mantemos contato (Moran, 1981). É através da interface que os usuários têm acesso às funções da aplicação. Fatores de satisfação subjetiva, de eficiência, de segurança, de custo de treinamento, de retorno de investimento, todos, dependem de um bom design de interface.

Na indústria de software, o design de interface tem sido conduzido através de processos iterativos de construção e avaliação de protótipos baseados em princípios e diretrizes empíricas, tal como proposto em The Windows Interface: Guidelines for Software Design (Microsoft, 1995) e Macintosh Human Interface Guidelines (Apple, 1992). Entretanto, estes princípios podem ser conflitantes em determinadas situações. Para resolvê-los, é necessário basear a prática de design de interfaces em uma fundamentação teórica (Hartson, 1998). Esta fundamentação orientará o designer ao longo da elaboração da sua solução particular para o conjunto de problemas que a aplicação pretende resolver.

Neste trabalho será realizado o estudo de usabilidade de um editor de processos da ferramenta E-flow (ferramenta de editoração de processos de workflow, que utiliza uma meta-linguagem específica para mapeamento de processos ligados a qualidade), sobre a perspectiva dos aspectos da Interação Homem Computador (IHC), que tem por objetivo principal fornecer aos pesquisadores e desenvolvedores de sistemas explicações e previsões para fenômenos de interação usuário-sistema e resultados práticos para o design da interface de usuário (ACM SIGCHI, 1992).

Serão analisadas: a facilidade de aprendizado do sistema, a facilidade de uso do sistema, a satisfação do usuário e flexibilidade de avaliar a possibilidade de o usuário acrescentar e modificar as funções e o ambiente inicial do sistema. Sobre a ótica deste estudo propõem-se mudanças na sua interface. Levando-se em consideração fará parte do escopo deste trabalho também o refatoramento da ferramenta contemplando uma nova apresentação, visando uma melhoria da relação entre os software e usuários.

Este trabalho está organizado da seguinte forma: o capítulo 1 apresenta conceitos referentes à interação homem-máquina; o capítulo 2 descreve os conceitos de análise e projeto orientado a objeto; o capítulo 3 descreve o conceitos e funcionamentos referentes aos sistemas de workflow; a capítulo 4 apresenta O o estudo de interface, usabilidade e refatoramento do Max-Flow; por fim, são apresentadas as considerações finais.

1. INTERAÇÃO HOMEM-MÁQUINA

IHC é uma área multidisciplinar, que envolve disciplinas como [Preece et al., 1994]: Ciência da Computação; Psicologia Cognitiva; Psicologia Social e Organizacional; Ergonomia ou Fatores Humanos; Lingüística; Inteligência Artificial; Filosofia, Sociologia e Antropologia; Engenharia e Design.

Este capítulo descreve conceitos referentes à interação do homem com a máquina (computador), e formas aplicáveis para a realização desta interação.

1.1 CONTEXTO IHC

No contexto de IHC devemos considerar quatro elementos básicos: o sistema, os usuários, os desenvolvedores e o ambiente de uso (domínio de aplicação) (Dix et al., 1993). Estes elementos estão envolvidos em dois processos importantes: a interação usuário-sistema e o desenvolvimento do sistema. Eles proporcionam estudos teóricos que podem ser aplicados ao desenvolvimento, para que isto ocorra é necessária à focalização nos seguintes aspectos:

● design e desenvolvimento do hardware e software;

● estudo da capacidade e limitação física e cognitiva dos usuários;

● instrumentação teórica e prática para o design e desenvolvimento de sistemas interativos;

● modelos de interfaces e do processo de interação usuário–sistema;

● análise do domínio e de aspectos sociais e organizacionais (Oliveira Netto, 2004).

1.2 TEORIA DA INTERAÇÃO

Segundo Oliveira Netto (2004), a Engenharia cognitiva centraliza-se na idéia de que a Interação Homem-Computador é completamente gerida pela interpretação e

avaliação de atividades executadas pelos usuários, estes devem traduzir os objetivos através da realização de eventos de entrada e julgar as reações do sistema a partir de eventos de saída. A Engenharia Cognitiva estabeleceu base para o que se denominou de “Engenharia Semiótica” (SOUZA, 1993), a qual utiliza a Teoria da Produção de Signos como fundamentos (ECO, 1976).

1.3 ENGENHARIA COGNITIVA

A Engenharia Cognitiva das teorias de *design* voltadas para o usuário, é uma das mais conhecidas. Norman considera que o projetista cria o seu modelo mental do sistema, chamado de *design*, com base nos modelos de usuário e tarefa. O modelo implementado passa a ser a imagem do sistema. A partir daí o usuário então interage com esta imagem do sistema e cria seu modelo mental da aplicação. O modelo mental, por sua vez, é o que permite ao usuário estabelecer suas intenções e seus objetivos relativos aos comandos e funções do sistema.

A figura 1 mostra o funcionamento de um processo de *design* na engenharia cognitiva:

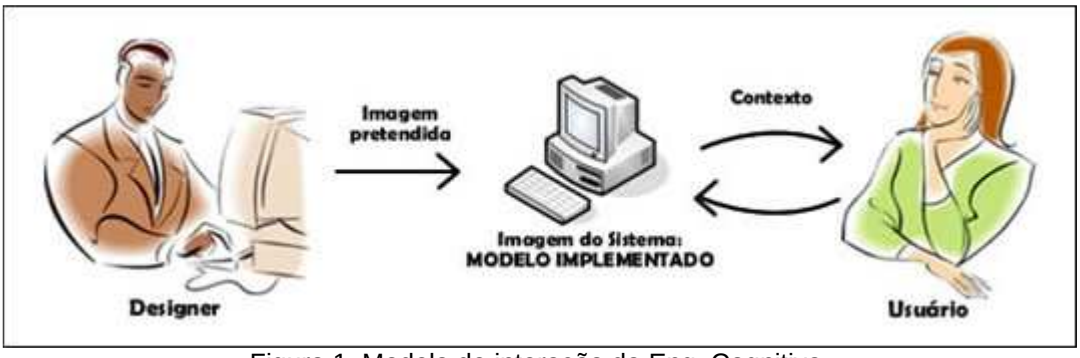

Figura 1. Modelo de interação da Eng. Cognitiva.

Fonte: Souza et al, 1999

Portanto, de acordo com a Engenharia Cognitiva o objetivo do designer é desenvolver um sistema que permita ao usuário, durante o processo de interação, criar um modelo mental consistente com o modelo que foi projetado pelo designer. Para isso ocorra, Norman (1986) acha necessário que o designer entenda o processo de interação entre o usuário e a interface do sistema, então a partir deste ponto que é proposta a teoria da ação.

A teoria da ação defende que a relação usuário-sistema seja mantida durante um ciclo de ação composto de sete etapas, ao longo dos quais, dois “golfos” deverão ser transpostos. Um deles, chamado de golfo de aplicação, envolve as etapas de definição da meta, da especificação das seqüências de ações para alcançá-las e das atividades físicas de execução. O outro chamado de golfo de avaliação deve ser superado por meio da percepção, da interação e da avaliação da meta.

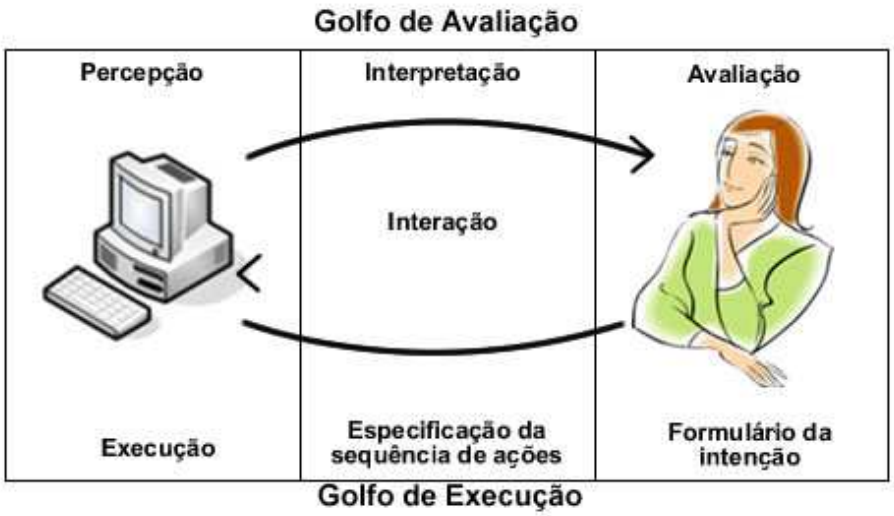

Figura 2. Ação do usuário durante a interação com o sistema.

Fonte: Souza et al, 1999

Ao utilizar o sistema, o objetivo do usuário é executar uma tarefa específica. Para isto, é necessário que ele primeiramente estabeleça as metas que deverão ser alcançadas através da interação das funções oferecidas pelo sistema, desta forma permitindo quais as ações que serão executadas para que a sua meta seja alcançada. É fato que até esta etapa do processo, houve um preparo mental por parte do usuário, para a execução da tarefa. Porém só isto não basta, é preciso que ele finalize o que foi planejado por meio de uma ação física. Estes são os momentos distintos que compreendem a travessia do golfo de execução, os quais não precisam necessariamente seguir a ordem descrita, já que as etapas de especificação e de planejamento podem ser realizadas intercaladamente, ou o comando pode ser executado mesmo que todas as ações ainda não tenham sido especificadas de forma definitiva.

Assim que o sistema executa a ação definida pelo usuário, o golfo de avaliação é iniciado, e a primeira etapa da travessia deste é a percepção do novo estado do

sistema pelo usuário, que deve interpretá-lo e avaliá-lo de acordo com a sua meta inicial. De acordo com esta avaliação, o usuário deve definir suas próximas ações. Sendo que, se o usuário não notar a mudança de estado do sistema, ele provavelmente concluirá que nada ocorreu e que sua meta inicial não foi atingida.

Abaixo segue um exemplo das etapas de interação usuário-sistema.

Em um sistema de biblioteca, um usuário que quer fazer uma consulta sobre um determinado livro ou artigo poderia passar pelas seguintes etapas de interação, de acordo com a abordagem centrada no usuário:

TABELA 1:ETAPAS DE INTERAÇÃO USUÁRIO SISTEMA

Etapa	**Descrição**
Formulação da Interação	Quero procurar a referência completa do livro Human-Computer Interaction, editado por Preece.
Especificação da seqüência de ações	Devo selecionar o comando de "**busca**" e entrar com os dados que eu tenho.
Execução	Ativo "busca" no menu; Digito o nome do livro no campo "Nome do livro"; Digito o nome do autor no campo "Nome do autor"; Seleciono OK.
Percepção	Aparece uma nova tela com os dados de livro.
Interpretação	Os dados apresentados correspondem a busca que eu fiz.
Avaliação	Encontrei as informações que eu queria. Completei a tarefa com sucesso.

FONTE: de Souza et al,1999.

Para realizar a "travessia" dos golfos, o usuário pode ser auxiliado pelo designer, que pode diminuí-los. Para que isso seja possível, ele precisa determinar quais são as ações e as estruturas de comandos das funções do sistema mais indicativas, deve selecionar os elementos de interfaces para transmitir a informação desejada e, por fim, optar por ***feedbacks*** mais representativos, pois, quanto mais próxima da tarefa e das necessidades do usuário for a linguagem de interface oferecida pelo projetista de interface, menos esforço cognitivo o usuário terá de fazer para alcançar seus objetivos.

De acordo com a figura 1, o processo de design (criação) em engenharia cognitiva se inicia com o modelo mental do sistema concebido pelo projetista de interface. No entanto a engenharia cognitiva está focando basicamente a relação entre o usuário e sistema, tendo como ênfase o produto final do processo de criação, o sistema, e o modo como o usuário o entende. Partindo desta observação, será analisada, em seguida, a engenharia semiótica, que, por estar focada basicamente no projetista de

sistemas e no processo de design do sistema, vem a ser um complemento da engenharia cognitiva.

1.4 ENGENHARIA SEMIÓTICA

A semiótica relaciona-se com tudo que possa ser “assumido” como “signo”; E signo é tudo que possa ser assumido como um substituto significativo de outra coisa. Essa “coisa” não precisa necessariamente existir, nem substituir de fato no momento em que o signo ocupa seu lugar. Nesse sentido, a semiótica é em princípio, a disciplina que tem por finalidade estudar tudo quanto possa ser utilizado para medir (ECO, 1976).

Visando tornar a interação com o usuário mais natural e menos hostil, as interfaces passaram a ser constituídas, entre outros itens, por elementos gráficos, onde imagens representando dados e tarefas disponíveis são manipuladas diretamente pelo usuário. Na realidade, tais itens não constituem os dados nem as tarefas; são apenas seus "signos". Portanto, a produção desses signos é fato crucial no bom desempenho de uma interface (Oliveira Netto, 2004).

A semiótica constitui a base teórica das abordagens semióticas. É a disciplina que, além de analisar os processos relacionados à produção e interpretação de signos, cuida de estudar os sistemas semióticos e de comunicação. Segundo Pierce (1931), signo é tudo aquilo que significa algo para alguém. Assim, tomando a língua portuguesa como exemplo, tanto a palavra "cão" quanto à fotografia de um cachorro são signos desse animal para falante do português (Oliveira Netto, 2004).

Com base na perspectiva semiótica, portanto, uma aplicação computacional pode ser considerada um ato de comunicação entre o designer, isto é, emissor de uma mensagem, e quem utiliza os sistemas concebidos por ele, no caso, o usuário-receptor (NADIM; ANDERSEN et al.; DE SOUZA; JORNA & VAN HEUSDEN, 1988,1993,1993,1996).

Para que a comunicação entre duas pessoas ocorra, é preciso que o emissor da mensagem expresse em um código que tanto ele, quanto o receptor conheçam.

Cada mensagem pode ser formada por um ou mais signos. Quando o receptor receber a mensagem, ele gera uma idéia daquilo que o emissor quis dizer e inicia o seu processo de compreensão. À idéia gerada a partir deste processo é dado o nome de interpretante, e o próprio interpretante pode estimular a criação de novos interpretantes na mente de quem recebe a mensagem, configurando uma cadeia quase infinita de associações possíveis. É a partir daí que o processo de compreensão é iniciado (Jakobson, 1970).

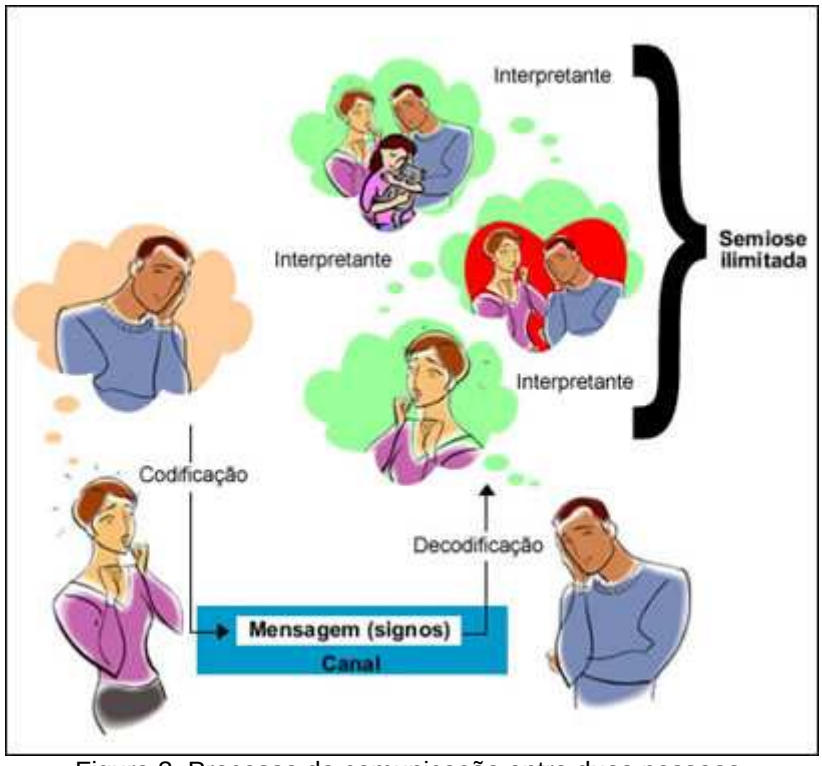

Figura 3. Processo de comunicação entre duas pessoas.
Fonte: Souza et al, 1999

Na engenharia cognitiva (de SOUZA, 1993/1996), em particular a interface de um sistema é considerada uma mensagem enviada do designer (projetista de interfaces) para o usuário, que visa responder duas perguntas muito importantes.

● Qual a interpretação do designer sobre o(s) problema(s) do usuário?

● Como o usuário pode interagir com a aplicação para resolver esse(s) problema(s)?

As respostas às perguntas acima são encontradas pelo usuário à medida que ele interage com a aplicação. Assim, trata-se de uma mensagem unilateral, uma vez que o usuário recebe uma mensagem concluída, o que impede de dar prosseguimento (de SOUZA, 1993) naquele mesmo contexto de interação. Ademais, como a própria

interface te a capacidade de trocar mensagens com o usuário, ela acaba sendo estabelecido um artefato de comunicação sobre comunicação, ou meta-comunicação.

Segundo Oliveira Netto (2004), o processo de comunicação entre o designer e usuário é apresentado na figura 4, e nela dois aspectos devem ser sublimados: a relação entre usuário e sistema, contida na mensagem transmitida do projetista de sistemas ao usuário, que ensinará ao usuário interagir com o sistema, e a necessidade de haver uma coincidência entre o almejado pelo design e a percepção efetiva que o usuário tem do modelo conceitual aplicado pelo projetista do sistema, a fim de possibilitar o sucesso da comunicação entre as duas partes.

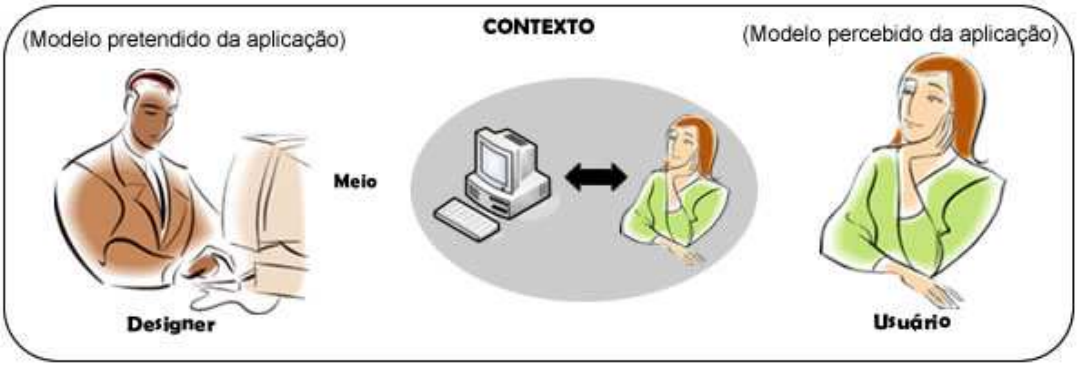

Figura 4. Processo de comunicação entre o designer e o usuário, na Eng. Semiótica.
Fonte: Norman, 1986

Na engenharia semiótica, o designer é o autor da mensagem destinada ao usuário, e a transmissão que ocorre através da interação característica do processo de metacomunicação. Com isso, vemos que a criação de interfaces abrange tanto a elaboração, como a comunicação de um modelo da aplicação. Sob o ponto de vista da engenharia semiótica, no cenário comunicativo, a o projetista de interfaces, deve permitir que os usuários tenham maiores chances de entender a aplicação.

Observando a figura 5, que representam duas telas de consulta distintas de uma aplicação, onde em uma tela a busca está restrita a somente um campo, já na tela ao lado é possível realizar a busca em vários campos.

Figura 5. Exemplo de diferentes mensagens para uma tarefa de consultas.
Fonte: Souza et al, 1999

1.5 ENGENHARIA SEMIÓTICA E ENGENHARIA COGNITIVA

Tanto na engenharia semiótica quanto na engenharia cognitiva vêem o processo de design se iniciando com o projetista de interfaces que cria o seu modelo mental da aplicação, com base neste implementa a própria aplicação. O usuário interage com esta aplicação e através dela cria o seu próprio modelo mental da aplicação. A criação da aplicação pelo projetista e a interação são assíncronas, ou seja, se dão em diferentes momentos no tempo.

Conforme relatado a engenharia cognitiva, se concentra na segunda etapa deste processo de design, ou seja, na interação usuário-sistema, deixando a etapa designer-sistema em segundo plano. Em outras palavras, a Engenharia Cognitiva dá subsídios para se definir a meta ideal do processo de design, um produto, cognitivamente adequado um determinado nível de usuários.

Já a engenharia semiótica, acopla as duas etapas, fazendo com que seu ponto de vista seja mais abstrato, no qual o designer envia ao usuário uma meta-mensagem. Desta forma, todos os resultados obtidos na Engenharia Cognitiva, continuam sendo válidos na Engenharia Semiótica. Sendo que a relação entre o usuário e sistema deixa de ser a principal preocupação da engenharia semiótica, trazendo para o seu lugar o projetista de interfaces e o processo de criação de sistemas, na figura 6 pode ser visualizado as diferentes mensagens para uma tarefa de consultas.

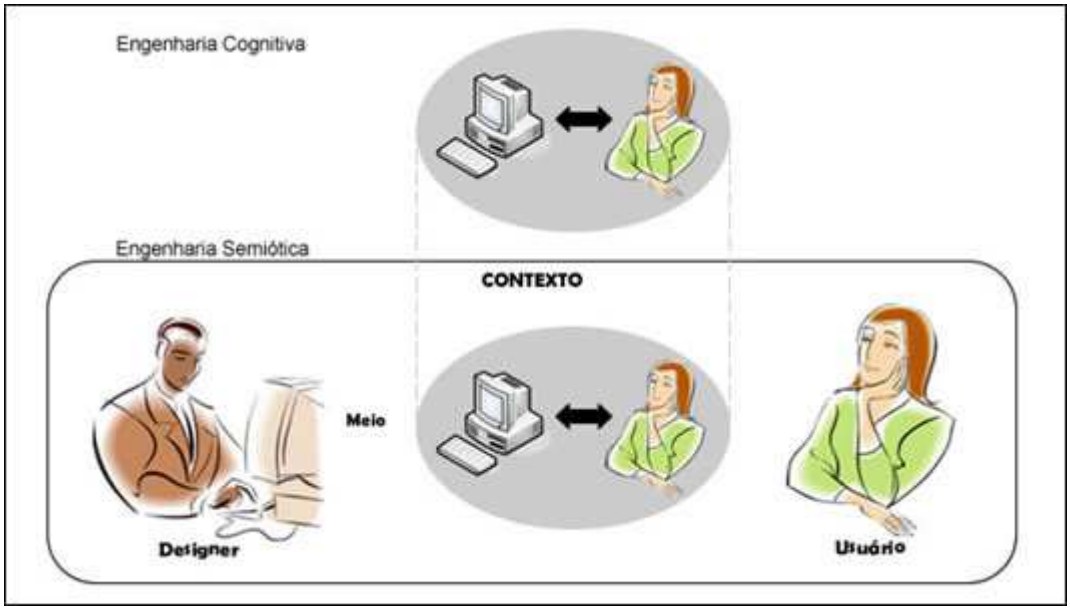

Figura 6. Relação entre a Eng. Cognitiva e a Eng. Semiótica.
Fonte: Souza et al, 1999

Visando destacar as virtudes e limitações do projetista de interfaces, a engenharia semiótica procura mostrar ao usuário que um sistema nada mais é do que uma das soluções possíveis para um problema, ou seja, não é necessariamente a única nem a definitiva. Partindo dessa premissa, sempre que tiver algum problema de interação com a aplicação, o usuário pode tentar entender o objetivo do designer e ajustar o seu próprio modelo mental da aplicação, a fim de aproximá-lo da melhor maneira possível do concebido pelo projetista de interfaces. Ao fazer isso, o usuário poderá ter um entendimento mais satisfatório das motivações do projetista durante o processo de criação, o que resultará em um uso mais eficiente da aplicação (Oliveira Netto, 2004).

A próxima seção descreve o conceito de interface gráfica, os elementos que a compõem, questões de usabilidades e um ambiente de desenvolvimento disponível para a construção desta interface.

1.6 INTERFACE GRÁFICA COM O USUÁRIO

A interface gráfica com o usuário (abreviadamente, a sigla GUI, do inglês *Graphical User Interface*) apresenta uma interface de programa. A GUI fornece a um programa uma “aparência” e um “comportamento” diferenciados. Fornecendo para diferentes aplicativos um conjunto consistente de componentes intuitivos de interface com o

usuário. As GUI's dão ao usuário um nível básico de familiaridade com um programa sem que ele jamais tenha usado o programa. Por outro lado, isto reduz o tempo exigido dos usuários para aprender a usar um programa e aumentar a sua habilidade de usar este programa de uma maneira produtiva (H. M. Deitel e P. J. Deitel, 2003).

A Interface com o Usuário é uma parte fundamental de um software; pois é a parte do sistema visível para o usuário através da qual ele se comunica para realizar suas tarefas. Ela pode se tornar uma fonte de motivação e até, dependendo de suas características, uma ferramenta para o usuário, ou então, se mal projetada, pode se transformar em um ponto decisivo na rejeição de um sistema (FOLE, 1990).

As interfaces atuais têm como objetivo fornecer uma interação humano-computador o mais "amigável" possível. Desta maneira permitindo ao usuário uma maior facilidade no momento de utilização, fornecendo seqüências simples e consistentes de interação, mostrando claramente as alternativas disponíveis a cada passo de interação, sem confundir nem deixar o usuário inseguro. Sendo que ela deve ser despercebida para que o usuário possa se fixar somente no problema que deseja resolver utilizando o sistema (Oliveira Netto, 2004).

1.7 USABILIDADE

Ao fazer referência à questão da qualidade da relação entre os sistemas computacionais e usuários, faz-se referência automaticamente à questão da usabilidade. Segundo de Souza et al (1999), a usabilidade depende dos seguintes aspectos:

Facilidade de aprendizado do sistema: compreende o tempo e o esforço necessários para que os usuários conseguir explorar o sistema e realizar suas tarefas;

Facilidade de uso: avalia o esforço físico e cognitivo do usuário durante o processo de interação com o sistema, medindo a velocidade de uso e o número de erros cometidos durante a execução de uma determinada tarefa;

Satisfação do usuário: avalia se o usuário gosta e sente prazer em trabalhar com o sistema, isto é, se o usuário se sente satisfeito com o sistema;

Flexibilidade: avalia a possibilidade de o usuário acrescentar e modificar as funções e o ambiente inicial do sistema. Sendo que este fator também mede a capacidade do usuário utilizar o sistema de maneira inteligente e criativa, realizando novas tarefas que não estavam previstas pelos desenvolvedores;

Produtividade: avalia se o uso do sistema permite que o usuário seja mais produtivo do que seria senão o utilizasse o sistema. (de SOUZA et al., 1990, p. 04).

1.8 COMPONENTES GUI

As interfaces do tipo GUI são construídas a partir de seus componente (às vezes chamados de controles ou *widgets* – a notação abreviada para *window gadgets*). O componente GUI é um objeto como o qual o usuário interage através do mouse, do teclado ou outra forma de entrada, como o reconhecimento de voz (H. M. Deitel e P. J. Deitel, 2003).

1.8.1 Janelas

Segundo Mark Minasi (1994), as Janelas são consideradas, o elemento central da interface gráfica de Sistemas Operacionais, especialmente no Windows.

Existem diversos tipos de Janelas dentre elas podemos destacar:

Janelas de aplicativo: Compreendem a estrutura visual para dados e comandos de um aplicativo. Ela é o componente visível quando se abre um aplicativo. Portanto, é nela que acontece a maioria das ações em um aplicativo (Mark Minasi, 1994), a figura 7 representa um exemplo de Janela de Aplicativo.

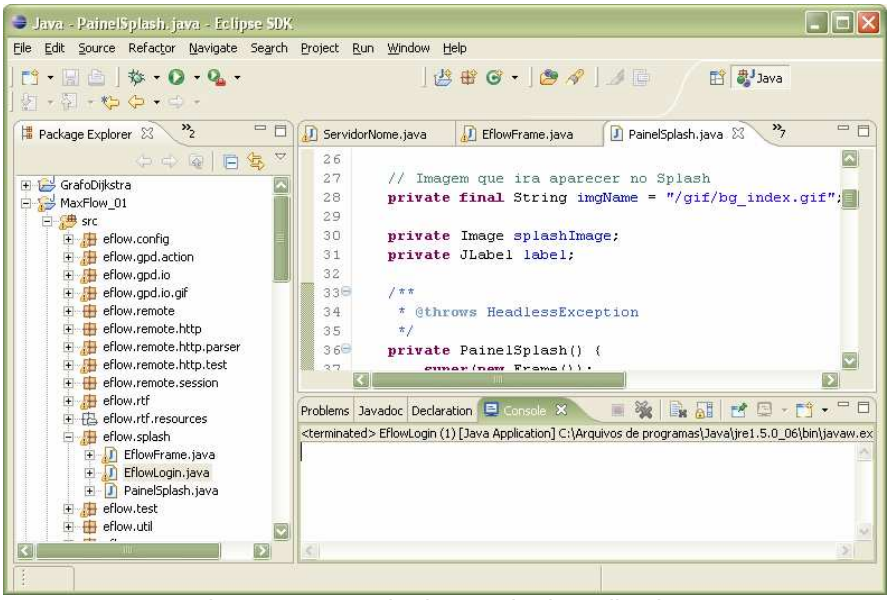

Figura 7. Exemplo de Janela de Aplicativo.

Janela de Interface de Múltiplos Documentos (MDI – Multiple Document Interface): Este tipo de janela permite que o aplicativo manipule mais de um documento de cada vez, isto é, documentos são executados simultaneamente, sendo que fica uma janela por vez ativa (Mark Minasi, 1994).

Janelas de Ajuda: São os famosos HELP, que fazem parte de um aplicativo independente, e pode ser movido para fora do aplicativo primário. O que não impede de ser utilizado sem o aplicativo está sendo executado pelo Sistema Operacional (Mark Minasi, 1994), na figura 8 pode ser visualizada uma janela de ajuda.

Figura 8. Exemplo de Janela de Ajuda, conhecida como “HELP”.

Caixas de Diálogo e de Mensagens: São caixas que exprimem comportamentos para determinadas funções executadas. Elas são utilizadas para passar informações

sobre o estado atual do sistema para os usuários, a figura 9 demonstra exemplos de caixas de diálogo.

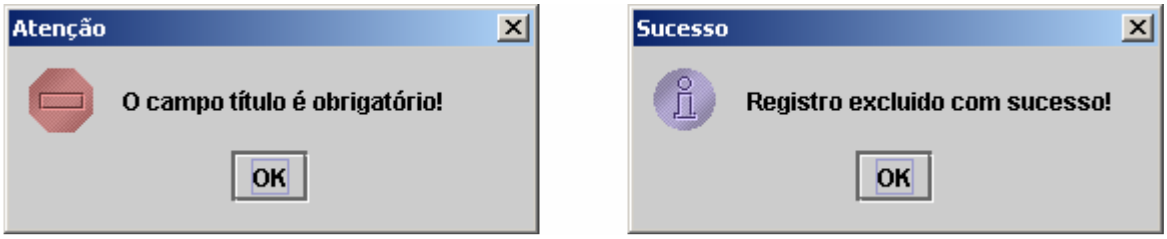

Figura 9. Exemplo de Caixas de diálogo e mensagens.

1.8.2 Ícones, Bitmaps e Desenhos

As interfaces gráficas utilizam signos que são representados através de imagens, que tem por finalidade transmitir informações de forma mais compreensível, simples e prazerosa. Sendo que muitas vezes a utilização de imagens indevidas pode ter o efeito contrário, então deve ser utilizado o bom senso e estudos para a utilização de imagens, *bitmaps* e desenhos nos aplicativos. Á figura 10 demonstra um exemplo de utilização destes recursos.

Figura 10. Exemplo de Imagens, Bitmaps e Desenhos.

Outro fator importante é saber os direitos para a utilização de imagens nos aplicativo, para que não existam violações de *copyright.* As violações compreendem a ações judiciais altamente onerosas, empresas grandes como a Apple e a Microsoft já brigaram durante anos devido a este tipo de violação.

1.8.3 Menus

Segundo Paap e Roske-Hosfstrand (1998), pode-se definir o menu como sendo um conjunto de opções apresentadas na tela de um sistema computacional, a partir das quais é possível interferir no funcionamento de uma interface. Para a utilização do menu, basta que o usuário apenas reconheça o item que deseja,. No entanto, os

itens nele contidos devem ser auto-explicativos, a fim de que seja assegurada a eficiência de funcionamento do menu.

Os menus são elementos quase que básicos em um aplicativo GUI, o que torna-os mais atrativos. Um aplicativo que possuem todas as funções presentes nos menus, incentiva a utilização por parte dos usuários. Diferentemente dos programas orientados a linhas de comandos, pois é necessário um maior tempo de utilização para que se tenha uma utilização adequada (Mark Minasi, 1994).

Dentre os tipos de menus existentes, podemos destacar dois níveis de padrão, os menus em cascata (que se ramificam em outros itens de menus) e os menus destacáveis (que podem ser movidos para outras localizações na tela). Que são utilizados de acordo com a sua necessidade. A figura 11 demonstra um exemplo de menu padrão.

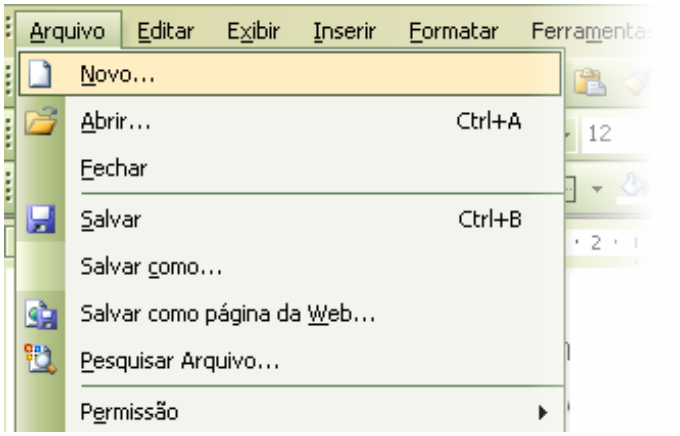

Figura 11. Exemplo de menu *drop-down*.

As barras de ferramentas são um componente GUI que pode ser utilizado como um menu, pelo fato de possuírem ícones, que ativam itens de menus. Elas são muito utilizadas, porém nem sempre são bem desenvolvidas. Estes ícones possuem uma barra de título que explicam a ação que será executada ao ele ser utilizado. A figura 12 demonstra um exemplo de barra de ferramenta.

Figura 12. Exemplo de barra de ferramentas.

1.8.4 Entrada de Dados

As interfaces gráficas utilizam principalmente o mouse como dispositivo de entrada, mais os aplicativos GUI, devem ter também o teclado como aliado para entrada de

dados que é considerado igualmente importante para inserções de informações no sistema, e ou substituir o mouse quando necessário, como também pode utilizar outros periféricos que permitem a entrada de dados nos terminais e em conjunto com os aplicativos (Mark Minasi, 1994), a figura 13 apresenta alguns dispositivos de entrada de dados em aplicativos.

Figura 13. Exemplo de alguns dispositivos de entrada de dados.

1.8.5 Cores

As cores são elementos muitos importantes quando se fala em design e ou criação. Cada uma com sua característica têm influencia sobre todos. Porém, dependendo da experiência positiva ou negativa relacionada a cada uma delas, canalizam de forma diferente as emoções. As cores servem tanto como para acalmar, como para estimular. As cores, conhecidas como cores quentes (vermelha e laranja) podem trazer estímulos a ânimos às pessoas. Já as cores conhecidas como cores frias (azul e verde) podem acalmar e trazer leveza. Sendo que tudo isto pode variar de acordo com cada um.

As cores quentes devem ser usadas em menor escala e misturadas às demais cores frias. Desta forma mantendo o equilíbrio entre a vitalidade e a tranqüilidade.

1.9 DEFININDO UM APLICATIVO GUI

Para iniciar a construção de um aplicativo GUI de “sucesso”, é aconselhável realizar o desenvolvimento de um protótipo. Iremos abordar os conceitos de prototipação, que são importantes para o desenvolvimento de um aplicativo.

1.9.1 Protótipos

Protótipos são modelos funcionais construídos a partir de especificações preliminares para simular a aparência e a funcionalidade, ainda que de forma primitiva e incompleta, de um software a ser desenvolvido. Através de sua utilização os futuros usuários do software, bem como aqueles que irão desenvolvê-lo, poderão interagir, avaliar, alterar e aprovar as características mais marcantes da interface gráfica com o usuário e da funcionalidade da aplicação proposta (POINTER Tecnologia da Informação, 2007).

Em um protótipo, a funcionalidade deve ser mostrada, ou simulada com o intuito de testar as ferramentas a serem desenvolvidas e a integração do GUI. Ela deve ser construída utilizando os conceitos abordados para o desenvolvimento de GUI, descritos anteriormente.

O protótipo é conhecido como o sistema desenvolvido de maneira mais ágil. O seu principal uso é ajudar os clientes e desenvolvedores a entender os requisitos de um sistema, pois o usuário pode experimentar o protótipo para ver como o sistema pode apoiar no trabalho (levantamento de requisitos), como também pode revelar erros e omissões nos requisitos (validação de requisitos). Ela pode ser considerada como uma atividade de redução de riscos que reduz os riscos nos requisitos.

Para o desenvolvimento de um protótipo basicamente são seguidas as seguintes etapas, ilustrados na figura 14.

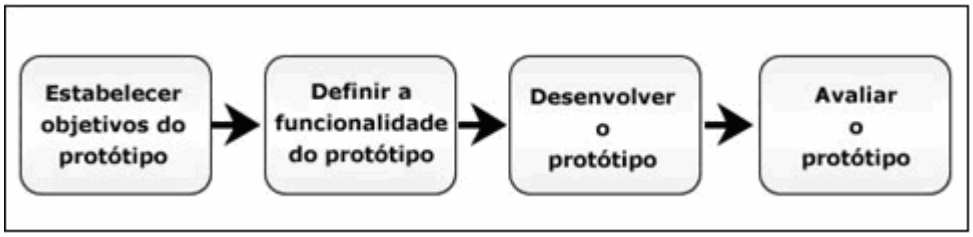

Figura 14. Processo de desenvolvimento de protótipo.

Estas etapas são fundamentais para que o protótipo atenda as suas as reais necessidades, pois estas serão aplicadas no *software.*

No processo de desenvolvimento de um protótipo existem dois tipos de abordagens, a prototipação evolucionária e a prototipação descartável. Na prototipação evolucionária é utilizada uma abordagem para o desenvolvimento do sistema onde um protótipo inicial é produzido e refinado através de vários estágios até atingir o sistema final. Na prototipação descartável, um protótipo o qual é usualmente uma implementação prática do sistema é produzida para ajudar a levantar os problemas com os requisitos e depois é “descartado”. O sistema é desenvolvido utilizando algum outro processo de desenvolvimento.

1.9.2 Java, Componentes de Interface Gráfica

A linguagem java oferece pacotes de classes construídos que facilitam o desenvolvimento de um aplicativo GUI, dentre eles podemos destacar o Swing. O Swing podem ser utilizados através da importação de suas classes que localizam-se no pacote javax.swing. Este pacote possui diversos componentes utilizados nas GUI’s, a maior parte destes componentes Swing (como são comumente denominados) são escritos, manipulados e exibidos completamente em Java (sendo conhecido com componentes Java puros) (Deitel, H. M. e Deitel, P. J., 2003).

Os componentes GUI oriundos do pacote *Abstract Wndowing Toolkit* java.awt (também conhecido como AWT) estão diretamente associados com recursos da interfaces gráfica com o usuário da plataforma local. Quando um aplicativo que utiliza os componentes deste pacote é executado em diferentes plataformas Java, os componentes GUI do programa são exibidos com uma aparência diferente em cada plataforma. Como também algumas vezes a maneira como o usuário interage com os componentes GUI de uma plataforma particular difere de uma plataforma para outra.

Já os componentes do Swing permitem que o programador especifique uma aparência e um comportamento uniformes em todas as plataformas. Além disso, o Swing permite fornecer uma aparência e um comportamento particulares para cada plataforma, ou mesmo alterar a aparência e o comportamento enquanto o programa está sendo executado.

Já os componentes do Swing permitem que o programador especifique uma aparência e um comportamento uniformes em todas as plataformas. Além disso, o Swing permite fornecer uma aparência e um comportamento particulares para cada plataforma, ou mesmo alterar a aparência e o comportamento enquanto o programa está sendo executado.

Os componentes Swing são frequentemente chamados de componentes peso-leve, pois são escritos inteiramente em Java de maneira que não são “sobrecarregados” pelos recursos GUI complexos da plataforma em que são utilizados. Os componentes AWT (muitos dos quais são equivalentes aos componentes Swing) que são vinculados à plataforma local são correspondentes chamados de componentes peso-pesado, pois eles se apóiam no sistema de janelas da plataforma local para determinar sua funcionalidade, sua aparência e seu comportamento.

1.9.3 JGraph

O JGraph é uma biblioteca de classes madura, com características interessantes no que se diz respeito à visualização gráfica. Ela foi escrita em Java. Ela pode ser utilizada em qualquer tipo de sistema operacional que tenha instalado a máquina virtual Java compatível com a versão 1.4 (JGRAPH, 2007).

Esta biblioteca de classes pode ser utilizada em aplicações do tipo cliente-side ou server-side. Ela dispõe de uma API simples, e poderosa permitindo o visível, interagir automaticamente com a disposição de executar a análise dos gráficos. Ela é recomendada para o desenvolvimento de aplicação que utilizem visualizações de gráficos. Estas englobam aplicações que utilizem diagramas de processos, *workflow* e visualização de BPM, fluxogramas, fluxo de tráfego, banco de dados e visualizações WWW, redes, diagrama de UML, circuitos eletrônicos, redes financeiras e sociais, mineração de dados, ciclos ecológicos, dentre outros (JGRAPH, 2007).

O a parte visível do JGraph é baseado na teoria matemática das redes, teoria de gráfico. Um gráfico consiste nos vértices, chamados também nós, e as linhas que realizam a conexão entre os nós. A figura 15 demonstra a utilização destes componentes gráficos.

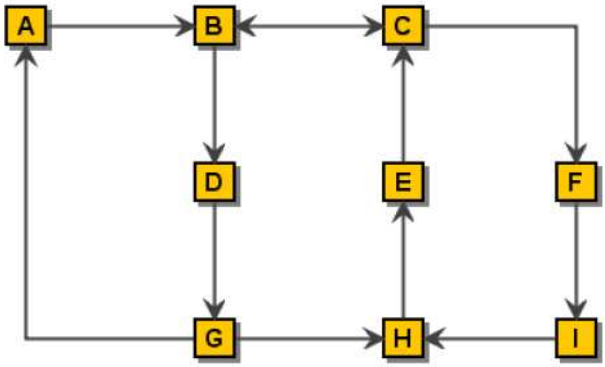

Figura 15. Representação de um gráfico simples, utilizando o JGraph.
Fonte: JGraph 2007

O JGraph permite a interação de maneira na qual pode-se realizar a alteração o modelo gráfico com uma aplicação GUI. Ele permite arrastar e empilhar, caixas e shapes, realizar ligações e desliga-las, arrastar, alterar tamanho e muito mais. Um dos principais benefícios é a flexibilidade de como a interação pode ser programada. Esta interação pode ser visualizada na figura 16.

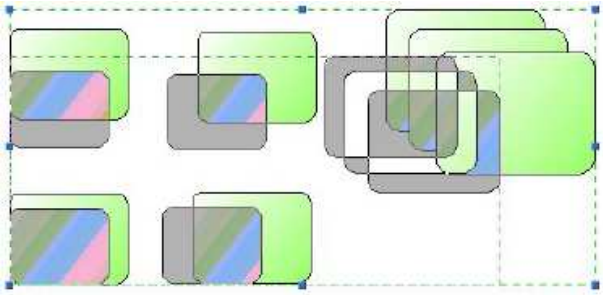

Figura 16. Exemplo de interação no JGraph
Fonte: JGraph 2007

O JGraph é compatível com todos os padrões do Swing, tais como o olhar de ligação e a sensação, transferência de dados,a acessibilidade, internacionalização e serialização. Permite a realização de características mais avançadas como voltar/avançar, imprimindo e dando suporte a XML.

A biblioteca JGraph faz parte do núcleo de software com código fonte aberto. Isto significa que o código fonte está livremente disponível. A licença é definida de acordo com os seus componentes:

- JGraph - licença do general público de biblioteca (LGPL) versão 2.1 da e versão 1.1.

- JGraph Layout Pro - JGraph Licença version 1.1. JGraph Layout Pro está livre para uso não comercial, sob os termos de licença acadêmico.

- JGraphpad Pro – JGraph Licença version 1.1 (JGraph, 2007).

2 ANÁLISE E PROJETO ORIENTADO A OBJETO

Este capítulo tem como finalidade descrever as habilidades básicas usadas na análise e no projeto orientado a objeto (A/POO). Essas habilidades são essências para a criação de um software bem-projetado, robusto e manutenível, usando tecnologias e linguagens orientadas a objeto, tais como Java ou C#.

2.1 ANÁLISE E PROJETO

A análise enfatiza uma investigação do problema e dos requisitos, em vez de uma solução, por exemplo, se desejamos um novo sistema online de comercialização, como ele será usado? Quais são as suas funções (Craig Larman, 2007).

“Análise” é um termo de significado amplo, pode ser melhor qualificado com análise de requisitos (uma investigação de requisitos) ou análise orientada a objeto (análise de investigação dos objetos do domínio) (Craig Larman, 2007).

2.2 ANÁLISE E PROJETO ORIENTADOS A OBJETOS

Durante a análise orientada a objetos, há uma ênfase em encontrar e descrever ou conceituar os objetos de acordo com o problema. Por exemplo, no caso de um sistema de informação de vôo, alguns dos conceitos incluem avião, vôo e piloto (Craig Larman, 2007).

Durante o projeto orientado a objetos, há uma ênfase na definição dos objetos de software e como eles colaboram para a satisfação dos requisitos. Por exemplo, um objeto de software avião pode ter umatributo numDaCauda e um método obterHistoricodoVoo (Craig Larman, 2007). A figura 17 demonstra a orientação objeto enfatizando a representação de objetos.

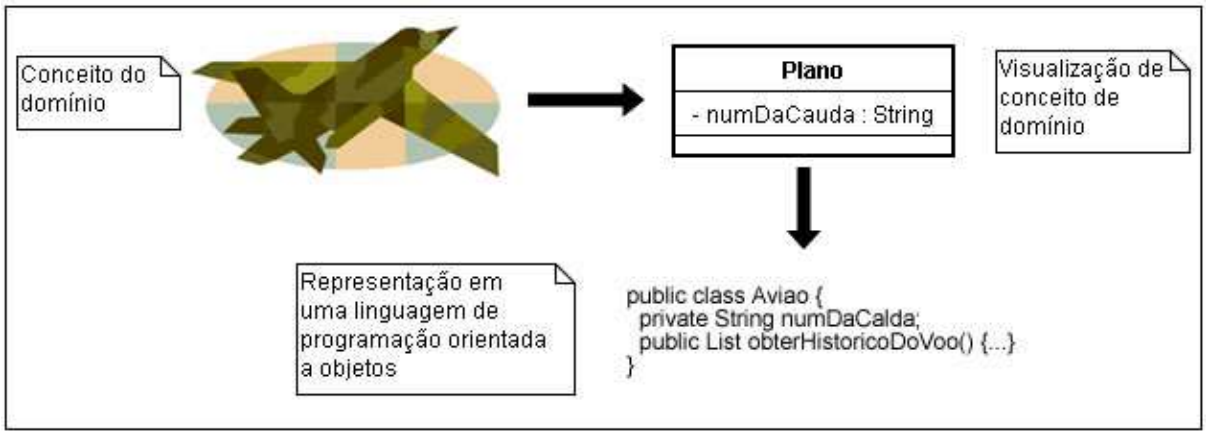

Figura 17. A orientação a objetos enfatiza a representação de objetos.
Fonte: Craig Larman, 2007

2.3 UML

> "A Linguagem de Modelagem Unificada (UML) é uma linguagem visual para especificar, construir e documentar os artefatos dos sistemas (Object Management Group, 2003 – www.omg.org)"

A UML descreve tipos de esboço de diagramas, tais como diagramas de classe e diagrama de seqüência. Ela não superpõe a eles uma perspectiva de modelagem (Craig Larman, 2007). Por exemplo, a mesma maneira de representar diagrama de classes na UML, pode ser utilizada para classes de software em Java ou para desenhar imagens de conceitos do mundo real.

2.3.1 Aplicação de UML: diagramas de pacotes

Diagramas de pacotes UML são usados freqüentemente para ilustrar a arquitetura lógica de um sistema – as camadas, subsistemas, pacotes (no significado Java), etc. Uma camada pode ser modelada como um pacote UML; por exemplo, a camada de aplicação lógica PKG como um pacote denominado PKG.

Um diagrama de pacotes UML fornece um modo de agrupar elementos. Um pacote UML pode agrupar qualquer coisa: classe, outros pacotes, casos de uso, etc. Aninhar pacotes é muito comum. Um pacote UML é um conceito mais geral do que simplesmente um pacote Java ou espaços de nome .NET, assim, um pacote UML pode representar esses – e muitos outros (Craig Larman, 2007).

A UML fornece notações alternativas para ilustrar pacotes aninhados externos e internos. Algumas vezes fica estranho desenhar uma caixa de pacote externo em volta do pacote interno. Na figura 18 são mostradas estas alternativas.

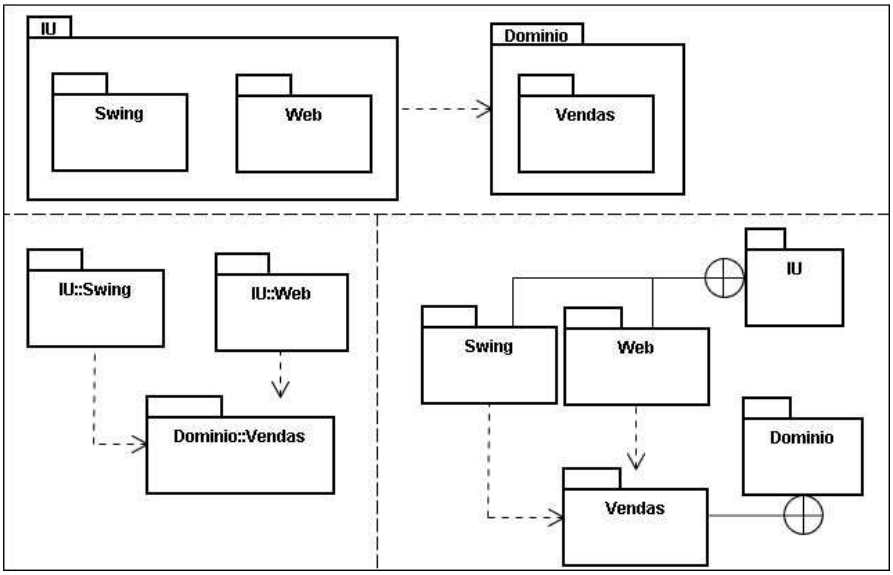

Figura 18. Abordagens alternativas UML para mostrar aninhamento e pacotes usando pacotes embutidos, nome UML plenamente qualificados e o símbolo cículo-cruz.
Fonte: Craig Larman, 2007

2.3.2 Aplicação de UML: diagrama de classe

A UML inclui diagramas de classe para ilustrar classes, interfaces e suas associações. Eles são utilizados para realização da modelagem estática de objetos.

Muito da denotação de alta freqüência em diagrama de classe pode ser resumida (e entendida) em uma figura, conforme na figura 19:

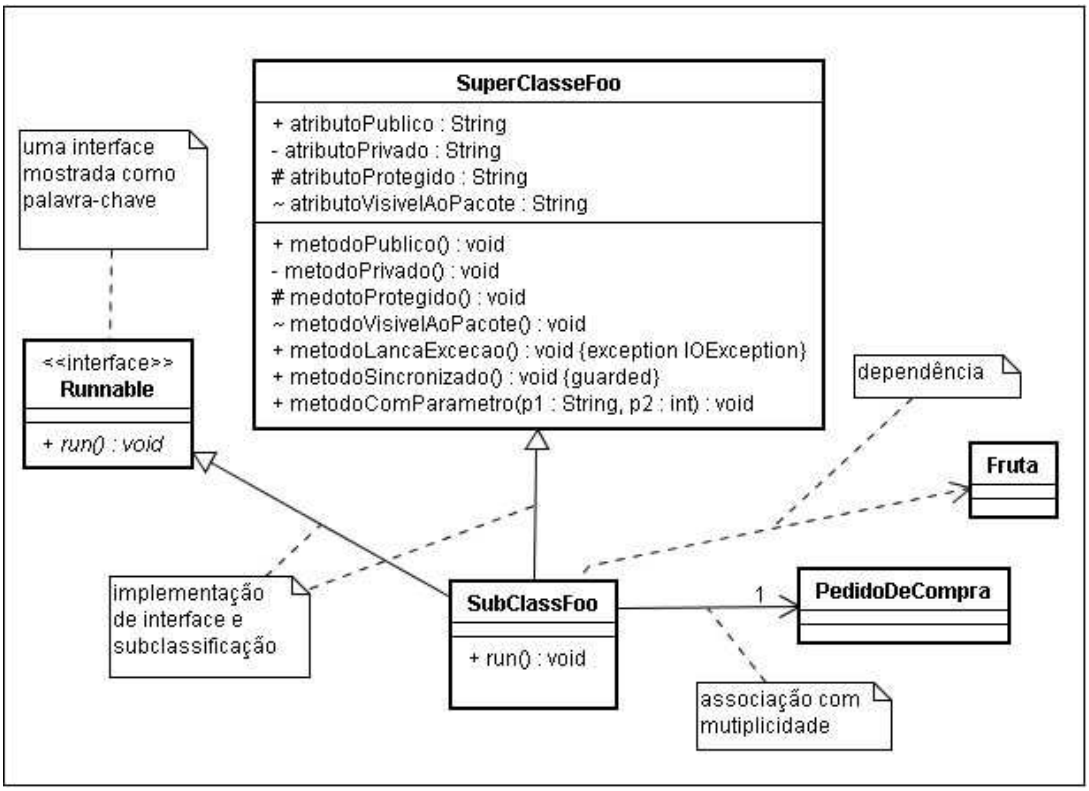

Figura 19. Notação comum de diagrama de classe UML.
Fonte: Craig Larman, 2007

A maioria dos elementos demonstrados na figura acima é opcional. Modeladores os desenham, mostram ou ocultam dependendo do contexto e das necessidades do leitor ou da ferramenta UML.

2.4 REFATORAÇÃO

Segundo Fowler (2004), refatoração é o processo de alteração de um sistema de *software* de maneira em que o comportamento externo do código não seja alterado, porém a sua estrutura interna deve ser melhorada. É uma maneira disciplinada de aperfeiçoar o código que minimiza a chance de introdução de falhas. Em essência, quando você usa refatoração, você está melhorando o projeto do código após este ter sido escrito.

A essência de refatoração é aplicar pequenas transformações preservando o comportamento (cada uma denominada uma “refatoração”), uma de cada vez. Depois de cada transformação, os testes de unidade são executados novamente para provar que a refatoração não causou uma regressão (falha). Assim, há um

relacionamento entre refatoração e DDT (Desenvolvimento Dirigido por Testes) – todos os testes de unidade apóiam o processo de refatoração (Craig Larman, 2007).

3 SISTEMAS DE WORKFLOW

Neste capítulo serão detalhados conceitos relacionados a sistemas de *workflow*. Porém, para falar de *workflow*, é necessário que antes sejam compreendidos de alguns conceitos sobre processos de negócios, conceitos sobre as atividades que compõem os processos e sobre a coordenação de tarefas de trabalhos. Serão abordados os conceitos básicos, juntamente com os sistemas de *workflow*.

O padrão adotado para a apresentação da teoria segue a que foi desenvolvida pela WfMC (*Workflow Management Coalition*). A WfMC é uma organização internacional sem fins lucrativos, composta por empresas fabricantes de *workflow*, usuários, universidades e pesquisadores. Além deste, existem outros modelos de padronizações no mercado, sendo que ainda não existe uma padronização onipresente nas ferramentas de *workflow*.

3.1 PROCESSO DE NEGÓCIO

“Processo de negócio é um conjunto de atividades que tem por objetivo transformar insumos (entradas), adicionando-lhes valor por meio de procedimentos, em bens ou serviços (saídas) que serão entregues e devem atender aos clientes” (CRUZ, 2003, p.63). Esta definição de Cruz demonstra a essência de um processo de negócio. Os processos de negócios podem ser representados pelos fluxogramas. A figura 20 demonstra um exemplo simples de um fluxograma representando um processo:

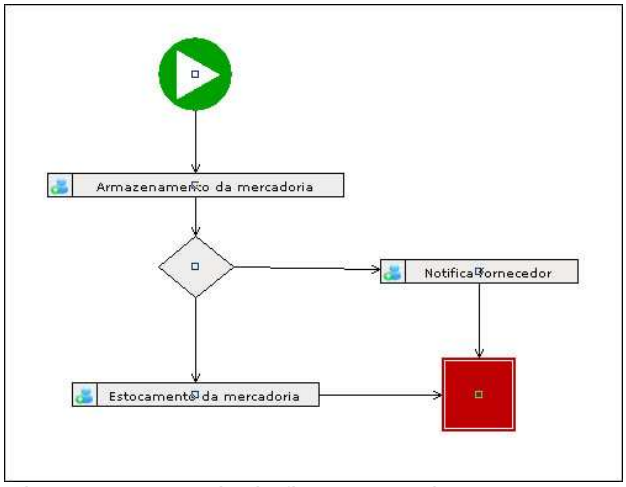

Figura 20. Exemplo de fluxograma de um processo.
Fonte: Costa, 2004

Este fluxograma, ilustrado na figura 20, representa uma definição simplificada de um processo de recebimento de mercadorias.

- **Atividade**: Conforme Nicolau, atividade é “um conjunto de eventos que ocorrem sobre a responsabilidade de um ator” (NICOLAU,1998). Ele ainda afirma que o fato da responsabilidade estar designada a um ator não impede que a atividade seja realizada por diversos participantes. No fluxograma de atividade demonstrado na imagem 20 pode-se visualizar a representação gráfica de cinco atividades: A atividade de início do processo; Armazenamento de mercadoria; Notifica fornecedor; Estoque da mercadoria; A atividade fim do processo.

- **Ator:** O ator ou participante do workflow tem a responsabilidade da execução parcial ou total das atividades definidas em um processo (Costa, 2004). Este ator pode ser uma pessoal ou um sistema. Tomando como o exemplo o processo da figura 20, o ator seria o responsável pela atividade de armazenamento da mercadoria.

- **Fluxo**: Caminho lógico ou físico que se percorre no processo de negócio. Este fluxo pode ser visualizado na figura 20 pelas linhas que interligam as atividades.

- **Padrões de Sincronismo:** os padrões de sincronismos são as regras que definem o fluxo dos processos (Costa, 2004). Estes sincronizam a execução das atividades definindo a rota seguida pelo processo. Na figura 20, existe um padrão de sincronismo representado pelo losango. Neste caso, ele representa uma decisão onde, se a mercadoria estiver sido entregada corretamente, esta irá ser encaminhada para estoque, caso contrário, o fornecedor da mercadoria recebe uma notificação da não conformidade na entrega.

- **Evento**: o evento dispara uma ação a ser executada através do workflow. Este evento refere-se a um evento do mundo físico, por exemplo, o recebimento de uma mercadoria é um evento que decorre no registro deste acontecimento no workflow iniciando assim um processo de aprovação de mercadoria.

- ***Workflow:*** é a automatização de processos que permitem a implementação de qualquer aplicação, usando a mesma ferramenta para diferentes soluções (WfCM, 2006). Eles são sistemas especializados no apoio aos processos de negócio. É

responsável pela automação de todo ou parte dos processos que lhes foram delegados (Costa, 2004).

3.2 ARQUITETURA DOS SISTEMAS DE WORKFLOW

Esta seção apresenta considerações sobre a arquitetura dos sistemas de workflow. O modelo adotado para a apresentação da arquitetura é o modelo especificado pela WfMC.

3.2.1 A arquitetura da WfMC

A WfMC é uma organização internacional sem fins lucrativos, composta por fabricantes, consumidores, analistas e pesquisadores. A sua missão é promover e desenvolver o uso do *workflow* estabelecendo padrões para terminologia de software e conectividade com outros produtos de *workflow* (WfMC, 2006).

Durante a análise de diversas ferramentas realizadas pela WfMC, percebeu-se que estas apresentavam características em comum, como por exemplo, a técnica de modelagem de processos, porém as técnicas eram individuais para cada ferramenta, o que prejudicava a comunicação entre as ferramentas. Então a WfMC, definiu padrões para que os sistemas ficarem mais homogêneos. Estes padrões aproveitam as características em comum identificadas e tornavam uniforme nos pontos em que havia divergência. A necessidade da interoperabilidade entre os sistemas foi a principal causa para a criação destes padrões para *workflow*.

O modelo de referência do workflow fornece a estrutura para o programa de padrões da WfMC, identifica as características comuns do sistema de *workflow* e define cinco interfaces funcionais, através das quais, o sistema de gerência do *workflow* interage com o ambiente (usuários, aplicações, ferramentas de computador e outros serviços de softwares). A figura 21 exibe os componentes e interfaces (RUIZ, 1999).

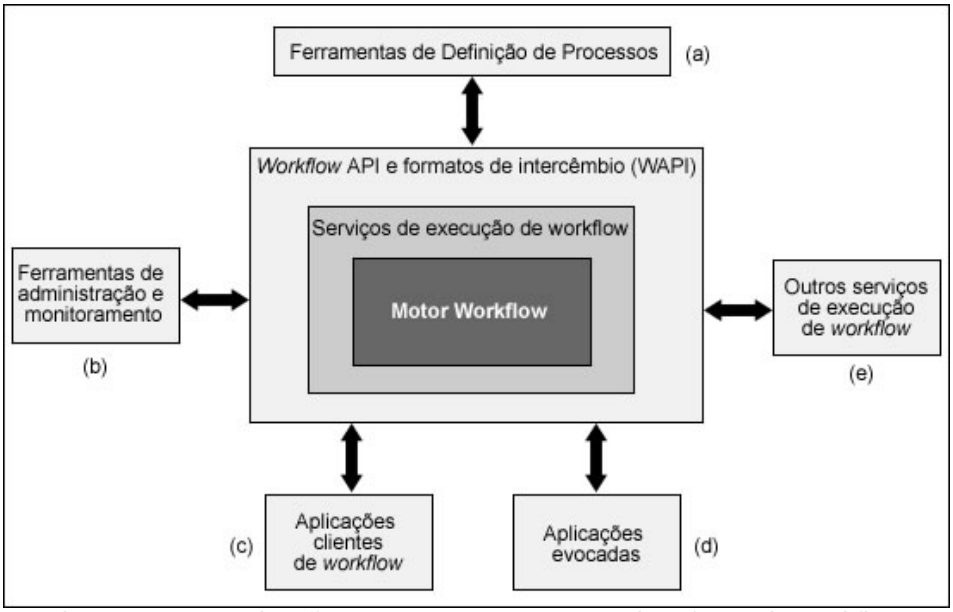

Figura 21. Comunicação entre os componentes e interfaces do *workflow*.
Fonte: WfMC, 2006

O WAPI é considerado um serviço de interface unificado, com objetivo de sustentar as funções de gerência do *workflow*, através de 05 (cinco) áreas funcionais, o que não devem ser consideradas como interfaces individuais. Uma vez que existem funções comuns a dois ou mais serviços de interface, fazendo com que estejam inter-relacionadas (WfMC, 2006).

3.3 JBPM

O jBPM (Java Business Process Management) é um poderoso BPM da JBoss (comunidade de projetos de código aberto), que viabiliza a criação dos processos de negócios e possui arquitetura modularizada, aliado a facilidade no desenvolvimento de aplicativos workflow. Utiliza um motor do processo flexível, escalável e fornece uma estrutura para criar, coordenar e monitorar os processos de negócios (JBOSS, 2006).

De acordo com JBOSS (2006), nos últimos anos, um número muito grande de esforços de especificação foi realizado na área de linguagem de definição de processos. Por exemplo: WfMC, XPDL, BPML, ebXML, BPE4WS, XLANG e WSCI. Após fazer pesquisas nas especificações existentes, foi descartada a reutilização de uma das especificações pré-existentes, por concluis que seriam apresentados problemas. As justificativas apresentadas são:

- inúmeras especificações e não um padrão;
- extremamente complexo;
- nenhum mecanismo fácil para código de integração de junção para lógica do processo de negócios.

A Jboss jBPM decidiu criar um padrão próprio de *workflow*, se baseando no WfMC, chamado de jPdl (linguagem de definição de processos). A sua implementação é feita na linguagem java em conjunto com o framework J2EE (*Java 2 Enterprise Edition*). O J2EE é um framework para o desenvolvimento de aplicações orientadas a Web. Este componente encapsula todas as funcionalidades implementadas pela ferramenta, sendo que serviços como páginas *Web* dinâmicas e a persistência dos dados são destaques (JBOSS, 2006).

3.4 E-FLOW

O E-flow é um editor de processos de *workflow*, isto é, uma ferramenta que automatiza a gerência do fluxo de processos organizacionais, utilizando uma meta-linguagem específica para mapeamento de processos ligados a qualidade, no mesmo ambiente (COSTA, 2004).

O E-flow utiliza o *workfow* jBpm (Java Bussiness Process Management) como base para a sua implementação, sendo que esta é baseada na especificação da WfMC (Workflow Management Coalition). Desta forma realizando uma extensão da metalinguagem definida pelo jBpm.

Para o desenvolvimento do E-Flow, foram realizadas pesquisas dos conceitos, técnicas sistemas e padrões de workflow. Abordando principalmente a arquitetura proposta pela WfMC, mais especificamente na *interface* aplicações cliente de *workflow*. Conforme visto na seção 3.2.1, esta *interface* é fundamental para interagir como mecanismo *workflow* (COSTA,2004).

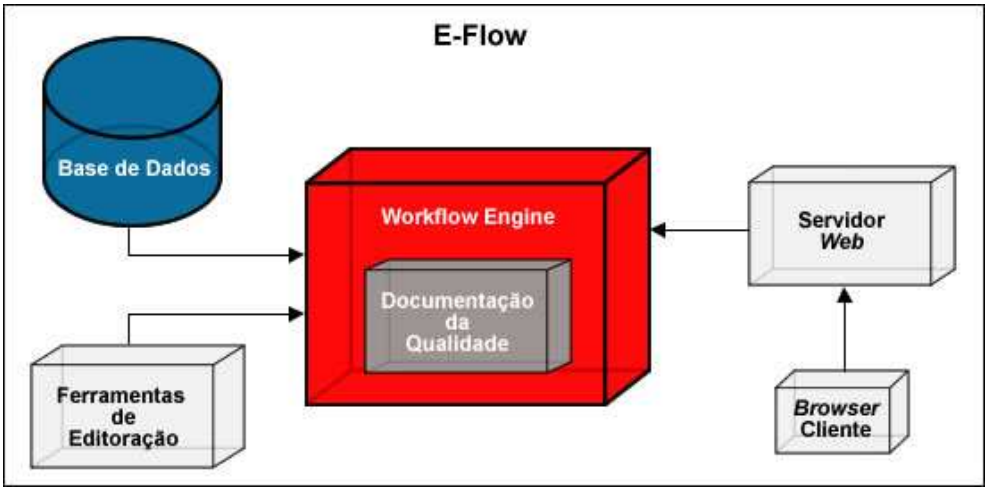

Figura 22. Arquitetura genérica do e-flow.
Fonte: Costa, 2004

A arquitetura genérica do E-Flow, representada na figura 22, mostra a possibilidade de distribuição dos componentes do E-Flow. O componente “base de dados” é refere-se a uma base relacional dos dados do E-Flow. O servidor *Web* tem a função de disponibilizar a interface *Web*. As ferramentas de editoração possibilitam o desenho e definição dos processos, além de gerar modelos de documentos. O componente *workflow engine* tem a função de promover os serviços de execução e monitoramento dos processos. Este componente descreve uma *interface* de programação (API) como forma de acesso aos seus serviços. O *workflow engine* utilizado pelo E-Flow é o JBPM, citado na seção 3.3. Integrado ao componente *workflow engine*, está o módulo da documentação da qualidade, responsável em gerar os documentos da qualidade do *e-flow* (COSTA, 2004).

4 MAX-FLOW: Um estudo de interface, usabilidade e refatoramento

Este capítulo apresenta a um estudo de interface, usabilidade e refatoramento do Max-Flow, uma ferramenta de editoração de processos associado com a documentação da qualidade, usando a XML.

O objetivo deste trabalho é propor uma especificação, implementação e refatoração de um editor de processos denominado E-Flow, a ferramenta foi resultado do projeto final de conclusão de curso E-flow: Uma solução de *worflow* para integração da gestão de processos com a documentação da qualidade usando xml (Costa, 2004), este gerado como parte da consulta desenvolvida por Costa. Esta ferramenta tem como principal requisito seguir a mesma padronização do diagrama de atividade da UML(*Unified Modeling Language*) e contemplar atributos estendidos para a definição de processos normativos. Os atributos estendidos fazem parte da definição de uma meta-linguagem que apóia estruturação dos documentos da qualidade. Os processos desenhados no E-Flow poderão ser exportados no padrão BPL (*Business Process Library*), isto permitirá que os processos possam ser utilizados em uma ferramenta de BPM (*Business Process Modeling*, significando em português "Modelagem de processos de negócio"). Detalhes do projeto E-Flow estão descrito na sessão 5.4 deste documento.

É importante ressaltar que um processo é composto por elementos que representam o início, as atividades, os sub-processos, as decisões e o término. Estes modelados utilizam a padronização da diagramação da UML, referenciada na sessão 3.3.

A refatoração do editor de processos E-Flow, foi denominado Max-Flow, para a especificação foi realizado um estudo sobre a antiga versão do E-flow, e elaborado um roteiro, destacando as possíveis melhorias. Para a definição da nova interface foram analisados critérios de usabilidade, avaliando fatores relativos à facilidade de uso, maior produtividade na diagramação dos processos, flexibilidade e satisfação do usuário. Outro critério de suma importância foi o critério da comunicabilidade, que procurou escolhas de signos de interface padronizados. Esses critérios foram fundamentados com em diretivas dos autores Mark Minasi e Oliveira Netto.

Visando resumir o que foi analisado até o momento, o processo de desenvolvimento de interfaces foi iniciado a partir de aspectos voltados aos usuários, levando-se em consideração as tarefas realizadas pela ferramenta. As etapas do processo de desenvolvimento foram as seguintes: especificação da funcionalidade e do modelo de interação, fabricação do protótipo da interface, avaliação do protótipo e desenvolvimento do produto final. O protótipo foi realizado, com base no ciclo de especificação definido pelo autor de Souza, na qual a análise é realizada em a partir da especificação, prototipação e avaliação, este ciclo pode ser visualizado na figura 23.

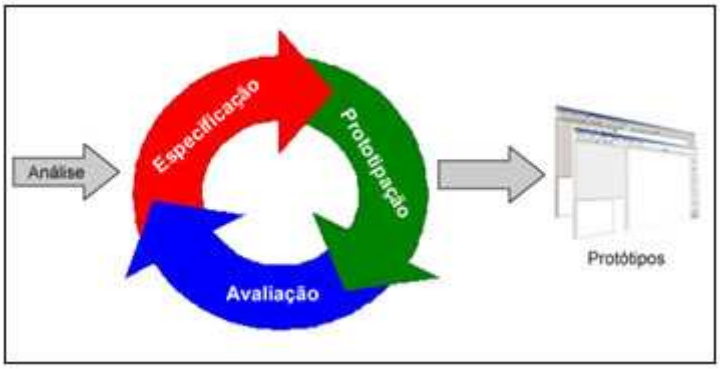

Figura 23. Processo de design de interfaces.
Fonte: de Souza et AL. ,1999

4.1 CONTEXTO DO PROBLEMA

Visando o conceito de interação e usabilidade do sistema, foi realizado um estudo da ferramenta E-flow versão 2.0.2, sendo a principal meta o levantamento de questões de usabilidade levando em consideração a interação homem-computador.

Neste estudo foi identificado que a ferramenta apresentava algumas deficiências na interação homem-computador. Levando-se em consideração que a interface é um dos principais motivos de aprovação e utilização dos sistemas em geral, foi proposta a remodelagem do design de interface da ferramenta. Sendo conduzido através de processos interativos de construção e avaliação de protótipos baseados em princípios e diretrizes empíricas. Fundamentação esta que serviu para orientação ao longo da elaboração da sua solução particular para o conjunto de problemas.

No decorrer desta sessão será descrita a estratégia utilizada para o criação da nova interface, detalhando as mudanças efetuadas, como também será demonstrado a o processo de desenvolvimento utilizado.

4.2 ESTRATÉGIAS E MUDANÇAS EFETUADAS NA INTERFACE GRÁFICA

Para facilitar o entendimento a respeito do projeto, primeiro abordam-se questões de refatoramento e usabilidade. Sendo realizada a transição da ferramenta E-Flow em relação a Max-Flow, elencando as diferenças de interface e usabilidade. Após esta apresentação as seções posteriores tratam de como foi realizado o desenvolvimento do Max-Flow.

Como conteúdo de explicação das duas interfaces, será realizado com a representação das telas em paralelo, ou seja, lado a lado, a partir das suas telas antigas representadas pelo E-Flow e as novas telas representadas pelo Max-Flow, desta forma facilitando a compreensão.

A seguir apresentam-se os seguintes elementos de interface: tela de login, tela do aplicativo, menu, barra de ferramentas, área de modelagem e seus elementos gráficos.

4.2.1 Tela de Login

A tela de login é o formulário de acesso ao sistema. Visando melhorar a comunicação desta tela, foram adicionadas algumas características dos diversos princípios existentes para o desenvolvimento de uma GUI. Dentre eles pode-se destacar a atração visual, e a facilidade de leitura e compreensão das informações passadas pela tela do aplicativo.

A alteração do layout desta tela teve o objetivo principal torná-la mais atrativa, com melhor representação da tela, utilizando recursos como imagens e formatçaão de componentes visuais como: campos de textos, botões e menu.

Para a entrada de dados, foi utilizada a tela no estilo de formulários, contendo caixas de textos, campos os quais o usuário deverá preencher e os botões, que são controles gráficos que indicam as possíveis ações disparadas pelos usuários.

A figura 24 permite que seja perceptível de maneira visual as alterações na tela de login.

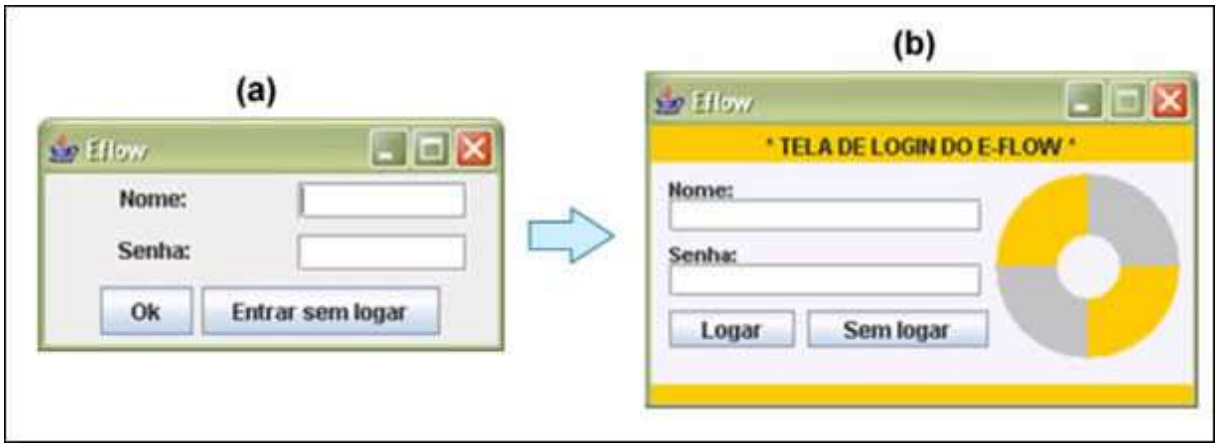

Figura 24. Comparativo entre as telas de login, da antiga versão e da versão atual respectivamente.

A ilustração (a) possui diferenças quando comparadas com a ilustração (b) da figura 24, dentre elas pode-se citar:

- O título, presente na ilustração (b), de forma a permitir que o usuário se informe facilmente em que local do sistema ele se encontra;
- A imagem da logomarca da ferramenta E-Flow, presente na ilustração (b), para atrair o usuário e familiarizá-lo com a ferramenta;
- Formatação dos campos e textos do formulário, na ilustração (b), de forma a melhorar a aparência visual dos campos;
- Definição adequada para os textos presentes no formulário, na ilustração (b), dando maior clareza referente às possíveis ações que poderão ser disparadas por parte dos usuários.

Nota-se que os elementos adicionados na ilustração (b), tornaram a tela mais atrativa e compreensiva, elementos estes que não estão presentes na ilustração (a).

4.2.2 Tela do Aplicativo

A tela do Aplicativo é também conhecida como tela de aplicação, compreende a estrutura visual para dados e comandos de um aplicativo. É nela que ocorrem as principais ações do sistema.

Na tela do aplicativo é realizada a editoração dos processos, com base no conceito de diagramação da UML.

Na figura 25 pode ser visualizada a disponibilização dos seus componentes.

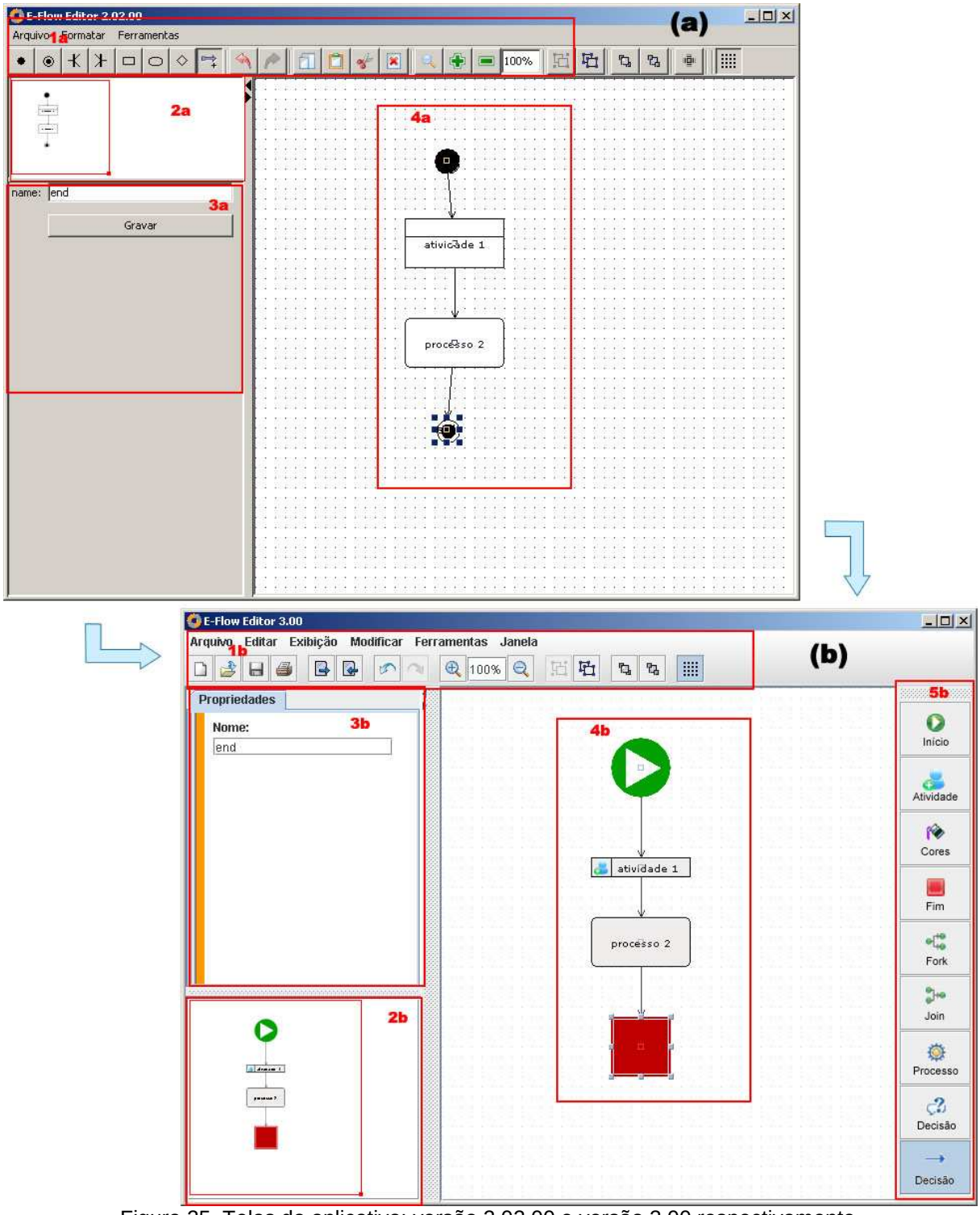

Figura 25. Telas do aplicativo: versão 2.02.00 e versão 3.00 respectivamente.

Como já foi mencionado anteriormente, a ilustração (a) refere-se à versão antiga da ferramenta, e a ilustração (b) refere-se à ferramenta atual. Pode-se analisar algumas diferenças abaixo:

- O menu e a barra de ferramentas região 1a da ilustração (a), na ilustração (b) definidas como regiões 1b e 5b, sofreram modificações, como inclusão de novas

opções no menu e separação dos elementos principais utilizados na modelagem dos processos (1b e 5b), desta forma o menu se tornou mais completo e as barras de ferramentas foi dividida em duas, dando um maior destaque para os elementos que possuem as principais funcionalidades no sistema, a criação do diagrama de processos;

- A região 2a da ilustração (a), foi modificada para baixo das propriedades referentes aos elementos gráficos do processo, definidos na região 2b da ilustração (b), que permitem a visualização do processo que está sendo modelado na área de modelagem;

- A região 3a da ilustração (a), foi colocada acima da opção de visualização da modelagem definida na região 3b da ilustração (b), permitindo assim uma melhor visualização, como também foram reestruturados os campos do formulário, e retirado a o botão gravar que não precisou ser utilizado na nova versão, desta forma deixando o usuário mais livre na realização das suas alterações e realizando a gravação automática.

- A região 4a da ilustração (a), demonstra os elementos, como a região 4b da ilustração (b), sendo que a região 4b possui elementos gráficos mais compatíveis com a descrição dos elementos gráficos, facilitando o entendimento, frisando que foi adicionado um ícone ao elemento de atividade, de forma a tornar mais clara a funcionalidade do elemento.

Das alterações que não eram disponibilizadas na versão anterior, pode-se destacar a alteração de *look and feel*, alteração esta que permite o usuário a possibilidade de modificar o ambiente inicial do sistema, estado de apresentação visual do sistema, que são demonstrados na figura 26.

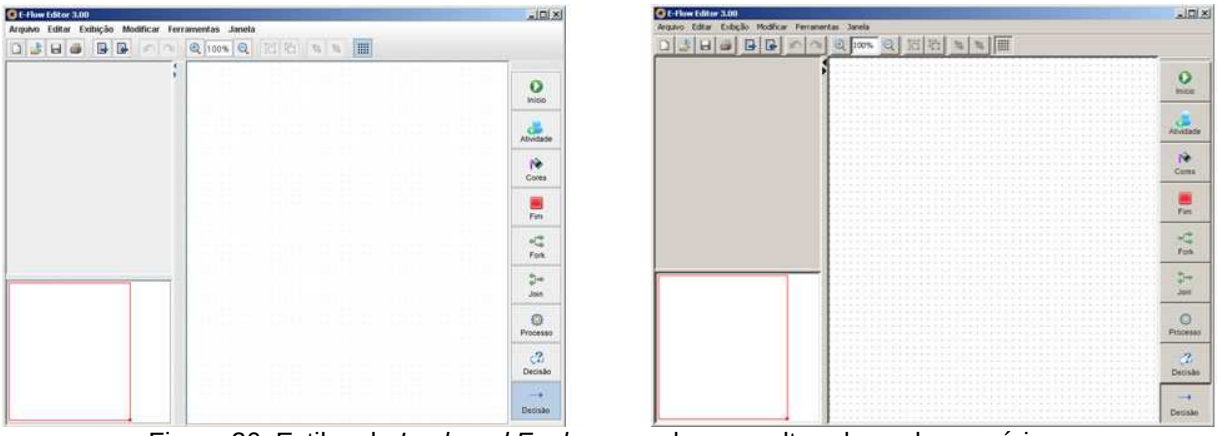

Figura 26. Estilos de *Look and Feel*, que podem ser alterados pelos usuários.

4.2.3 Menu

No menu já existente na versão anterior, poucas opções do programa eram apresentadas. Então nesta versão foi reestruturado um novo menu, contemplando todas as funções existentes do aplicativo. Segue na figura 27 o comparativo visual entre os menus.

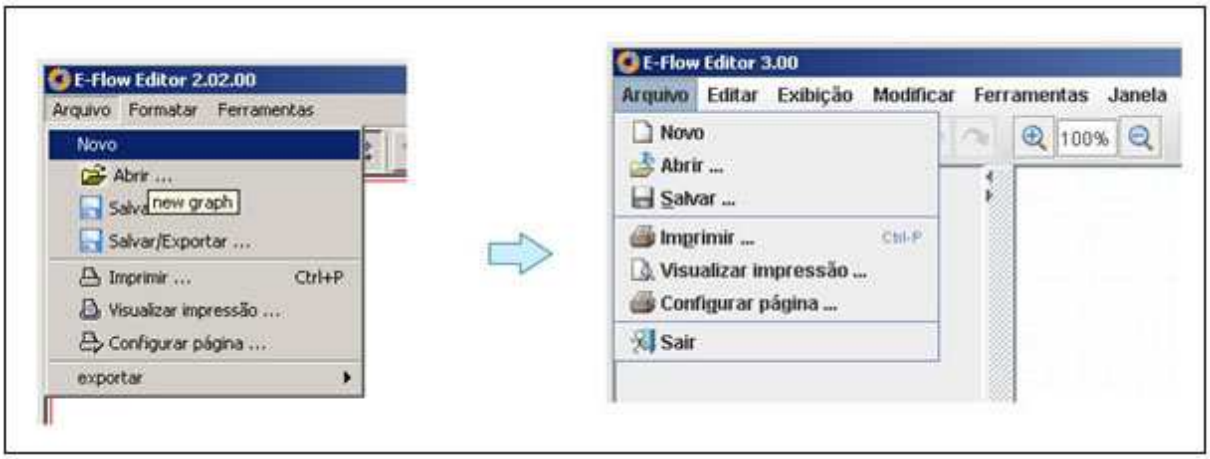

Figura 27. Comparativo das opções de menu e funções existentes, entre a versão anterior e a atual, respectivamente.

Neste novo menu além das funções adicionadas, foram incluídos signos, que permitem a identificação e associação das imagens com os elementos presentes nos ícones disponibilizados na barra de ferramentas. Como também foi realizada a organização estrutural de acordo com as opções presentes no menu.

Este menu utiliza a sua forma de apresentação conhecida como menu *drop-down*, ou seja, um menu que aparece quando seu título é selecionado, e desaparece quando se seleciona uma das opções por ele oferecidas, representado na figura 6.5. Este é o mais antigo tipo de menu do mundo das interfaces gráficas (Mark Minasi,

1994), e é utilizado, juntamente com o menu Cascata, que ramificam opções do menu padrão. Eles aparecem à direita da opção de menu drop-down, como pode ser visto na figura 28.

Figura 28. Ilustração de um menu em cascata.

Como resultado da reestruturação do menu, a nova versão possui um menu mais completo e organizado. Contendo novas opções de menu com as suas respectivas ações de acordo com as opções disponíveis.

4.2.4 Barra de Ferramentas

A barra de ferramentas, mas conhecida como tira de ícones que ativam itens de menus, utilizam ícones que representam suas ações. Na versão anterior existia apenas uma barra de ferramentas, esta barra de ferramentas era composta pelos ícones mais importantes nas etapas de construção de um processo, alguns ícones com comandos básicos (estes são avançar, voltar, recortar, colar, copiar, excluir), alguns poucos utilizados pelos usuários, outros ícones importantes que eram mais utilizados pelos usuários não faziam parte como, por exemplo: abrir, salvar e imprimir.

É importante ressaltar que os ícones mais importantes e mais utilizados na etapa de construção dos processos, não possuíam um bom destaque visual e perceptível quando comparado com a nova ferramenta, estas modificações introduziram conceitos que viabilizam principalmente facilitar a usabilidade, avaliado pelo esforço físico e cognitivo do usuário durante o processo de interação, e produtividade, permitindo com que o usuário seja mais produtivo do que sem a utilização do sistema.

Na nova versão a barra de ferramentas foi dividida em duas, separando os elementos principais da para uma barra de ferramenta, permitindo a realização da

mobilidade desta barra de ferramenta por qualquer região da tela do aplicativo, podendo ser fixada em qualquer extremidade da tela. Os ícones presentes nesta barra foram modificados, aumentando os signos, definidos como as imagens, e descritivo que se referem aos elementos que compõem o processo. As novas barras de ferramentas podem ser visualizadas na figura 29, como a barra de ferramenta da versão anterior.

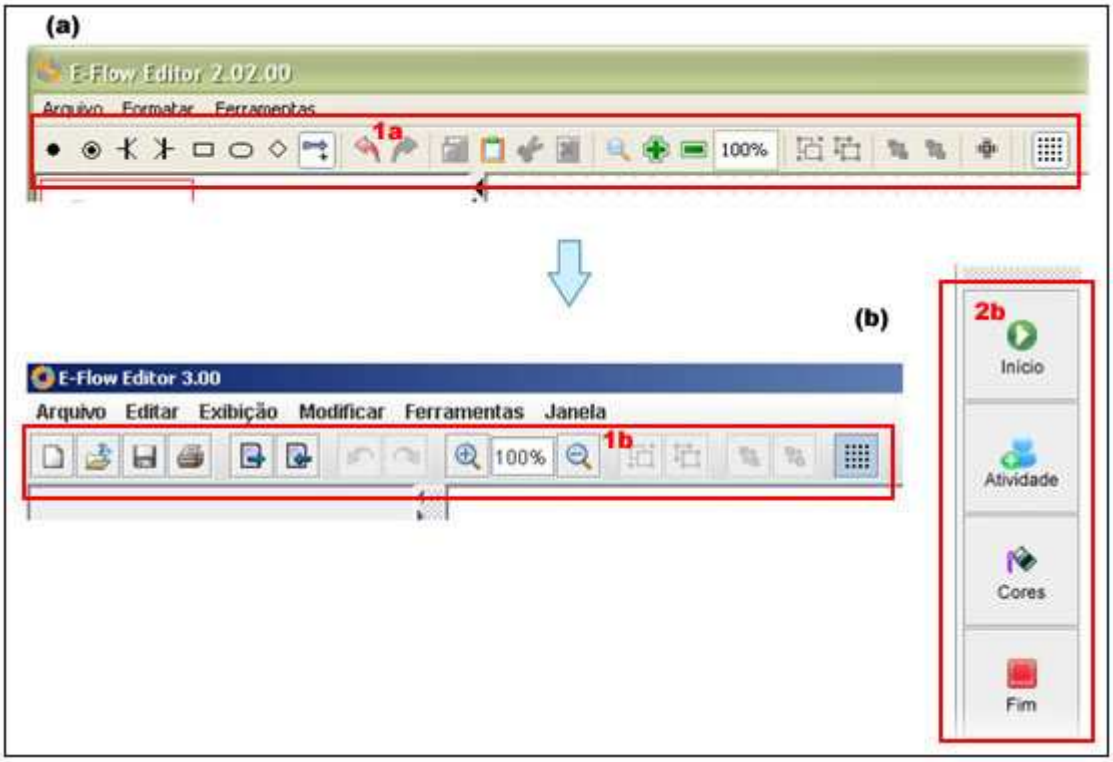

Figura 29. Barras de Ferramentas da versão anterior e atual, respectivamente.

Como já foi descrito anteriormente, a ilustração (a) refere-se à versão antiga da ferramenta, e a ilustração (b) refere-se à ferramenta atual. Pode-se analisar algumas diferenças abaixo:

● A ilustração (a) é composta de apenas uma barra de ferramentas, definida na ilustração como 1a, já a ilustração (b), apresenta duas barras de ferramentas 2a e 2b, a barra 1a, representada na ilustração (a) possui os elementos básicos para a criação dos processos, juntamente com os elementos básicos de manipulação e utilização da ferramenta, por exemplo, voltar e avançar, já a ilustração (b) possui duas barras a 2a, que contém os elementos básicos de utilização e manipulação da ferramenta, ressaltando que 2b apresenta os ícones que apresentam as funcionalidades mais utilizadas pelo sistema, onde também foram retirados alguns ícones desnecessários, os quais não eram utilizados com muita freqüência, nota-se

também que foi adicionada a descrição referente aos elementos, desta forma facilitando a identificação.

● Outra funcionalidade presente na nova ferramenta foi à opção de arrastar a barra de ferramenta 2b por toda a região do aplicativo, de acordo com as necessidade do usuário, podendo esta ser inserida em qualquer uma das extremidades da tela, caso o usuário queira mudar o estado da aplicação.

4.2.5 Área de Modelagem/Elementos Gráficos

A área de modelagem é a região a qual permite os usuários realizarem a editoração dos processos. Nesta região são inseridos elementos gráficos que são utilizados para a modelagem dos processos, baseado nos conceitos da diagramação da UML, nesta versão foram acrescentadas cores aos elementos e nova representatividade para alguns elementos como formatação dos elementos de inicio e termino de processos, e inclusão de uma imagem no elemento referente à atividade, tornando-o mais atrativo e perceptível, pois este representa uma ação muito importante no processo de construção dos diagramas. A figura 30 demonstra a comparação entre os elementos gráficos.

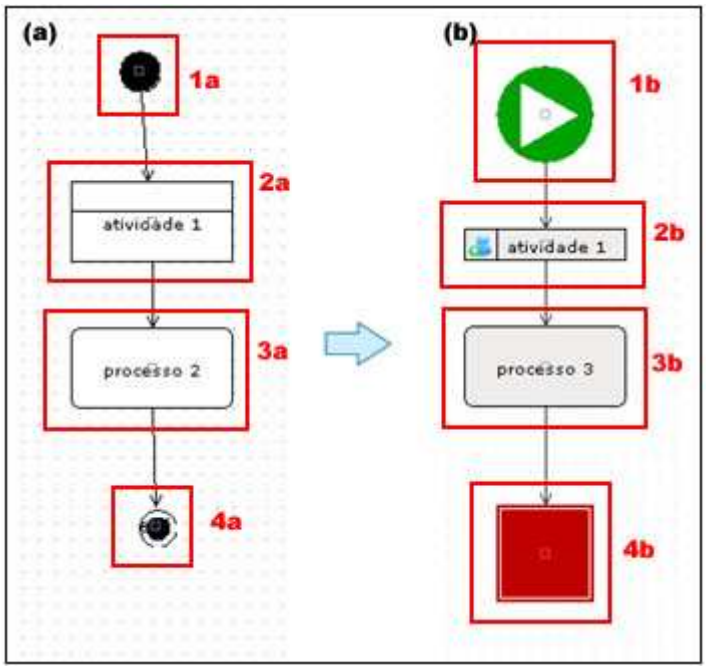

Figura 30. Comparação entre os elementos gráficos disponíveis na versão anterior e na atual respectivamente.

Como já foi descrito anteriormente, a ilustração (a) refere-se à aos elementos que compõem em processo na versão antiga da ferramenta, e a ilustração (b) refere-se

aos elementos que compõem o processo na ferramenta atual. Foram analisadas as seguintes diferenças abaixo:

● A ilustração (a) é composta de elementos mais simples, definidos na ilustração como 1a (início do processo), 1b (atividade), 1c (sub-processo) e 1d(fim do processo) , já a ilustração (b), é composta pelos seguintes elementos 2ª(início do processo), 2b(atividade), 2c(sub-processo) e 2d(fim do processo). A ilustração (b), quando comparado com a ilustração (a), apresenta elementos mais ricos no que se refere à visualização e entendimento dos signos, de forma a facilitar o entendimento tornando a ferramenta mais prazerosa durante a sua utilização.

- O elemento 1a, que é representado como 1b na nova versão, refere-se ao início do processo, nele é perceptível a diferença, com a utilização de cores e signos que facilitam a idéia de inicio, o verde que é conhecido como liberado e a seta (*play,* como já é conhecido e utilizado em muitos aparelhos eletrônicos) sendo que este elemento corresponde com o ícone responsável pela sua inserção no processo.

- O elemento 2a, que é representado como 2b na nova versão, refere-se a atividade, neste elemento foi destacado com a inserção de uma imagem ilustrativa, correspondente com o ícone responsável pela sua inserção no processo.

- O elemento 3a, que é representado como 3b na nova versão, refere-se a outro processo já definido que é utilizado como um sub-processo por este processo “pai”. Este elemento continua basicamente o mesmo, apenas foi definida uma cor na sua inicio de criação, de forma a permitir que este elemento seja mais chamativo.

- O elemento 4a, que é representado como 4b na nova versão, refere-se ao término do processo, nele é perceptível a diferença, que foi representado pelo quadrado, que é um signo muito conhecido como pare (*stop,* como já é conhecido e muito utilizado nos aparelhos eletrônicos), juntamente com a cor vermelha que passa a mesma idéia.

4.3 DESENVOLVIMENTO/IMPLEMENTAÇÃO

Nesta seção serão descritas as etapas que antecederam o desenvolvimento da ferramenta, como também as tecnologias utilizadas durante este processo.

O processo de desenvolvimento iniciou com a especificação da funcionalidade e do modelo de interação, fabricação do protótipo da interface, avaliação do protótipo e desenvolvimento do produto final. Etapas estas que serão descritas a seguir.

4.3.1 Especificação da funcionalidade e modelo de interação

A ferramenta foi especificada para a realização da editoração de processos de workflow. Estes processos são definidos utilizando-se a padronização do diagrama de atividade da UML e contemplar atributos estendidos para a definição de processos normativos. Os atributos estendidos fazem parte da definição de uma meta-linguagem que apóia estruturação dos documentos da qualidade. Os processos desenhados no E-Flow poderão ser exportados no padrão BPL, isto permitirá que os processos possam ser utilizados em uma ferramenta de BPM.

A interação será realizada com a utilização de controles como o mouse e o teclado, estes irão permitir que o usuário realize o mapeamento tarefa-ação, isto é, realização de um ou vários conjuntos de tarefas para a realização de uma ação disponível pela ferramenta. Para que haja esta interação será utilizados recursos conhecidos como signos, os quais permitem a que recursos e componentes presentes na GUI sejam entendidos por parte dos usuários de forma simples e objetiva.

4.3.2 Protótipo

A abordagem utilizada para a prototipação no processo de desenvolvimento de software foi a prototipação descartável. Foi desenvolvida para demonstrar a real necessidade de atualização da ferramenta e demonstração da utilização de recursos GUI’s e depois foi descartado. É interessante comentar que o processo de desenvolvimento utilizado na versão anteriormente não previa estas questões de usabilidade, tanto quanto esta nova versão desenvolvida.

Durante o processo de desenvolvimento deste protótipo foram realizados os seguintes passos:

4.3.3 Estabelecimento dos objetivos do protótipo

O protótipo teve como objetivo demonstrar uma nova interface visual e interativa para com os seus usuários. De forma a facilitar a interação e utilização do sistema.

4.3.4 Desenvolvimento

O protótipo do Max-Flow, foi desenvolvido utilizando a linguagem Java versão 1.5.0_06, fato de oferecer pacotes de classes prontos que facilitam o desenvolvimento de um aplicativo GUI, como também outros recursos utilizados na ferramenta foram desenvolvidos utilizando a mesma linguagem, como o JGraph versão 5.10.00.0, uma biblioteca de classes madura, com características interessantes no que se diz respeito à visualização gráfica.

Dentre os pacotes disponibilizados para a criação do protótipo podemos destacar o pacote Swing. Que foi utilizado através da importação de suas classes (javax.swing). Este pacote possui diversos componentes, que foram utilizados no desenvolvimento do protótipo da GUI. Este pacote permitiu a especificação da aparência e comportamento uniforme em todas as plataformas. Além disso, o Swing permitiu fornecer uma aparência e um comportamento particulares para cada plataforma, como também a alteração da aparência e o comportamento enquanto o programa está sendo executado.

No protótipo foi definida a área de modelagem da ferramenta, mesmo estas sem suas funcionalidades completas, mas permitindo que a aparência visual fosse definida. Para a construção desta área de modelagem foi utilizada a biblioteca de classes JGraph. Esta biblioteca possui características interessantes no que se diz respeito à visualização gráfica. Escrita em Java, para ser inteiramente um componente compatível do balanço, visualmente e em sua arquitetura do projeto. Pode ser funcionado em qualquer tipo de sistema operacional que tenha instalado na máquina versão 1.4 do Java ou mais atrasado.

Neste protótipo algumas funcionalidades foram mostradas pelo fato da ferramenta já possuir todas as atribuições necessárias para a sua utilização, porém ela não

adotava diversos conceitos necessários para construção de aplicativos GUI’s, ou não foram aplicados da maneira mais adequada.

É importante ressaltar que o protótipo não foi utilizado pelos usuários da antiga ferramenta, apesar da ferramenta já ser comercializada e utilizada, e não ser alvo de crítica pelos seus usuários, pois atendem as necessidades necessárias. Pois os estudos de usabilidade permitiram que esta fosse apenas desenvolvida tendo como base a fundamentação teórica referente à construção de ferramentas GUI. No anexo I detalha-se os aspectos de projeto e implementação da ferramenta Max-Flow

CONSIDERAÇÕES

Neste capítulo, apresentam-se as considerações sobre como desenvolver uma ferramenta de forma a permitir uma melhor interação homem-máquina, visto que isto é um fator crítico para o sucesso na construção de softwares. A seguir serão apresentados os resultados e as contribuições obtidos com a pesquisa.

Ao longo do estudo deste trabalho foi realizado o estudo de usabilidade da ferramenta E-Flow, sobre a perspectiva dos aspectos da Interação Homem Computador. Este estudo favoreceu a especificação e implementação da ferramenta, que utilizaram conceitos principalmente de interação e usabilidade para prover a apresentação de uma nova interface.

De forma a permitir uma melhor interação foi desenvolvido um protótipo, que mostrou à sua importância em um processo de desenvolvimento de software. Esta importância se deu, sobretudo, pelo fato dele fazer parte do processo iterativo, cujo objetivo final é o usuário. A utilização de protótipos é uma prática comum para avaliar a interpretação dos requisitos de projeto, as possibilidades de solução e as soluções efetivamente sugeridas.

A correlação com a qualidade em relação aos sistemas computacionais, faz-se referência automaticamente à questão da usabilidade, pois os aspectos que indicam uma melhor usabilidade defendem questões como: a facilidade de aprendizado do sistema, facilidade de uso, satisfação dos usuários, possibilidade de o usuário acrescentar e modificar as funções e o ambiente inicial do sistema e a produtividade. Estes aspectos nortearam as necessidade de refatoração da interface da ferramenta E-Flow, gerando uma nova ferramenta denominada Max-Flow.

O Max-Flow foi desenvolvido para ser facilmente usada pelo usuário, fornecendo seqüências simples e consistentes de interação, mostrando claramente alternativas disponíveis a cada passo de interação, sem confundir nem deixá-lo inseguro. Assim, gerou-se uma aplicação mais intuitiva, possibilitando que o usuário se fixe somente no problema que deseja resolver.

Pode-se constatar que o processo de desenvolvimento baseado em prototipação utilizado na pesquisa foi adequado, pois o requisito de refatoração da ferramenta E-Flow sobre a ótica da interação e da usabilidade foram alcançados. O método proposto neste trabalho com três etapas se mostrou adequado A seqüência das etapas adotadas foram: o estudo de usabilidade da ferramenta, seguido do desenvolvimento do protótipo, que foi descartado, dando inicio ao processo de refatoração e atualização da ferramenta, sendo que a nova versão da ferramenta colocou em prática as questões de interação e usabilidade, elementos fundamentais para a realização de uma melhor interação homem-máquina.

Para continuidade e ampliação da temática do trabalho são sugeridas as seguintes abordagens:

- A realização de testes e acompanhamento de usabilidade, juntamente com os usuários da ferramenta, observando as questões de interação, verificando e destacando os princípios que fundamentam as questões de interação;

- Estudo da viabilidade de desenvolvimento da ferramenta no ambiente *web*.

REFERÊNCIAS BIBLIOGRÁFICAS

BPM X Workflow, Disponível em < http://www.cryo.com.br/Site/Files/Introducao_BPM_White_Paper.pdf >. Acessado em 15 JAN 2007.

CERVO Armando L.; BERVIAN Pedro A.; Metodologia Científica: São Paulo: Pearson Prentice Hall, 2002

COSTA, Vitor Franco. E-flow: Uma solução de *workflow* para integração da gestão de processos com a documentação da qualidade usando xml. 2004. 75 f. Monografia (Bacharel em Informática) – Curso de bacharelado em informática, Universidade Católica do Salvador.

FOWLER, Martin; Refatoração: aperfeiçoando o projeto do código existente / trad. Acaun Fernandes. Porto Alegre: Bookman, 2004

H. M. Deitel, P. J. Deitel, trad. Carlos Arthur Lang Lisboa Java como programar 4º edição: Porto Alegre: Bookman, 2003

Introdução ao BPM, Disponível em < http://www.cryo.com.br/Site/Files/Introducao_BPM_White_Paper.pdf >. Acessado em 15 JAN 2007.

JBOSS jBpm 2006 Disponível em: <http://www.jboss.com/products/jbpm>. Acesso em: 28 AGO 2006.

JGraph and JGraph Layout Pro User Manual, Disponível em < http://www.jgraph.com/pub/jgraphmanual.pdf>. Acessado em 15 JAN 2007.

LARMAN, Craig; Utilizando UML e padrões: uma introdução à analise e ao projeto orientados a objetos e ao desenvolvedor iterativo 3º edição, 2007.

MINASI, Mark. Segredos de projeto de interface gráfica com o usuário. Tradução Flavio Eduardo Morgado. Rio de Janeiro: Infobook,, 1994.

SEGUNDO, Inaldo. Uma solução de gerência de projetos, baseado na junção dos conceitos do PMI e BPM. 2006. 68 f. Monografia (Bacharel em Sistemas de Informação) – Curso de bacharelado em Sistemas de Informação, Centro Universitário da Bahia.

Pointer Tecnologia da Informação, O que é prototipação, Disponível em < http://www.pointerti.com/prototyping.htm >. Acessado em 15 JAN 2007.

Prototipação de software, Disponível em < http://www.dcce.ibilce.unesp.br/~ines/cursos/eng_soft/aula07.pdf >. Acessado em 15 JAN 2007.

WfMC XPDL Disponível em: <http://www.wfmc.org/standards/XPDL.htm>. Acesso em: 13 SET 2006.

XML, Disponível em < http://pt.wikipedia.org/wiki/XML >. Acessado em 15 JAN 2007.

ANEXO A – CLASSES QUE FORAM ALTERADAS E INSERIDAS NO PROJETO

Este anexo apresenta as classes que foram adicionadas e alteradas durante o processo de implementação do projeto.

Na figura 31 podem ser visualizadas as classes que foram adicionadas no projeto.

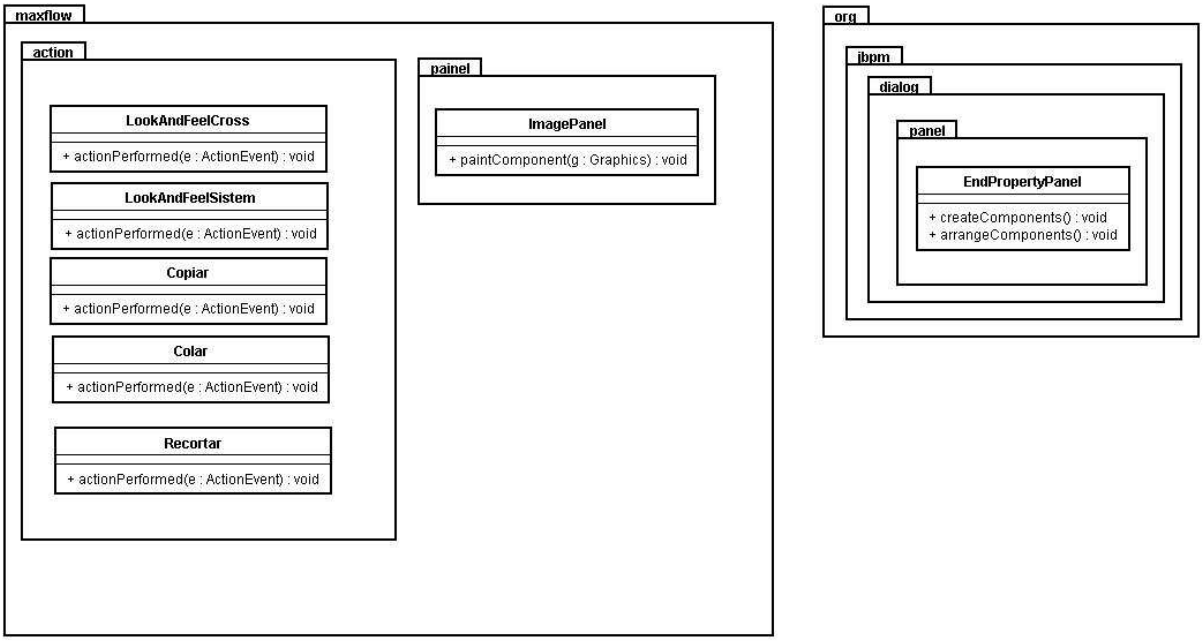

Figura 31. Diagrama de Classe que representa as novas classes adicionadas ao projeto.

Durante o processo de criação foi definida uma arquitetura lógica para classes que definiram novas ações e utilização da arquitetura já existente. Segundo Larman, a arquitetura lógica é uma organização em larga escala das classes de software em pacotes (ou espaços de nomes), subsistemas e camadas.

Para a arquitetura criada foi utilizada a escala das classes de software em pacotes e camadas. Que é definida por Larman como um agrupamento de granularidade muito grossa de classes, pacotes ou subsistemas que tem responsabilidade coesiva sobre um tópico importante do sistema.

Algumas classes foram adicionadas no pacote maxflow, mesmo sendo estas internamente organizadas utilizando o mesmo conceito de camada. Nesta foram adicionados dois pacotes, uma para determinar as classes que referencia novas ações e um para definir um painel com algumas propriedades definidas.

A classe que foi criada no pacote já existente, levou em consideração a granularidade com classes já existentes no pacote panel (org.jbpm.dialog.panel).

Durante o processo de desenvolvimento foram modificadas diversas classes de pacotes distintos. Estas classes serão relatadas de acordo com a sua arquitetura lógica, desta forma permitindo um maior detalhe de acordo com a sua granularidade.

A primeira classe modificada refere-se à tela de login, EflowLogin.java. Ela encontra-se no seguinte pacote eflow.splash. Neste pacote apenas esta classe foi alterada. Ela foi alterada com o intuito de oferecer uma melhor aparência visual para tela e refatoramento do código.

Nesta Classe os métodos foram renomeados para facilitar a manutenção e entendimento do código e foram adicionados alguns métodos com novas funcionalidades. Estes podem ser visualizados na figura 32.

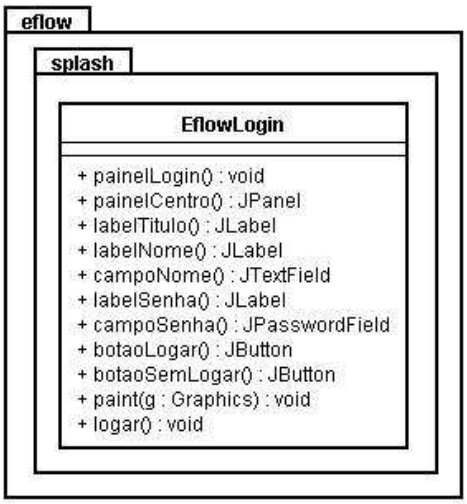

Figura 32. Diagrama de Classe que representa a arquitetura lógica da classe EflowLogin, contendo os métodos que foram adicionados e alterados.

Outra classe do pacote eflow foi alterada a Configuration.java (eflow.config), ela foi alterada para permitir que seja definido *look and feels* diferentes para a ferramenta. Pode ser visualizado na figura 33.

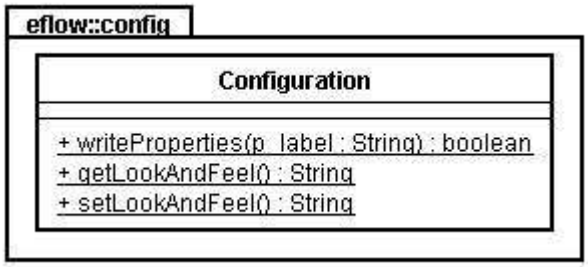

Figura 33. Diagrama de Classe que representa a arquitetura lógica da classe Configuration, contendo os métodos que foram adicionados e alterados.

As classes do jbpm que foram alteradas serão mostradas a seguir começando pela ProcessDesigner, classe fundamental da ferramenta. Nesta foram realizadas as principais modificações, pois ela é a classe principal da ferramenta. Vale destacar que além da modificação de alguns métodos existentes, outros foram adicionados para oferecer uma maior interação e aderência da ferramenta. Esta classe foi representada na figura 34.

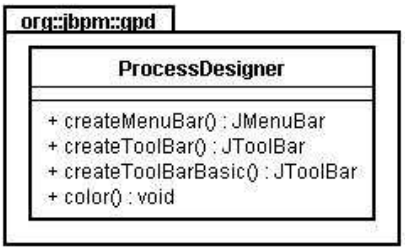

Figura 34. Diagrama de Classe que representa a arquitetura lógica da classe ProcessDesigner, contendo os métodos que foram adicionados e alterados.

No pacote panel (org.jbpm.gpd.panel) foram realizas alterações em diversas classes. Classes estas que se referem ao painel de propriedades do referente a um elemento. O diagrama de classe da figura 35, ilustra o este pacote.

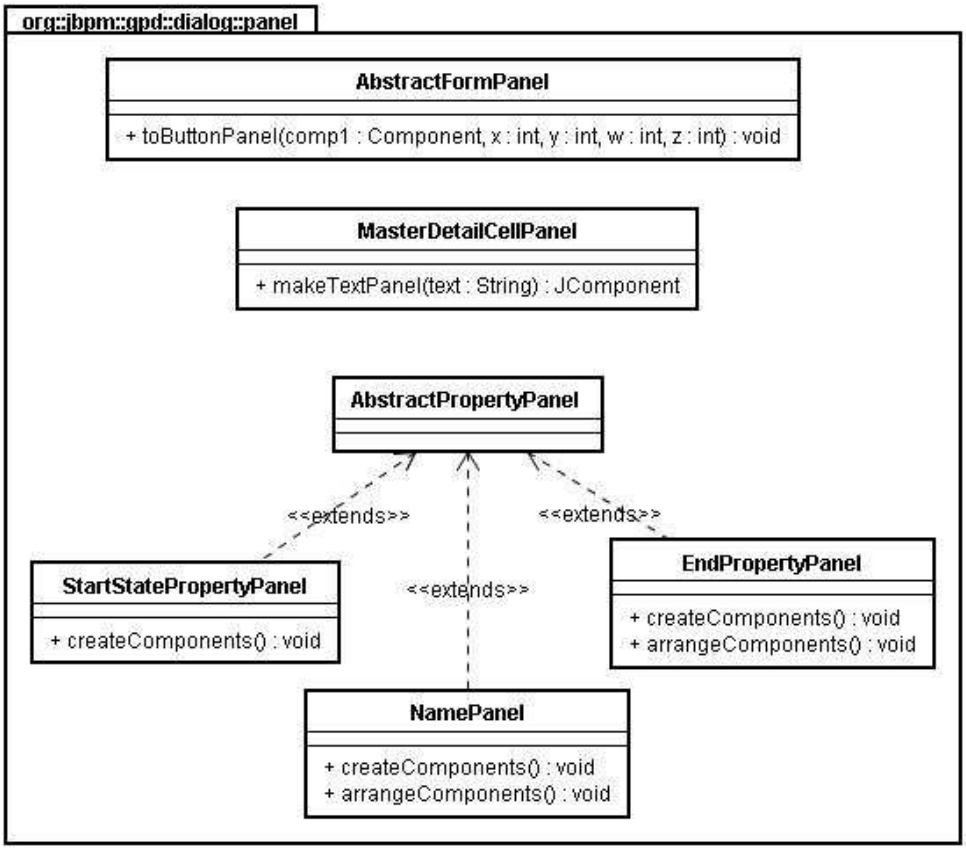

Figura 35. Diagrama de Classe que representa a arquitetura lógica do pacote panel, contendo as classes seus respectivos métodos que foram adicionados e alterados.

Dentro deste pacote temos o pacote detailpanel (org.jbpm.gpd.dialog.panel.detailpanel), que teve algumas de classes modificadas. Dentre elas estão: SubProcessPanel, TransitonPanel, ProcessPanel, StatePanel. Estas classes possuem as funções dos painéis definidas para cada elemento. Na figura 36 podem ser visualizados as classes modificadas.

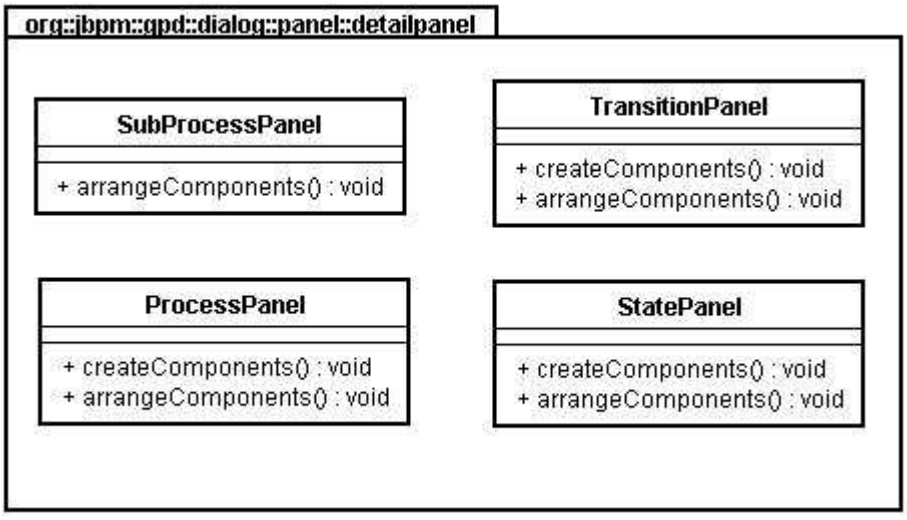

Figura 36. Diagrama de Classe que representa a arquitetura lógica do pacote detailpanel, contendo as classes seus respectivos métodos que foram adicionados e alterados.

O pacote renderer (org.jbpm.gpd.renderer), teve a alteração na classe responsável pelo elemento mais utilizado para a realização de um processo através da ferramenta, a atividade. A figura 37 ilustra a classe que foi alterada.

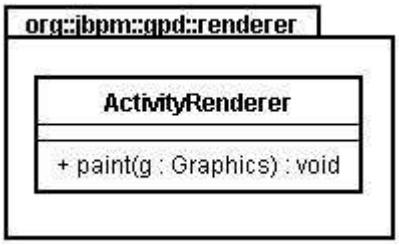

Figura 37. Diagrama de Classe que representa a arquitetura lógica do pacote renderer, contendo a classe seu método que foi alterado.

O pacote view (org.jbpm.gpd.view) teve duas classes alteradas, estas foram alteradas de forma a permitir que os elementos de inicio e término dos um processo fosse modificados. As classes alteradas neste pacote pode ser visualizado na figura 38.

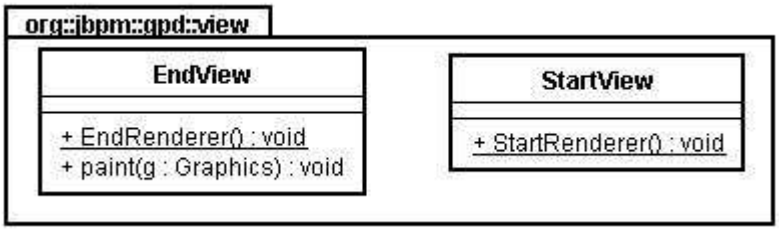

Figura 38. Diagrama de Classe que representa os elementos gráficos.

Outro pacote teve alteração em duas classes para está modificando propriedades dos seus elementos, estes elementos foram o de início e término dos processos. As classes podem ser visualizadas na figura 39.

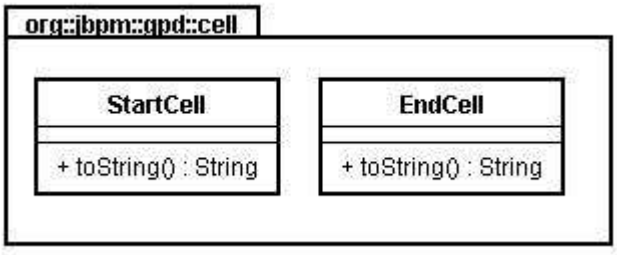

Figura 39. Diagrama de Classe que representa as propriedades dos elementos gráficos.