GERSON PESENTE FOCKING

PROPOSTA DE RECURSOS ERGONÔMICOS PARA AMBIENTES DE AUTORIA E PARA BIBLIOTECAS DE OBJETOS DE INTERAÇÃO REUTILIZÁVEIS

FLORIANÓPOLIS
2000

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

PROPOSTA DE RECURSOS ERGONÔMICOS PARA AMBIENTES DE AUTORIA E PARA BIBLIOTECAS DE OBJETOS DE INTERAÇÃO REUTILIZÁVEIS

Dissertação submetida à Universidade Federal de Santa Catarina, como parte dos requisitos para a obtenção do Grau de Mestre em Ciência da Computação.

GERSON PESENTE FOCKING

Florianópolis, fevereiro de 2000

PROPOSTA DE RECURSOS ERGONÔMICOS PARA AMBIENTES DE AUTORIA E PARA BIBLIOTECAS DE OBJETOS DE INTERAÇÃO REUTILIZÁVEIS

Gerson Pesente Focking

' Esta dissertação foi julgada adequada para a obtenção do Título de Mestre em Ciência da Computação, Área de Concentração em *Sistemas de Computação*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.'

Prof. Walter de Abreu Cybis, Dr. Eng.
Orientador

Prof. Fernando Álvaro Ostuni Gauthier. Dr. Eng.
Coordenador do Programa de Pós-Graduação em Ciência da Computação

Banca Examinadora

Prof. Walter de Abreu Cybis, Dr. Eng.
Orientador / Presidente

Prof. Raul Sidnei Wazlawick, Dr. Eng.
Examinador

Prof. Vitorio Bruno Mazzola, Dr.
Examinador

AGRADECIMENTOS

Agradeço à Deus, que sempre guiou meus passos, dirigiu minha vida e colocou em meu caminho pessoas que estenderam-me suas mãos, colaborando e incentivando na realização deste trabalho. Em especial agradeço:

Meus pais, Ezequiel e Prima, pelo apoio financeiro e moral. Meus irmãos, Ivan e Deize, pela amizade e o carinho que sempre me presentearam;

Minha noiva, Maria, a amiga de todas as horas;

Professor Walter de Abreu Cybis, que mais parece um irmão, pela orientação, apoio e amizade;

Os amigos do LabIUtil, Vera, Danieli, Mário, Wellington, Luciano, Kelly, Gustavo... caboclada amiga;

Os amigos que compartilharam comigo esta empreitada: Gentil, Walter, M. Lúcia, Sandro, Sinval, Sâmela, Rosângela, Laura, Désiré, e demais colegas....;

Agradeço, à equipe de professores do LSC/UFSC, Raul, Edla, Mariani e os colegas Bernd e Rivero pelo apoio, amizade e ensinamentos;

Ao INE, na pessoa do prof. Jovelino Falqueto;

À UFSC e ao PPGCC, pela oportunidade;

Às amigas e sempre tão prestativas Verinha e Valdete, vocês são dez;

A toda equipe de professores do PPGCC, que nunca negaram-me atenção.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciência da Computação

PROPOSTA DE RECURSOS ERGONÔMICOS PARA AMBIENTES DE AUTORIA E PARA BIBLIOTECAS DE OBJETOS DE INTERAÇÃO REUTILIZÁVEIS

Gerson Pesente Focking

Orientador: Prof. Dr. Walter de Abreu Cybis;
Área de Concentração: Sistemas de Computação;
Palavras-Chave: Projeto de interfaces humano-computador, Ergonomia de interfaces, Ferramentas de desenvolvimento;

RESUMO

O primeira meta deste estudo foi analisar e compreender como a engenharia de software, as técnicas, métodos e ferramentas formais necessárias à produção de software, tratam o desenvolvimento da interface do usuário. Esta análise evidenciou que as metodologias atualmente aplicadas na concepção de interfaces do usuário, não utilizam técnicas que busquem efetivamente especificar e avaliar os principais requisitos do usuário e da tarefa a ser informatizada. Com base nestes resultados, são apresentadas três técnicas de análise e projeto que buscam complementar estas metodologias tradicionais, fornecendo suporte para a integração de fatores humanos nas interações homem-computador (MUSE), descrevendo e analisando como a informação é utilizada durante a realização das tarefas (AIU) e gerando interfaces através de modelos de dados baseados em regras ergonômicas (GENIUS). A partir da compreensão destes métodos, realizou-se um estudo investigatório sobre ferramentas para desenvolvimento de interfaces, buscando avaliar o estado da arte destas ferramentas, verificando e apresentando as características de algumas delas, seus objetos de interação, suas potencialidades, funcionalidades e falhas. Este estudo permitiu formalizar várias considerações sobre estas ferramentas, que mescladas com requisitos ergonômicos para interfaces do usuário encontrados em vários autores e guias de estilo (Bastien & Scapin, Smith & Mosier, Motif, Microsoft, LabIUtil), formam um quadro de propostas de recursos ergonômicos para um possível ambiente de autoria e seus respectivos objetos de interação, na intenção de construir uma ferramenta com características de utilização ergonômicas, de uso fácil e que produza interfaces dentro de normas e padrões também ergonômicos.

PROPOSAL OF ERGONOMIC RESOURCES FOR AMBIENT OF CRIATION AND FOR LIBRARIES OF OBJECTS OF INTERACTION REUSED

Gerson Pesente Focking

February of 2000

Advisor: Prof. Dr. Walter de Abreu Cybis;
Area of Concentration: Systems of Computation;
Key words: Design of Human Computer Interfaces,
Interfaces Ergonomics , Tools of Development;

ABSTRACT

The first goal of this study went analyze and to understand as the software engineering, the techniques, methods and necessary formal tools to the software production, treat the development of the user's interface. This analysis evidenced that the methodologies now applied in the conception of the user's interfaces, they don't use techniques that look for indeed to specify and to evaluate the user's main requirements and of the task to be computerized. With base in these results, they are presented three analysis techniques and project that look for complemental these traditional methodologies, supplying support for the integration of human factors in the interactions man-computer (MUSE), describing and analyzing as the information is used during the accomplishment of the tasks (AIU) and generating interfaces through models of data based on ergonomic rules (GENIUS). Starting from the understanding of these methods, has took place a study to investigate on tools for development of interfaces, looking for to evaluate the state of the art of these tools, verifying and presenting the characteristics of some of them, its interaction objects, its potentialities, functionalities and flaws. This study allowed to formalize several considerations on these tools, that mixed with ergonomic requirements for the user's interfaces found in several authors and guides of style (Bastien & Scapin, Smith & Mosier, Motif, Microsoft, LabIUtil), they form a collection of proposals of ergonomic resources for a possible ambient of criation and its respective interaction objects, in the intention of building a tool with ergonomic use characteristics, of easy use and that produces interfaces inside also of norms and patterns ergonomic.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO I

1. INTRODUÇÃO

As interfaces humano-computador são componentes muito importantes nos sistemas informatizados, pois para a maioria dos usuários, elas representam todo o aplicativo, englobando os objetos de interação que a compõem e as funções e procedimentos que manipulam os dados apresentados através destes objetos. Projetar e implementar interfaces de qualidade ainda é um desafio, devido aos fatores com diferentes perspectivas que devem ser contemplados, entre eles, os requisitos da tarefa, os aspectos inerentes aos dados manipulados, as restrições tecnológicas, mas principalmente, os fatores relativos ao usuário, são os que apresentam maiores obstáculos para a concepção de interfaces humano-computador. Construir interfaces para pessoas com diferentes aptidões físicas, que possuem objetivos distintos e diversos níveis de conhecimento é uma tarefa que necessita ser ancorada em metodologias consagradas e conduzida por profissionais com conhecimento e experiência de acordo.

A usabilidade é fator preponderante em sistemas informatizados, pois deve ser o objetivo de todo aplicativo, permitir que o usuário realize mais facilmente suas tarefas. Interfaces intuitivas e de uso fácil podem ser produzidas quando durante o projeto é promovida a participação do usuário final, seus requisitos e os da tarefa são identificados, analisados e projetados, os fatores humanos são considerados, sistemas semelhantes são avaliados e um protótipo inicial da interface é criado, para progressivamente receber alterações, refinamentos e avaliações.

Este trabalho procura contribuir com a engenharia de software, especificamente no desenvolvimento das interfaces do usuário, através da apresentação de técnicas de análise e projeto, que visam complementar os métodos tradicionais de concepção de sistemas interativos, com abordagens voltadas à criação de interfaces dentro de padrões de usabilidade. Realiza-se também, um estudo a respeito de ferramentas para desenvolvimento de interfaces, onde são apresentadas características técnicas do ambiente

de funcionamento, dos objetos de interação disponibilizados e dos tipos de interfaces possíveis de desenvolver.

Muitas das ferramentas apresentadas neste trabalho, providenciam um bom nível de apoio para o desenvolvimento de uma grande faixa de aplicativos, independentemente ou em conjunto com outras ferramentas. É importante reconhecer que as ferramentas de desenvolvimento de interfaces, talvez não apresentem a receita ideal e nem possam prever a concepção de interfaces não apropriadas e possivelmente de difícil uso para um aplicativo. Entretanto, elas podem e frequentemente apoiam a prototipagem rápida de interfaces, permitindo que o projetista realize testes já nos primeiros estágios do ciclo de vida do projeto, verificando se os princípios da interação humano-computador estão sendo respeitados e se a semântica do aplicativo está sendo atingida.

Este trabalho busca contribuir também com a ciência da computação, quando formaliza um conjunto de propostas para o desenvolvimento de um ambiente de autoria, que permita a criação de interfaces com características de usabilidade, por profissionais que não aqueles com conhecimento específico em ergonomia de interfaces humano-computador. Estas propostas surgiram de três diferentes fontes. A primeira, foi o estudo do estado da arte das ferramentas para concepção de interfaces, que permitiu visualizar diferentes abordagens e filtrar funcionalidades julgadas interessantes. A segunda, um CheckList de validação de objetos de interação, desenvolvido pelo LabIUtil (Laboratório de Utilizabilidade da Informática - UFSC), que formata um conjunto de recomendações ergonômicas para os elementos que compõem interfaces do usuário. A terceira, são empirismos, funções observadas em outros sistemas e que presume-se válidas para um ambiente para construção de interfaces.

1.1 ESTRUTURA DO TRABALHO

Esta dissertação é composta por cinco capítulos, estruturados de forma a discutir os temas por área específica: técnicas tradicionais de análise e projeto de sistemas interativos, técnicas específicas para análise e projeto de interfaces e ferramentas de desenvolvimento de interfaces.

O capítulo I apresenta aspectos referentes à origem da proposta, sua estrutura e considerações sobre o estudo realizado, visando justificar sua realização. No capítulo II, são pesquisadas as metodologias tradicionais para o desenvolvimento de sistemas interativos, verificando como é tratada a abordagem ergonômica, durante a concepção da interface do usuário. No capítulo III, focaliza-se as metodologias que visam acrescentar características ergonômicas ao ciclo de desenvolvimento de software. No capítulo IV, busca-se conhecer e avaliar as ferramentas de desenvolvimento de interfaces disponíveis, analisando os objetos de interação disponibilizados para a construção de interfaces e características tais como, usabilidade, funcionalidade, flexibilidade, portabilidade, suporte. No capítulo V, formaliza-se um conjunto de propostas para um ambiente de autoria ergonômico que possa favorecer o projeto e o desenvolvimento de interfaces ergonômicas.

1.2 PROBLEMÁTICA

- As metodologias tradicionais apresentadas pela engenharia de software, não prevêem meios para identificar e definir os fatores humanos (relacionados com o usuário e a tarefa) específicos e necessários para o desenvolvimento de interfaces;
- Inexistência de ferramentas para a construção de interfaces, que apresentem características (recursos) ergonômicas de utilização e que permitam criar interfaces em conformidade com recomendações ergonômicas.

1.3 SOLUÇÕES POSSÍVEIS

- Integração de técnicas e métodos que considerem os fatores humanos durante o desenvolvimento do aplicativo e da respectiva interface com as metodologias tradicionais da engenharia de software;
- Produzir ferramentas que apoiem o desenvolvimento da usabilidade nas interfaces criadas.

1.4 OBJETIVO GERAL

- Pesquisar os métodos e ferramentas envolvidas no processo de concepção de sistemas interativos, avaliando como é tratada a concepção das interfaces do usuário.

1.5 OBJETIVOS ESPECÍFICOS

- Pesquisar as metodologias tradicionais para o desenvolvimento de sistemas interativos, verificando como é tratada a abordagem ergonômica, durante a concepção da interface do usuário;
- Pesquisar metodologias que visam acrescentar características ergonômicas ao ciclo de desenvolvimento de software;
- Pesquisar as ferramentas de desenvolvimento de interfaces disponíveis, analisando seus objetos de interação e características tais como, usabilidade, funcionalidade, flexibilidade, portabilidade, suporte;
- Estudar o material de referência sobre ergonomia de interfaces do usuário, visando formar uma base de conhecimento, que permita avaliar a qualidade ergonômica das ferramentas de desenvolvimento de interfaces;
- Formalizar um conjunto de propostas para um ambiente de autoria ergonômico que possa favorecer o projeto de interfaces ergonômicas.

1.6 LIMITAÇÕES DA PESQUISA

- Buscou-se a compreensão dos métodos e das ferramentas envolvidas no processo de desenvolvimento de interfaces;
- Neste trabalho não será tratada a integração destes métodos com as ferramentas;
- Durante a pesquisa referente à ferramentas para desenvolvimento de interfaces, encontrou-se dificuldades na obtenção de ferramentas para análise.

1.7 JUSTIFICATIVA

- Permitir que profissionais de informática desenvolvam interfaces com qualidades ergonômicas.

1.8 METODOLOGIA

- Revisão da bibliografia especializada em métodos, normas técnicas e ferramentas;
- Análise técnica das ferramentas para desenvolvimento de interfaces disponíveis;
- Elaboração de propostas para o desenvolvimento de um ambiente de autoria.

CAPÍTULO II

2. PROJETO DE INTERFACES DO USUÁRIO – AS ABORDAGENS DA ENGENHARIA DE SOFTWARE

O termo " Interfaces do Usuário" refere-se aos métodos e periféricos que são usados para acomodar a interação entre homem e máquina. As interfaces do usuário podem ter diversas formas, mas sempre realizam duas tarefas fundamentais: enviar informação da máquina para o usuário, e enviar informação do usuário para a máquina. Os problemas que surgem na interação homem/computador expressam-se no diálogo através da interface, pois a forma da interface influencia fortemente como o usuário vê e entende a funcionalidade do sistema.

A interface com o usuário compreende os aspectos do sistema com os quais o usuário entra em contato tanto física quanto perceptual e cognitivamente. A interface inclui o modo como a informação é representada e processada, as funções do computador, procedimentos, sintaxe, organização de dados, feedback, assim como outras ferramentas (documentos em papel e manuais). Neste capítulo, são pesquisadas as metodologias tradicionais para o desenvolvimento de sistemas interativos, verificando como é tratada a abordagem ergonômica, durante a concepção da interface do usuário. Discute-se, também, o projeto de interfaces baseando-se nos fatores humanos, nos modelos utilizados para representar a interface, nos principais princípios formalizados pelas técnicas de manipulação e apresentação de objetos de interação e as arquiteturas que organizam o software de um sistema informatizado.

2.1 AS INTERFACES DO USUÁRIO - HISTÓRICO

John Walker, (Autodesk Inc.) refere-se ao relacionamento que nos primórdios da informática os cientistas estabeleceram com as primeiras gerações de computadores: foi uma relação de tipo direto, baseada na manipulação de interruptores e botões que se

alinhavam no console de computadores imponentes como o UNIVAC, cujo corpo central ocupava o espaço útil de uma enorme sala.

A geração seguinte resolveu o problema da interface através da introdução do sistema de cartões perfurados para mediar a relação do homem com a máquina. Com a introdução dos monitores o processo de operação ficou facilitado, aumentando a rapidez de acesso e a transferência dos dados. Porém, a linguagem dos sistemas operacionais não permitiam uma interface amigável e a operação, acrescida à baixa resolução nos monitores, manteve-se como domínio reservado a especialistas em mainframes.

Com a massificação decorrente da comercialização mundial dos computadores, as questões relacionadas com a interatividade e os modelos de interface tornaram-se evidentes, levando a comunidade científica a aprofundar os estudos destes problemas tanto na sua vertente teórica, através da aplicação de metodologias e técnicas de análise e projeto de interfaces; como na sua vertente prática, através da utilização de ferramentas que auxiliam e gerenciam o desenvolvimento de interfaces automaticamente. A capacidade de processamento gráfico evoluiu rapidamente e o primeiro minicomputador existente no mundo, o DEC-PDP, instalado em 1962 no Massachussets Institut of Technology - M.I.T., deu origem a uma experiência que conduziu à criação do primeiro jogo de computador, pouco tempo depois de ter sido ligado a um monitor.

Spacewar, o nome escolhido para a simulação, aproveitou a tela preta para simular o universo onde se deslocavam naves rudimentares. Não deixa de ser significativo que o diálogo entre a máquina e o homem tenha desde o início apoiado-se sobre os fundamentos da simulação: toda e qualquer representação exige a aceitação de um terreno comum, partilhado por mais de um agente.

As primeiras interfaces baseadas em vídeo, possuíam a forma textual para comunicar a informação do computador para o usuário, enquanto que o usuário utilizava um teclado com aparência de máquina de escrever para diretamente inserir informação no computador. Esta inovação permitiu que usuários não especialistas interagissem mais facilmente com o computador. Apesar da evolução, ainda era exigido dos usuários a memorização de muitos comandos necessários para a realização de determinadas tarefas e

uma quantidade razoável de tempo para treinamento eram essenciais para efetivamente utilizar um computador.

As interfaces do usuário entraram na era moderna quando projetistas do Centro de Pesquisa da Xerox de Palo Alto quebraram o antigo paradigma de interfaces baseadas em caracteres e desenvolveram a chamada "Interface Gráfica do Usuário" (GUI). Existiram dois fatores que separaram o velho paradigma do novo: o primeiro, foi o uso de elementos gráficos para apresentar visualmente junto com texto a informação ao usuário, o segundo, foi a apresentação de várias formas possíveis de interação entre o usuário e a máquina, minimizando a memorização e a entrada manual de comandos. Isto reduziu significantemente o tempo de treinamento necessário para usar o computador e usuários iniciantes estavam hábeis a produzir quase que instantaneamente.

2.2 O PROCESSO DE DESENVOLVIMENTO DE INTERFACES SOB A ÓTICA DA E. S.

O projeto da interface do usuário é o processo de concepção dos objetos de software e hardware que determinam os modos e as estruturas de interação (ou modelos de interação)[1]. A concepção do modelo de interação envolve determinar quais os comandos, o layout de telas e os objetos de interação, definindo o comportamento e a aparência da interface. O projeto e o desenvolvimento de interfaces requer o conhecimento teórico a respeito dos fenômenos envolvidos, os modelos para o processo de desenvolvimento que descrevam as diversas etapas necessárias e como elas são conduzidas e também as diretrizes, técnicas, linguagens, formalismos e ferramentas de apoio a estas etapas. Na execução do projeto utiliza-se de psicologia, fatores humanos, tecnologia dos dispositivos de entrada e saída, tipos de diálogos, análise de tarefas, princípios, guidelines, padrões, regras, protótipos e avaliação de sistemas existentes. O resultado do projeto é a especificação do projeto, geralmente através de especificações formais, protótipos ou ainda documentos informais.

---

[1] Segundo Liesemberg [Liesemberg, 96], *modelo de interação é o conjunto de protocolos que permite ao usuário interagir com a aplicação.* Podemos citar os seguintes modelos de interação: linguagens de comandos, menus, linguagem natural, preenchimento de formulário, manipulação direta e indireta, metáfora do desktop, WIMP e estilo demonstracional.

O objetivo de qualquer projetista de interfaces deve ser projetar e implementar interfaces com qualidade, ou seja, intuitivas, fáceis de usar e que permitam ao usuário maximizar a eficiência e eficácia de sua tarefa durante sua utilização. A melhor maneira de assegurar que um projeto produzirá uma interface de qualidade é usar um processo de desenvolvimento para o projeto como um todo, bem definido e ordenado, atuando sobre o aplicativo (algoritmos e funções) e principalmente sobre o projeto da interface. A melhor interface do usuário não será bem recebida se o aplicativo for pobremente projetado sendo o inverso também verdadeiro.

Na maioria das abordagens estruturadas, o desenvolvimento da interface está inserido no processo de construção do software que a implementa. Existem diversos modelos para o desenvolvimento do software, como o modelo *cascata*, o modelo *espiral*, o modelo *transformação* e o modelo *evolutivo-iterativo* baseado em *prototipação*. Nenhum destes modelos apresenta de forma explícita de quando ou como deve ser conduzido o processo de desenvolvimento das interfaces [Liesemberg, 96].

Todas as metodologias de projeto separam o processo de desenvolvimento em pequenas fases, visando refinar o máximo cada uma delas e buscando acertar mais e errar menos. Não entraremos em detalhes na descrição dessas fases, pois elas já foram bastante discutidas na literatura de engenharia de software (Ex. [Presmann, 95, Sommerville, 96]).

Fase	**Descrição das atividades**
Obtenção dos Requisitos →	Determina os requisitos para a aplicação;
Projeto Conceitual →	Modela o aplicativo conforme a área de concentração;
Projeto Lógico →	Projeta em termos gerais como a aplicação irá operar;
Projeto Físico →	Projeta em termos específicos como a aplicação será construída;
Implementação →	A aplicação é desenvolvida;
Testes de Usabilidade →	Testa a usabilidade do sistema e da interface;

Deve-se considerar entretanto que o projeto de um aplicativo não deve ficar restrito às fases acima descritas, pois cada projeto identifica um processo e uma tarefa em particular, formado por características próprias e que possui métodos adequados para sua realização. A abordagem acima descrita generaliza as diversas metodologias de

desenvolvimento de software e interfaces, analisadas sob a ótica da engenharia de software.

2.3 O PROJETO DE INTERFACES DO USUÁRIO

À medida que os sistemas computacionais evoluem e conquistam um número cada vez maior de usuários, os aspectos da interface estão assumindo um papel de fundamental importância na construção de sistemas informatizados. Esta evolução tem sido uma conseqüência, por um lado, da evolução do hardware dos computadores e, por outro lado, da necessidade em eliminar o "terror tecnológico" criado graças à existência de interfaces de difícil aprendizado, complexas no uso e totalmente frustrantes em boa parte dos casos.

A construção de interfaces ainda não possui um padrão de desenvolvimento sistemático que possa ser aplicado com sucesso garantido. Definir uma interface com a qual o usuário irá interagir e, paulatinamente, refinar esta definição através da avaliação de protótipos/modelos formais, até a obtenção de um projeto satisfatório é uma necessidade. A implementação pode ainda revelar dificuldades que não foram adequadamente contempladas no projeto e, neste caso, novamente o projeto é realimentado com novos dados, modificado e a implementação evolui. A figura 2.1 ilustra esta evolução. Nota-se que avaliação e iteração são também características no ciclo de vida estrela da figura 2.2.

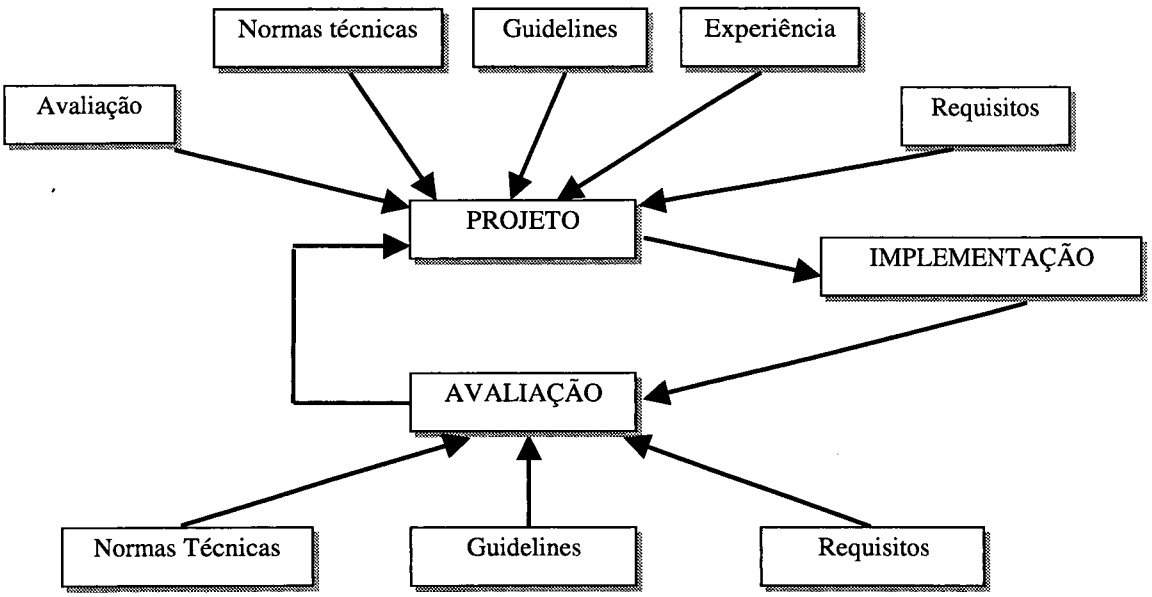

Figura 2.1: Ciclo de vida de uma interface [Liesemberg, 95]

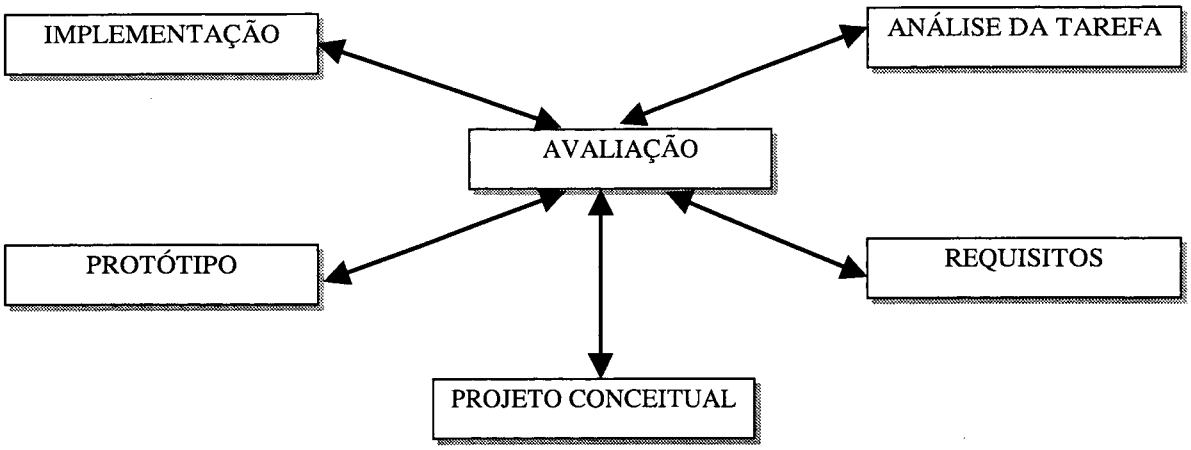

Figura 2.2: Ciclo de vida estrela [Hartson, 89]

2.3.1 OS FATORES HUMANOS

A necessidade de oferecer ao usuário interfaces que efetivamente possam ser utilizadas e que incrementem e agilizem a realização da tarefa, faz com que o projeto das interfaces do usuário deixe de ser visto como mais uma componente do projeto do software como um todo, mas que passe a ser considerada uma atividade onde uma série de fatores técnicos e, principalmente, humanos sejam levados em conta de modo a que o usuário possa explorar completamente, e da forma mais amigável possível, as potencialidades do software.[Mazzola, 98].

O projeto de interfaces do usuário deve contemplar alguns conceitos que serão fundamentais para o encaminhamento de suas atividades no decorrer do processo, entre eles cita-se o mecanismo da percepção humana, particularmente com relação aos sentidos de visão, audição e de tato, que exprimem a capacidade do ser humano em adquirir, armazenar e utilizar as informações. Na maior parte das operações realizadas num computador, o usuário faz uso dos olhos e do cérebro para o processamento das informações, sendo capaz de diferenciar os vários tipos de informação segundo diversos parâmetros visuais (a cor, a forma, as dimensões). A leitura é um outro processo importante na comunicação do usuário com o sistema, apesar da grande tendência em se utilizar recursos gráficos nas interfaces do usuário.

Os diferentes níveis de habilidade das pessoas são um outro aspecto de importância no projeto de uma interface do usuário. O projeto deve dirigir o desenvolvimento da interface para o usuário típico do software. Uma interface que seja orientada e adequada ao uso por um engenheiro pode representar grandes dificuldades de utilização por parte de um trabalhador inexperiente, no caso de sistemas de aplicação vertical (destinados a áreas de aplicação específicas), é importante que a linguagem utilizada suporte termos e conceitos próprios da área.

As tarefas essenciais do sistema devem ser um outro aspecto a ser considerado no projeto da interface, porque em grande parte dos casos, serão as mesmas que eram realizadas no sistema não-automatizado e a introdução do sistema para automatizar uma tarefa, raramente viabiliza a realização de novas tarefas, mas permite que as tarefas antes realizadas manualmente possam vir a ser realizadas de forma mais eficiente e menos dispendiosas.

2.3.2 OS DIFERENTES MODELOS UTILIZADOS NO PROJETO

Dado que as interfaces do usuário envolverão diferentes elementos do sistema computacional (pelo menos o software e pessoas), o projeto deve manipular diferentes modelos para obter um resultado em termos de interface que seja compatível com os fatores técnicos e humanos desejados. Um primeiro modelo a ser definido e utilizado pelo projeto é o modelo de projeto, que está relacionado à representação de todos os aspectos do software (procedimentos, arquitetura, dados).

Um outro modelo a ser desenvolvido pelo projeto é um modelo de usuário, o qual permite descrever o perfil típico do usuário do sistema. Parâmetros como idade, sexo, formação, motivação, capacidades físicas, personalidade, são considerados na construção deste modelo. Além destas características, o grau de experiência e de utilização de um software pode intervir, o que permite classificar os usuários em principiantes, instruídos e intermitentes e instruídos e freqüentes. A localização nesta classificação corresponde ao grau de conhecimento semântico (o conhecimento das funções básicas relativas à aplicação) e conhecimento sintático (a forma de interação com o software) apresentado pelo usuário.

A percepção do sistema corresponde a um modelo estabelecido pelo usuário que representa a sua visão do que será o software. Neste modelo, o usuário descreve como ele imagina que será sua interação com o software: os objetos da aplicação que ele considera importantes (e que devem, portanto, ser ressaltados), as operações que ele desejará realizar sobre os objetos do sistema, como ele pretende visualizar estes objetos na interface do sistema e como ele espera que sejam apresentados os resultados.

A imagem do sistema é uma representação do projetista do que será oferecido na forma de interface e todo o material de apoio à utilização (manuais, videoteipes, livros). A coincidência entre o modelo de percepção do sistema (usuário) e o modelo da imagem do sistema (projetista) permitirá projetar uma interface que satisfaça os anseios de seus usuários e que viabilize a utilização efetiva do aplicativo. O projeto definirá os aspectos da interface de acordo com os modelos produzidos, procurando harmonizar os desejos dos usuários com as limitações impostas pelas técnicas e ferramentas disponíveis e pelos critérios de projeto do sistema.

2.4 ARQUITETURAS DE SOFTWARE

Uma arquitetura de software é um modelo abstrato que estabelece a organização do software de um sistema, ela identifica componentes, divide funções e estabelece um protocolo de comunicação entre eles. A definição de uma arquitetura apropriada é fundamental para possibilitar a evolução e a manutenção do sistema.

As ferramentas de desenvolvimento de interfaces visam oferecer suporte à implementação do software de uma interface, mas em geral não permitem um mapeamento simples e direto das decisões tomadas para o projeto arquitetônico escolhido para a interface. Esta ausência de suporte à implementação de arquiteturas de software é resultante da distancia semântica existente entre os modelos abstratos das arquiteturas e os paradigmas de implementação adotados pelas ferramentas de interface. Em conseqüência, a facilidade de implementação passa a ser uma medida de qualidade das arquiteturas: o valor prático de uma arquitetura é inversamente proporcional à dificuldade de sua implementação com ou sem o apoio de ferramentas de interface.

Por outro lado, se uma ferramenta não organiza adequadamente o código que gera, então os seus benefícios podem ser obscurecidos, especialmente para sistemas interativos de grande porte. Mesmo que uma ferramenta gere automaticamente o código para realizar uma parte significativa das funções da interface, sua utilidade será comprometida se ela não oferece uma estrutura apropriada para a confecção do código restante. Os problemas envolvidos incluem a identificação deste código restante e o seu relacionamento com aquele fornecido pela ferramenta. Desta forma, se a ferramenta não oferece suporte, então a reutilização e a manutenção do código da interface passa a depender da disciplina e habilidade do projetista.

Atualmente existem um grande número de propostas para arquiteturas de software e um número ainda maior de ferramentas de interface. No entanto, enquanto diferentes arquiteturas se mostram adequadas para atender a diversos tipos de requisitos dos sistemas interativos, raras ferramentas oferecem suporte apropriado à implementação destas arquiteturas.

2.4.1 CONTROLE DE DIÁLOGO

É o componente responsável pela sintaxe de interação com o usuário, pelo layout dos objetos de interação e pela comunicação entre os objetos de interação e o restante do software do aplicativo, inclusive as conversões entre os dados manipulados por esses componentes. Esse componente requisita os serviços oferecidos pelo aplicativo, em decorrência da ação do usuário sobre os objetos de interação. As funções do controle de diálogo estão entre as mais complexas de uma interface, por exemplo, um valor obtido pela aplicação, como temperatura, é convertido pelo controle de diálogo em um inteiro que determina a posição de um marcador de termômetro exibido pelo *widget* correspondente ao usuário. O controle de diálogo é responsável também por manter a consistência entre as várias apresentações de um mesmo dado. A seguir são comentados alguns tipos de arquiteturas mais comentadas na literatura.

2.4.2 ARQUITETURA MONOLÍTICA SEQÜENCIAL (MODULARES)

Essa arquitetura foi uma tentativa iniciante para construir sistemas interativos. Ela é conhecida como Modelo de Seeheim [Mark, 85], possui três níveis de abstração, agrupados em três respectivas camadas como mostra a figura 2.3:

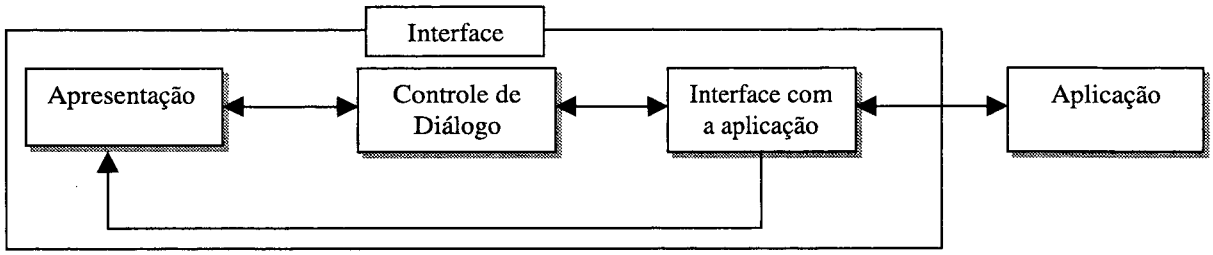

Figura 2.3: Modelo de Seeheim

A idéia básica do modelo é fazer uma analogia com a lingüística: a aplicação exerce o papel semântico, o controle de diálogo trata das questões sintáticas e a interface cuida de aspectos léxicos de interação. Por exemplo, para preencher um formulário, ao usuário é apresentado os campos, ele digita as palavras nos campos (parte léxica), ao confirmar a operação, a(s) palavra(s) serão analisadas sintaticamente pelo controle de diálogo, para avaliar se o formato está de acordo ou não com o especificado (parte sintática), se positivo, o controle de diálogo manda os dados à aplicação e esta se empenha no processamento (parte semântica); caso contrário, o controle de diálogo relata o fato ao usuário, através das mensagens de erro. Podemos ver que o controle de diálogo acaba fazendo o papel de filtro de erros para aplicação.

2.4.2.1 *Arquitetura Modular Slinky / Arch*

É um metamodelo que atente diferentes objetivos dentro de um conjunto de requisitos e funcionalidades de interfaces previamente fixados [Bass et all, 92]. Cada arquitetura, derivada do metamodelo, é especificada de acordo com o conjunto de requisitos do aplicativo.

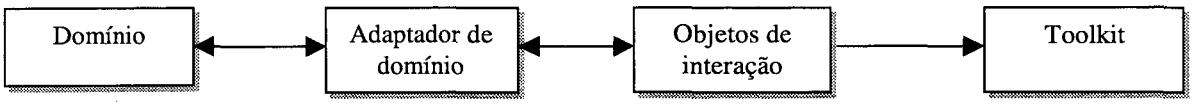

Figura 2.4: O metamodelo Slinky padrão e seus módulos

No metamodelo o domínio realiza as tarefas do aplicativo, o controle de diálogo é desempenhado pelo adaptador de domínio e a apresentação fornece os objetos de interação, implementando-os em diferentes toolkits. As funções desempenhadas por cada módulo são definidas de acordo com os objetivos do sistema que está sendo projetado.

A arquitetura Arch [Bass et all, 92] baseia-se na tecnologia disponível e foi projetada com o intuito de minimizar e localizar os efeitos de alterações na tecnologia das ferramentas envolvidas na construção de interfaces. A independência entre os módulos é enfatizado e uma toolkit pode ser substituída por outra sem interferir nos demais módulos. A figura abaixo ilustra esta arquitetura:

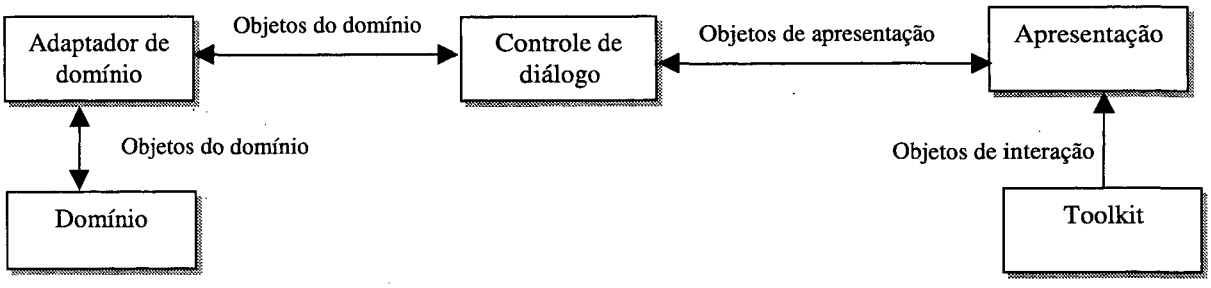

Figura 2.5: O metamodelo da arquitetura Arch

2.4.3 ARQUITETURAS BASEADAS EM AGENTES

Um agente é um sistema de informações completo, inclui receptores e transmissores de eventos, uma memória para manter um estado e um processador para processar os eventos de entrada [Bass & Coutaz, 91]. Agentes comunicam-se com outros agentes, inclusive o usuário. Nestas arquiteturas, um sistema é organizado em um conjunto de agentes especializados que descentralizam as execuções das funções de um sistema interativo e cooperam entre si para realizá-las.

2.4.3.1 *A Arquitetura MVC – Model – View – Controller [Bass & Coutaz, 91]*

É a arquitetura mais referenciada no desenvolvimento de sistemas interativos, divide uma interface em um conjunto de tríades de objetos, cada um pertencente às classes *model*, *view* ou *controller*, ilustradas a seguir:

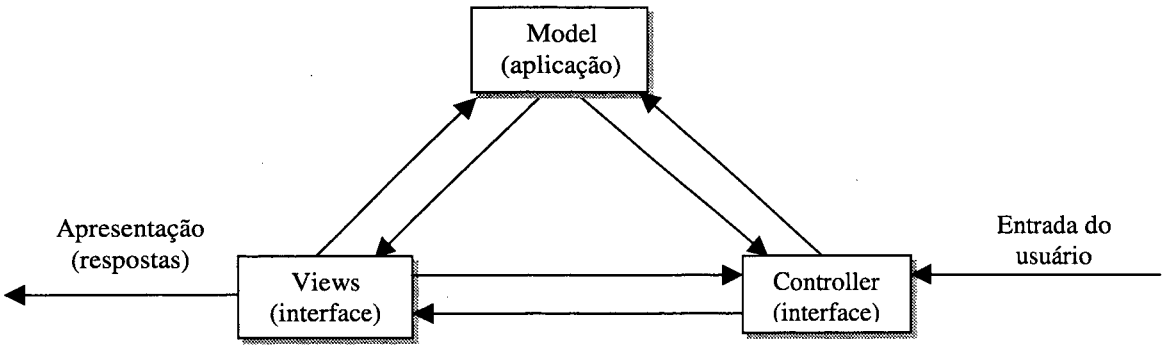

Figura 2.6: Modelo esquemático do modelo MVC

Conforme a figura acima, as considerações pertinentes à aplicação (objeto *Model*, representado, por exemplo, por tabelas em banco de dados), são isoladas daquelas referentes à interface (objeto *Controller* ou *View*). O código que trata das entradas do usuário é da responsabilidade de objetos *Controller* e o código responsável pelas respostas do sistema, são de responsabilidade dos objetos *View*. Por exemplo, quando o usuário pressiona um botão, o respectivo objeto *Controller* chama o método pertinente do objeto *Model* para tratar o evento do usuário. Um objeto *Controller*, chama os métodos de objetos *View* para solicitar a modificação da apresentação em função de mudanças no contexto da interface.

2.4.3.2 *Arquitetura Multiagente*

O modelo multiagente estrutura o sistema interativo numa coleção de agentes especializados que produzem e reagem à eventos. Uma classe especial de agentes que implementam a comunicação com o usuário são chamados de objetos de interação. A idéia principal do modelo multiagente é separar o núcleo funcional da interface em cada nível de abstração, dessa maneira, a preocupação do sistema pode ser distribuído para cada agente

cooperador especializado no assunto. Isso possibilitou os mecanismos de modularidade, paralelismo e distribuição e estes por sua vez possuem as seguintes vantagens:

- Design interativo: com a modularidade, fica mais fácil modificar o comportamento de cada agente sem afetar o resto do sistema, oferecendo alta coesão e baixo acoplamento;
- Aplicações distribuídas: um agente pode ser visto como uma unidade de processamento, assim cada um deles pode ser implementado em estações diferentes e é possível usar uma instância de agente para apresentar um objeto em várias estações, e isso é essencial para implementar um groupware;
- Diálogos em multithreads: um agente pode ser associado a um thread de atividade do usuário, dessa maneira, o usuário poderá suspender uma atividade ou disparar várias atividades em paralelo;
- Um modelo multiagente pode ser facilmente implementado em linguagem orientado a objeto.

A arquitetura PAC [Salber et. all., 94], é um modelo multiagente que estrutura os sistemas interativos como uma hierarquia de agentes. Em cada nível de abstração, o agente é apresentado com três facetas lógicas, mostrado graficamente na figura 2.7:

- Presentation (interface);
- Abstraction (núcleo funcional);
- Control (controle de diálogo), este liga os dois itens anteriores, armazena o estado local para suportar multithread e mantém o relacionamento com outros agentes.

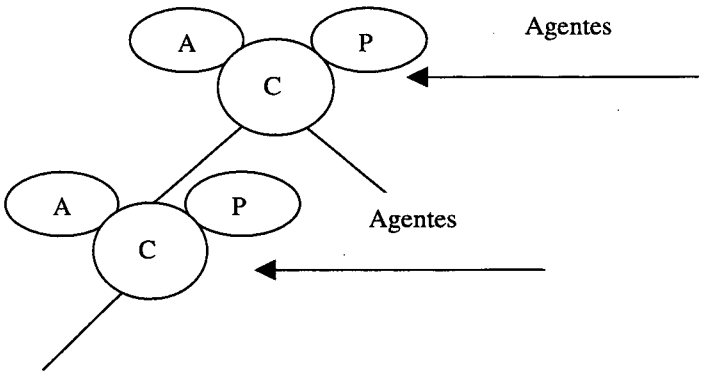

Figura 2.7: A apresentação esquemática da arquitetura PAC

2.5 CONCLUSÕES

Apresentou-se neste capítulo, uma compilação de características que a Engenharia de Software, sob a ótica de vários autores, sugere que sejam contemplados para o desenvolvimento de software com características de usabilidade. A dificuldade atual encontra-se na inexistência de um modelo de desenvolvimento de sistemas interativos que incorpore técnicas específicas para obtenção e análise dos requisitos da interface, permitindo que em paralelo ao avanço da especificação e implementação das funções e procedimentos do sistema, a interface do usuário seja desenvolvida segundo critérios que buscam a usabilidade do sistema como um todo.

Percebe-se também a necessidade de selecionar logo no início do ciclo de vida de um sistema, uma arquitetura que se adeqüe ao sistema e permita a evolução dirigida do processo de desenvolvimento, utilizando os recursos de hardware e software (ferramentas de suporte à implementação do software de interfaces), de maneira mais adequada possível, facilitando a implementação da interface.

Na próxima seção, aborda-se técnicas que se destinam a complementar as metodologias tradicionais de desenvolvimento de software, inserindo no ciclo de vida atividades que buscam contemplar a ergonomia no desenvolvimento de interfaces.

CAPÍTULO III

3. DESENVOLVIMENTO DE INTERFACES - ABORDAGENS ERGONÔMICAS

3.1 PARÂMETROS DE PROJETO

Durante o desenvolvimento de uma interface, existem alguns parâmetros visíveis pelo usuário que devem fazer parte das preocupações da equipe de projeto, e seguramente devem ser analisadas. O tempo de resposta é um parâmetro que, se mal definido (ou omitido num projeto) pode conduzir a resultados frustrantes, ações indesejadas, abandono da utilização do sistema. Não é necessário que o tempo de resposta a comandos do usuário seja o mais curto possível durante toda a interação com o aplicativo, às vezes pode ser interessante um tempo de resposta maior (sem causar, entretanto, aborrecimento ao usuário), de forma a permitir que o usuário reflita sobre as operações que está realizando. O tempo de resposta deve ser projetado observando dois aspectos: a extensão, ou seja, o espaço de tempo entre um comando do usuário e a resposta do sistema e a variabilidade, que se refere ao desvio de tempo de resposta do tempo de resposta médio de todas as funções do sistema.

A ajuda ao usuário é outro parâmetro importante, uma vez que é muito comum a ocorrência de dúvidas durante a utilização de um sistema informatizado. Na maioria dos aplicativos, dois tipos de facilidades de ajuda podem ser definidas: a ajuda integrada, que é projetada juntamente com o software e que permite que o usuário obtenha respostas rapidamente sobre tópicos relativos à ação que ele está realizando, e a ajuda "add-on", a qual é incorporada após a instalação do software e que exige que o usuário percorra uma lista relativamente longa de tópicos para encontrar a resposta à sua dúvida, percebe-se que o primeiro tipo de ajuda torna a interface mais amigável.

O tratamento das mensagens de erro é um outro aspecto com o qual o projeto deve ater-se. A forma como as mensagens são apresentadas para o usuário deve ser cuidadosamente estudada, pois em muitos casos, as mensagens de erro indicam situações

(anormalidades) às quais os usuários não têm nenhum controle (por exemplo, memória insuficiente para executar uma dada função), em outros casos, elas sinalizam uma manipulação incorreta por parte do usuário, a qual pode ser corrigida (por exemplo, indicação de um parâmetro fora dos limites válidos), neste caso, a mensagem deve apresentar informações suficientemente completas e que permitam que o usuário recupere a origem do erro. No caso em que o erro provoque conseqüências danosas à execução de uma dada tarefa (por exemplo, mudança no conteúdo de um arquivo), estas deverão ser explicitadas na mensagem. Resumindo, a qualidade das mensagens favorece o aprendizado do sistema, indicando ao usuário a razão ou a natureza do erro cometido, o que ele fez de errado, o que ele deveria ter feito e o que ele deve fazer.

A nomenclatura dos rótulos, comandos ou menus e opções, correspondem a um outro aspecto que deve ser tratado com especial atenção pelo projeto, sendo que estes devem ser familiares à tarefa e ao usuário. Mesmo aplicativos com interfaces gráficas de última geração devem oferecer ao usuário a opção de uso do teclado para acessar todas (ou as mais importantes) funções disponíveis no aplicativo, através do uso de padronizações em relação aos rótulos ou atalhos de teclado. Um exemplo claro deste tipo de padronização está na maior parte das aplicações construídas para os ambientes gráficos, que utilizam o mesmo rótulo para ações compatíveis, como as ações de cópia e reprodução de blocos de texto ou gráficos (copiar/colar), as ações de gravação de arquivos (salvar/salvar como...) e o encerramento da utilização de um programa (sair).

Um outro aspecto presente em aplicações mais recentes é o conceito de macros, que podem ser definidas pelo usuário para armazenar as seqüências mais comuns de comandos usados no software, segundo este conceito, ao invés de digitar todos os comandos necessários à realização de uma dada tarefa, o usuário digita simplesmente o nome da macro.

O projeto de uma interface deve sempre basear-se primeiro nas técnicas básicas de manipulação de objetos (modelos de interação), que são a chave para desenvolver interfaces consistentes, para em um segundo estágio adicionar as técnicas especializadas como complemento aos métodos de realizar as operações. De modo a cumprir o objetivo

de gerar uma interface que permita ao usuário explorar completa e eficientemente as potencialidades do software, um conjunto de princípios deve ser observado. Não existe uma fórmula ótima para o desenvolvimento das interfaces, mas, em linhas gerais, pode-se classificar os princípios sob três diferentes pontos de vista: a interação, a exibição de informações e a entrada de dados.

3.1.1 A INTERAÇÃO

No que diz respeito à interação, pode-se relacionar os seguintes princípios:

- A coerência, na definição das escolhas dos títulos de menu, entradas de comandos, utilização de recursos visuais e auditivos nas respostas a comandos do usuário e outras funções importantes;
- A consistência, onde os aspectos da interface (aparência e comportamento) devem manter-se consistentes de um aplicativo para outro e principalmente de uma tela para outra;
- A simplicidade, a interação do usuário com a interface deve ocorrer de forma simples e intuitiva;
- Exigir confirmação de qualquer ação destrutiva não trivial (eliminação de arquivos, abandono do software, alterações substanciais de configuração), proteger eventualmente com senha ações que possam acarretar em conseqüências catastróficas para o sistema;
- Possibilitar a reversão ("undo") da maior parte das ações;
- Redução da quantidade de informações a ser memorizada no intervalo entre ações;
- Otimização de diálogo, definindo cuidadosamente o layout das telas e dos atalhos de teclado;
- Admissão de erros do usuário (atalhos de teclado incorretos, comandos e dados inadequados), evitando a ocorrência de funcionamento anormal (robustez);
- O feedback, diz respeito às respostas do sistema às ações do usuário;
- Categorização das atividades, organizando a tela segundo a classificação escolhida (operações de arquivos, operações de edição, ajuda, formatação, configurações);
- Oferecimento de facilidades de ajuda sensíveis ao contexto;

- As mensagens do sistema, devem ser tratadas e concebidas de maneira a atuarem como elementos de ajuda na interação homem-máquina;
- Nomeação de comandos através de verbos ou expressões verbais simples.

3.1.2 A EXIBIÇÃO DE INFORMAÇÕES

Com relação à exibição de informações, os seguintes aspectos devem ser observados:

- A concisão, ou seja, mostrar apenas informações relevantes ao contexto do sistema;
- Priorizar a utilização de símbolos gráficos na expressão de informações, sempre que possível;
- A limitação do uso da memória humana, reduzindo a probabilidade da ocorrência de erros e aumentando o rendimento durante a execução da tarefa;
- A carga cognitiva deve ser reduzida com o uso de modelos mentais apropriados (metáforas do mundo real) para a representação da informação;
- Precisão com relação às mensagens de erro, permitindo que o usuário decida e recupere o erro ou que ele possa evitar uma situação semelhante;
- Em informações textuais, o uso de letras maiúsculas e minúsculas facilita o entendimento;
- Uso de janelas para separar diferentes tipos de informação;
- A atenção do usuário, através da cautela na utilização de recursos e técnicas, consegue-se manter a legibilidade e a clareza da interface;
- Utilizar displays análogos para representar valores de grandezas reais de um dado sistema;
- O layout, através do uso eficiente do espaço da tela;

3.1.3 A ENTRADA DE DADOS

Durante a utilização de um software com alto nível de interatividade, o usuário passa boa parte do seu tempo fazendo a escolha de comandos e inserindo dados de processamento. De forma a melhorar as condições nas quais estas interações ocorrem, os seguintes princípios devem ser considerados:

- Minimização do número de ações necessárias à entrada de dados, reduzindo, principalmente a quantidade de texto a ser digitado. Recursos interessantes para esta redução são as listas, tabelas e dados pré-formatados;
- A coerência visual da interface em relação às entradas, mantendo cores, tamanhos e formas compatíveis com o restante da interface;
- Configuração da entrada por parte do usuário, permitindo minimizar as ações, em função das suas habilidades;
- Desativação de comandos inadequados ao contexto das ações realizadas num dado momento, o que pode evitar que o usuário gere erros de execução pela introdução de um comando inválido (por exemplo, invalidar o comando Reduzir Zoom, quando a redução chegou ao seu limite);
- Fornecimento de poder de interação ao usuário, permitindo que ele controle a navegação através do software, eliminando ações de comando e recuperando-se de erros sem a necessidade de abandonar o aplicativo;
- Minimizar o esforço do usuário para a entrada de dados (evitar que o usuário tenha de digitar unidades na entrada de grandezas físicas, utilizando uma unidade default modificável; eliminar a necessidade de digitação de ",00" quando os valores não contiverem componentes decimais).
- A intuitividade é a qualidade fundamental de uma interface, significando o quão fácil é de usar a interface e o projeto da interface deve basear-se tanto em psicologia cognitiva quanto em engenharia de computadores e projetos gráficos [Mazzola, 98].

Os computadores automatizam e revolucionam a grande maioria das atividades humanas, incorporando muito rapidamente novos conceitos e tecnologias aos métodos utilizados normalmente na realização das tarefas do homem. Em contraste, o ser humano não consegue acompanhar esta evolução no sentido de aprender e compreender as novidades tecnológicas que surgem quase que diariamente.

Grande parte dos usuários de sistemas computacionais não é especialista em informática e tem como expectativa básica que um aplicativo informatizado possua uma interface intuitiva, de fácil uso e que permita a rápida compreensão do seu funcionamento. Os usuários preferem sistemas de uso fácil, mesmo possuindo funcionalidade reduzida, ao

invés de sistemas muito ricos em termos de funcionalidades, mas de uso "complicado". Portanto, pessoas não especializadas em computação estão usando diretamente computadores, tornando a interface do usuário um dos fatores preponderantes para o sucesso ou não de um sistema computacional.

Assim, a interface, não se restringe apenas a um meio de apresentação dos sistemas interativos, para os usuários ela representa o aplicativo como um todo, englobando funções, procedimentos e elementos de apresentação. As técnicas e processos tradicionais da área de Engenharia de Software, infelizmente, não se mostram eficazes para o desenvolvimento de interfaces. Em função disto, modelos e técnicas específicas de análise, projeto e implementação de interfaces são continuamente propostos e o processo de geração da interface está sendo realizado à parte do processo de desenvolvimento do componente responsável pela funcionalidade do sistema à qual a interface dá acesso.

O número de variáveis envolvidas na construção de interfaces é muito grande e em decorrência disso, não existe um processo o suficientemente geral e abrangente, que possa ser considerado o melhor para a maioria dos casos. Existem, contudo, muitas propostas de métodos que foram aplicados em casos particulares e que podem ser utilizados e adaptados em situações similares.

Segundo Newman [Newman, 94], *a maioria dos artigos sobre interfaces homem computador aborda novas soluções de projeto.* Criar novas soluções não deve se constituir de atividades diárias, estas devem surgir a partir do amadurecimento dos métodos existentes, somados a experiência adquirida durante a aplicação destes métodos no desenvolvimento de interfaces.

A seguir são apresentadas técnicas, que podem ser incorporadas aos métodos tradicionais de desenvolvimento de sistemas, comentados no capítulo 1, e que através da análise ergonômica da tarefa, da integração dos fatores humanos na análise e projeto das interfaces, da descrição e análise da utilização da informação e a utilização de um método e uma ferramenta para a geração de interfaces do usuário através de modelos de dados baseados em regras ergonômicas, buscam localizar e analisar elementos informacionais

essenciais no projeto e desenvolvimento de interfaces. O resultado da utilização destas técnicas, pode ser o refinamento da especificação dos requisitos necessários para a implementação de interfaces com características de usabilidade.

3.2 ANÁLISE ERGONÔMICA DA TAREFA

3.2.1 ANÁLISE E MODELAGEM DA TAREFA

O processo de análise da tarefa é normalmente conduzido segundo uma abordagem de refinamentos sucessivos. O projetista identifica as principais tarefas executadas no contexto da aplicação, sendo que cada tarefa pode ser, eventualmente, refinada em duas ou mais sub-tarefas. A partir da identificação e refinamento das tarefas, estas são modeladas e o projeto da interface pode ser encaminhado, observando-se os seguintes passos: [Cybis, 97].

- estabelecimento dos objetivos de cada tarefa;
- definição da seqüência de ações necessárias para cada tarefa;
- especificar como as ações de cada tarefa estarão acessíveis a nível de interface;
- indicar o aspecto visual da interface para cada acesso a uma ação da seqüência especificada (estado do sistema);
- definir os mecanismos disponíveis para que o usuário possa alterar o estado do sistema;
- especificar o efeito de cada mecanismo de controle no estado do sistema;
- indicar o significado de cada estado do sistema (que informações deverão ser oferecidas pela interface em cada estado).

3.2.2 ANÁLISE ERGONÔMICA DO TRABALHO - AET

O objetivo da AET é de validar um sistema produtivo a partir de suas relações homem-máquina, caracterizadas pelas dimensões tarefa e atividade, lógica de utilização e lógica de funcionamento. A metodologia clássica da AET prevê três etapas de análises: a análise da demanda, a análise da tarefa e a análise da atividade.

3.2.2.1 *A ANÁLISE DA DEMANDA*

A análise da demanda tem como foco de estudo a própria análise que será realizada. Desta forma, em reuniões com os gerentes do trabalho e com os operadores do sistema, os projetistas devem discutir e apresentar o que se pretende com a análise ergonômica do trabalho. São previstos dois momentos:

- Apresentação aos parceiros: Na reunião de apresentação dos analistas aos envolvidos com o sistema são abordados os seguintes tópicos:
  - Identificação: Apresentar-se aos parceiros sociais, gerentes e usuários, esclarecendo sobre os objetivos, métodos e limites da intervenção proposta;
  - Objetivos, métodos e limites: Explicar o que se pretende fazer e como. Precisar os limites colocados pelo tempo de desenvolvimento. Esclarecer sobre os resultados esperados para este estudo;
  - Situar o trabalho: Saber como o sistema está situado em relação ao conjunto dos procedimentos definidos na unidade técnica e organizacional;
  - Fontes de informação: Conhecer as fontes de dados necessários para analisar a situação de trabalho.
  - Planejamento da análise: O planejamento da análise refere-se às definições quanto aos métodos de trabalho, cronograma aproximativo e envolve uma revisão bibliográfica sobre as questões pertinentes à análise.

3.2.2.2 *A ANÁLISE DA TAREFA*

A tarefa se refere àquilo que os operadores devem realizar durante a preparação, operação, e manutenção de um sistema. Ela possui duas perspectivas básicas: as dos gerentes e as dos operadores. Os primeiros tem uma visão dos objetivos e dos métodos que os operadores devem buscar e empregar em seu trabalho. Esta visão é geralmente formalizada pela empresa. Os operadores, por seu lado, possuem uma representação própria de como realizar as tarefas que lhes são solicitadas. Estas representações do trabalho é que são o alvo da análise da tarefa. Ela é realizada através de entrevistas com os gerentes do trabalho e com os usuários do sistema. O tema central das discussões é a tarefa

a ser analisada, e a abordagem se faz a partir das perspectivas que gerentes e usuários têm sobre o funcionamento e utilização do sistema.

- A visão dos gerentes: Em geral os gerentes não são usuários diretos do sistema. Em relação a esses, os gerentes possuem uma visão mais geral e mais ampla, incluindo dados sobre as interligações e os aspectos econômicos do sistema. Nesta etapa deve-se tentar obter elementos sobre:
- Representação do sistema: objetivos, procedimentos, regras de utilização, restrições;
- Dados organizacionais: organização do trabalho, ligações entre os serviços, quem opera o sistema, quais os turnos de trabalho;
- Dados econômicos: custo do sistema, vantagens proporcionadas (onde e quanto, diretas/indiretas, prejuízos).

- Posto de trabalho: No caso do trabalho informatizado, a análise do posto de trabalho refere-se à:
- Ambiente: layout geral, iluminação e ruído;
- Ferramentas de apoio: estado da tecnologia, repartição de funções homem-máquina;
- Fluxo de informações: documentos oficiais, com as informações de entrada e de saída: trânsito (quem emite e quem recebe), estrutura dos documentos, volume e modo de utilização.

- Reconhecendo a tarefa: Nesta etapa, parte-se para obter uma descrição sobre a tarefa a partir dos pontos de vistas de usuários e gerentes do sistema. Com uns e outros, são realizadas entrevistas dirigidas buscando evidenciar as características do processo de realização da tarefa. O reconhecimento do processo da tarefa é particularmente importante quando o objetivo da análise é a concepção de um novo sistema ou de um sistema informatizado que venha apoiar o sistema atual. Assim, deve-se tentar obter uma descrição detalhando os seguintes elementos:

- Objetivo último a alcançar (o que o operador deve realizar);
- Decomposição em sub-tarefas/sub-objetivos;
- Relações entre as sub-tarefas (seqüenciais, paralelas, alternativas, facultativas);
- Nomes, denominações e definições das sub-tarefas;

- Objetivos a alcançar nas sub-tarefas;
- Métodos ou a seqüência de ações que o operador utiliza em cada sub-tarefa para alcançar seus objetivos (como o operador realiza sua tarefa);
- Estados inicial e final do sistema para cada sub-tarefa (quais informações são utilizadas e quais as informações que são produzidas em cada etapa);
- Pré e pós-condições das sub-tarefas, que se referem a atributos dos elementos pertencentes aos estados inicial e final do sistema, que devem ser satisfeitos para autorizar o início de determinada ação.

De posse destes elementos pode-se realizar uma descrição hierarquizada do processo da tarefa. Essa descrição permite que se tenha uma compreensão geral sobre tarefas que se realizam de forma simultânea, paralela, seqüencial, alternativa ou opcional. Os elementos obtidos serão aplicados no processo de concepção da interface homem-computador do futuro sistema. Essa visão geral, é complementada por um levantamento em termos das informações envolvidas com as tarefas e sub-tarefas. Desta forma deve-se conhecer quais são:

- As informações necessárias e a ordem em que tornam-se disponíveis;
- As informações que são difíceis de obter;
- As informações que são inúteis (não são utilizadas);
- As informações que são impertinentes (que atrapalham e induzem a erros de interpretação).

Também deve-se obter informação complementar sobre as tarefas e sub-tarefas;

- Quais são as mais freqüentes;
- Quais são os horários (duração) e modo de utilização;
- Quais as deficiências, problemas e incidentes encontrados. Em particular deve-se identificar suas causas, condições de aparecimento, freqüência e os procedimentos para a recuperação.

3.2.2.3 *A ANÁLISE DA ATIVIDADE*

O próximo passo da análise refere-se à validação das descrições e informações que foram coletadas e que compõem as representações sobre o trabalho. Para tanto esta etapa

deve prever a realização de observações "in-loco" do trabalho do usuário. É especialmente recomendado que seja solicitado aos operadores que verbalizem sobre quais os objetivos, critérios, diagnósticos e razões das decisões tomadas. Pode-se prever meios de registro em vídeo, áudio e um bloco de anotações para capturar os aspectos citados em três tipos de situações; situação de normalidade, situações consideradas críticas e situações de erros e incidentes.

- Situações de Normalidade: A análise de atividades normais é feita através de observações contínuas procurando abranger toda a duração do trabalho. Em especial deve-se confrontar a descrição da tarefa realizada na etapa anterior de análise com o que é realmente realizado pelo usuário em termos de objetivo último a alcançar, decomposição da atividade em sub-atividades/sub-objetivos, relações entre as sub-atividades (sequenciais, paralelas, alternativas, facultativas). Deve-se verificar em particular, os graus de dificuldades na realização das atividades.

- Situações Críticas: A partir do que foi levantado na etapa de análise da tarefa, algumas situações podem ser consideradas problemáticas ou críticas e devem ser observadas com maior atenção. A observação destas situações se faz então por amostragem de períodos escolhidos. Nelas deve-se confrontar a descrição da tarefa prescrita com a atividade realizada pelo usuário em termos de métodos empregados e do fluxo de informação nesta parte do sistema.

- Erros e Incidentes: As situações de erros e incidentes são de difícil observação, tanto pela dificuldade de prever sua ocorrência como pela dificuldade de seguir seu processo. Recomenda-se então que sejam preparadas simulações, in-loco (no local de trabalho) ou em laboratório. Dependendo do sistema, as do primeiro tipo podem ser dispendiosas para empresa, operador e analista. Portanto sua realização deve ser alvo de uma estudo custo X benefício cuidadoso. Um outro tipo de simulação poder ser realizado em um cenário fora da situação de trabalho. Uma situação hipotética deve ser apresentada verbalmente ao usuário que deverá descrever os procedimentos para a recuperação da situação de normalidade.

3.3 MUSE – MÉTODO PARA ENGENHARIA DA USABILIDADE [Lim, 96]

O MUSE é um método de análise e projeto desenvolvido para ser usado por engenheiros de fatores humanos na criação de interfaces do usuário. O método visa melhorar a prática das interações humano-computador nas interfaces do usuário, através do fornecimento de suporte para a integração de fatores humanos com métodos estruturados de engenharia de software. O resultado do método MUSE é a especificação da interface, que é incorporada à especificação do produto produzido pelo método de Engenharia de Software apoiado pelo MUSE.

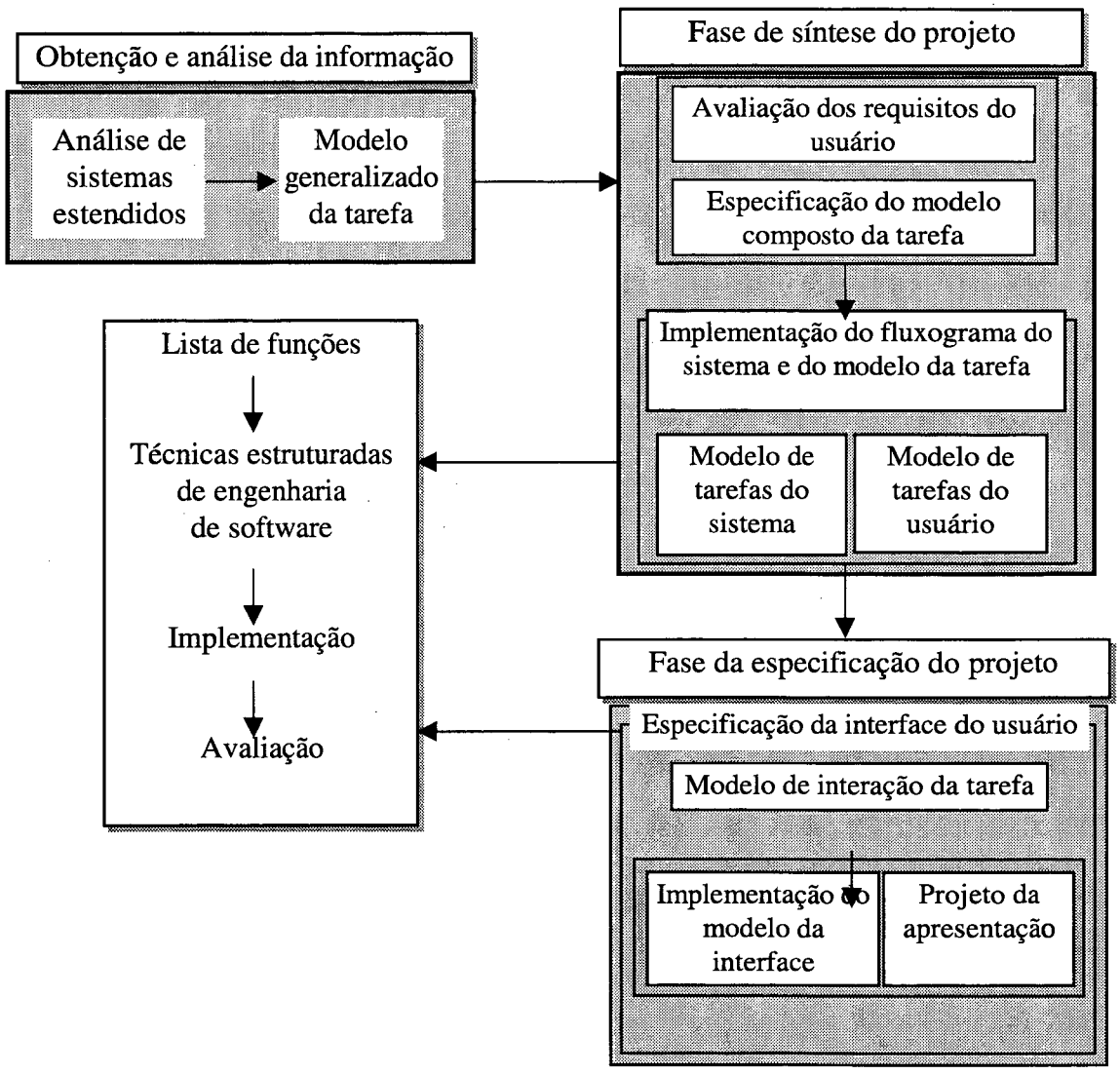

Figura 3.1 : Representação esquemática do método MUSE [Lim, 96].

O método MUSE apoia o projeto de uma maneira "top-down", especificando os detalhes da interface, baseando-se na informação oriunda da especificação das características gerais da tarefa, da análise dos requisitos e da análise de sistemas estendidos. A aplicação do método MUSE é um processo iterativo, e a figura 3.1 mostra um diagrama esquemático do método apoiando um método de engenharia de software não especificado.

O MUSE é composto de três fases: a fase de obtenção e análise da informação, a fase de síntese do projeto e a fase de especificação do projeto. A fase de obtenção e análise da informação apoia inicialmente a avaliação e o reuso de componentes de sistemas estendidos e a manutenção da consistência do projeto em relação aos requisitos do usuário. A fase de síntese do projeto apoia inicialmente a parte conceitual do produto de interação e a manutenção da consistência do projeto em relação à semântica do domínio da tarefa e a interpretação dos fatores humanos identificados nos requisitos do usuário dos sistemas estendidos analisados. A fase de especificação do projeto apoia inicialmente o projeto detalhado da interface. A checagem obrigatória e o intercâmbio de informação com o método de engenharia de software ocorre para assegurar que a interface é implementável e o projeto como um todo é consistente.

3.3.1 AS CARACTERÍSTICAS DA APLICAÇÃO DO MUSE

O método MUSE, visando melhorar a habilidade de profissionais em Interfaces Humano-Computador (IHC) no projeto de produtos que observam todos os requisitos do usuário, está preocupado em assegurar que:

- O produto seja considerado completo e apropriado em todos os níveis do projeto, do conceitual ao detalhado;
- O produto esteja consistente durante todos os níveis de projeto (incluindo a obtenção dos requisitos do usuário);
- O conhecimento do domínio é avaliado e aplicado no produto em momentos apropriados do projeto;

- O conhecimento dos fatores humanos (assim como do método de engenharia de software escolhido) é avaliado e pode ser aplicado no produto em momentos apropriados do projeto;
- Sistemas estendidos de diferentes qualidades são avaliados e podem ser integrados ao produto em momentos apropriados do projeto;
- O método MUSE atinge estas "preocupações", em parte, por estipular anteriormente o que será gerado em cada fase do projeto.

3.3.2 FASE DA OBTENÇÃO E ANÁLISE DA INFORMAÇÃO

Nesta fase, uma lista de todos os sistemas estendidos que devem ser analisados e avaliados é construída, ela contém os sistemas estendidos (existentes), sistemas com tarefas semelhantes e sistemas com finalidades relacionadas. Esta análise promove um melhor entendimento dos requisitos do usuário e, técnicas de fatores humanos são usadas para analisar os sistemas estendidos de duas maneiras: uma análise da tarefa realizada, por exemplo, com base em respostas de entrevistas com os usuários e outra através de documentos gerados pelos usuários durante a realização habitual de suas tarefas. Estas análises são expressas na forma de "diagramas estruturados" que são acompanhados por tabelas com a descrição das tarefas. Estas tabelas demonstram as implicações e especulações concernentes ao projeto. Em [Stork, 94], é encontrado um estudo de caso completo descrevendo a utilização do MUSE.

Com estas informações é desenvolvido o projeto conceitual à nível de tarefa, baseado nos requisitos do usuário e nas implicações e especulações produzidas na análise dos sistemas estendidos. Este projeto, chamado de "Modelo Generalizado da Tarefa do Sistema Estendido" é documentado também na forma de diagramas estruturados e tabelas. Segundo Stork [Stork, 94], "*os diagramas demonstram que o projeto é consistente com os requisitos do usuário e isto surge do entendimento adquirido durante a análise dos sistemas estendidos*". As tabelas identificam os fatores humanos e apresentam a generalização de atividades que ocorrem durante a realização das tarefas. A análise das tarefas, registradas no modelo generalizado da tarefa, gera o "Modelo da Tarefa do Sistema", também na forma de diagramas estruturados e tabelas de documentação.

Análises adicionais do MUSE desenvolvem os seguintes produtos: a semântica do domínio do sistema estendido é capturada através de entrevistas e observações e representada por diagramas de rede e tabelas chamadas "Domínio do Projeto" onde ocorrem discussões sobre o sistema estendido, em seguida, é realizada a avaliação do impacto do domínio sobre o conteúdo, o formato e o modo de apresentação do projeto estendido nas tarefas off-line (aquelas que não são apoiadas pelo sistema estendido). São analisados também os relacionamentos entre os comportamentos específicos do usuário das tarefas off-line e as tarefas realizadas pelo sistema. O comportamento da interface e as tarefas do sistema são mostradas em diagramas estruturados e em tabelas chamadas "Modelo de Interação do Sistema Estendido". O layout e a aparência do sistema estendido é apresentado em diagramas chamados "Layout Ilustrado da Interface do Sistema Estendido".

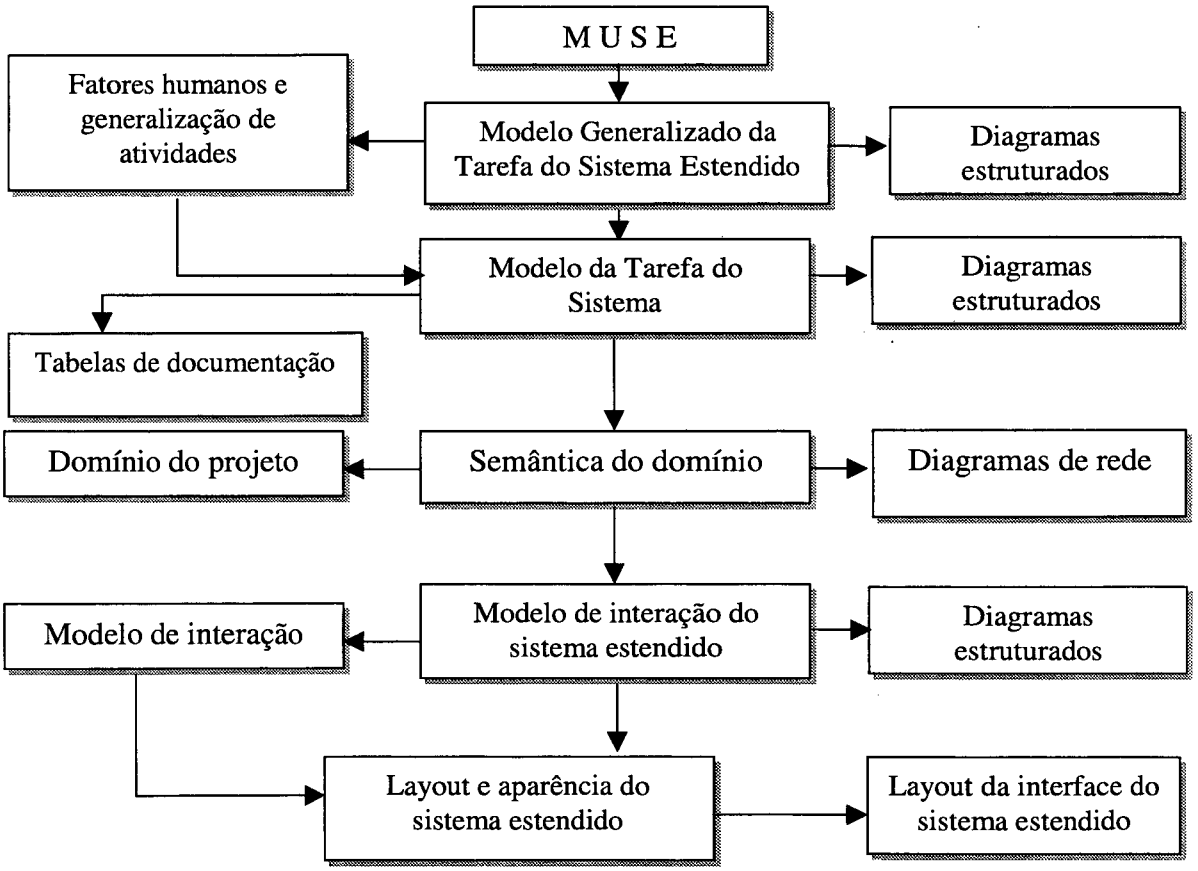

Figura 3.2: Esquema gráfico da fase da obtenção e análise da informação.

3.3.3 FASE DE SÍNTESE DO PROJETO

De maneira a assegurar a consistência com os requisitos do usuário e as primeiras avaliações dos fatores humanos, um resumo textual é criado, baseado nestes requisitos e na análise do sistema estendido. O documento é chamado "Declaração das Necessidades do Usuário", onde é declarado qualquer critério explicito de projeto, como por exemplo o custo, qualquer critério implícito de projeto, como o grau de funcionalidade do sistema estendido. É declarado também, qualquer critério explícito e/ou implícito de performance do sistema, denotados pelas restrições e, por último, é declarado qualquer fator humano conhecido e que são pertinentes ao sistema, tais como os defendidos em [Shneiderman, 92], [Bastien & Scapin, 93]. A presença destas guidelines confirmam o projeto de alto nível no Modelo Generalizado da Tarefa do sistema estendido.

O projeto conceitual criado da conjunção entre os requisitos do usuário e as tarefas do computador é incrementado nesta fase, originando o documento chamado "Modelo Composto da Tarefa ", também representado por diagramas estruturados e tabelas, onde importantes decisões de projeto são racionalizadas, tais como, desenvolver características no sistema que sejam análogos às tarefas realizadas atualmente. As tarefas on-line (aquelas apoiadas pelo sistema) são identificadas. Decomposições adicionais das tarefas off-line asseguram que o sistema possuirá conteúdo, formato e modo de apresentação conforme especificado na análise do sistema estendido realizada anteriormente. São desenvolvidos diagramas e tabelas para as tarefas on-line, tarefas que são implementadas sempre considerando-se os fatores humanos pertinentes.

3.3.4 FASE DA ESPECIFICAÇÃO DO PROJETO

A última fase do método MUSE é a especificação e projeto da interface do usuário, criando um documento chamado "Modelo da Interface", que baseado na especificação das tarefas on-line, produz uma declaração detalhada e específica da interface do usuário. A interface do usuário é gerada e refinada iterativamente. Uma vez que a mesma seja considerada satisfatória, todos os produtos resultantes desta fase são liberados para a equipe da engenharia de software implementar. Na fase de síntese do projeto, a tabela de

tarefas on-line, as quais implementam as funções do modelo composto da tarefa, é decomposta dentro do modelo de interação da tarefa, e estas são decompostas futuramente dentro do modelo da interface e do projeto da apresentação.

Stork [Stork, 94], que aplicou o método em estudos de caso, assegura que todos os produtos necessários ao sistema foram gerados conforme os requisitos e que o produto final foi considerado completamente e apropriadamente por todos os níveis do projeto; a consistência foi mantida durante todos os níveis do projeto; o conhecimento do domínio foi avaliado e aplicado para o produto final nos níveis apropriados do projeto; os fatores humanos conhecidos foram avaliados e aplicados para o produto final nos níveis apropriados do projeto; qualidades desejáveis dos sistemas foram avaliados e integrados, quando apropriados, dentro do projeto. Deve ficar claro, entretanto, que o método MUSE não serve para desenvolver o projeto, ele apresenta razões básicas para a tomada de decisões sobre fatores humanos, e providencia um protótipo para as primeiras iterações, não sendo ainda a solução perfeita.

3.4 O MÉTODO AIU - ANÁLISE DA UTILIZAÇÃO DA INFORMAÇÃO [Gulliksen, et. all., 97]

O método AIU visa complementar os métodos usados para projeto da interface do usuário, através da identificação de requisitos adicionais para a interação humano-computador. O método focaliza o trabalho apoiado pelo computador, relacionado à carga cognitiva, aspectos que os usuários finais frequentemente não estão cientes. Trata-se de um método para descrever e analisar como as entidades de informação identificadas na análise da informação são utilizadas nas situações de trabalho. O método baseia-se em entrevistas e observações, onde o especialista analisa a rotina de manipulação física da informação e solicita de um conjunto representativo de usuários finais, informações sobre as características (requisitos) de suas tarefas, as decisões necessárias, prioridades e ações. O método apoia o projetista de interfaces com requisitos estruturados para a interação humano-computador dentro do contexto da tarefa e é integrado com o modelo de desenvolvimento centrado no usuário.

O projeto cooperativo de aplicativos informatizados é defendido em [Bjerknes, Ehn & King, apud Gulliksen, 97], [Damodaran apud Gulliksen, 97] e aplicado visando ampliar o direito dos usuários finais de cooperar e influenciar no projeto e desenvolvimento de sua futura situação de trabalho. Apesar de os usuários finais muitas vezes não estarem cientes do comportamento de sua tarefa e não possuírem especialização em projeto de interfaces humano-computador, a sua participação deve tornar-se mais eficiente durante a modelagem do domínio da tarefa, no estabelecimento das metas da tarefa e o esclarecimento de como a tarefa é executada atualmente. O projeto da interface do usuário deve ser realizada por projetistas habilidosos com a cooperação dos usuários finais nas fases certas, após um profundo conhecimento e entendimento a cerca do domínio da tarefa.

3.4.1 A NECESSIDADE DE FOCAR A UTILIZAÇÃO DA INFORMAÇÃO

O domínio da tarefa e o entendimento de como e quando a informação é utilizada devem ser completamente definidos antes que o projeto e o desenvolvimento da interface se iniciem. Portanto, não somente quais dados são usados na interface do usuário, mas também como estes dados são usados, são essenciais quando se projetar uma interface do usuário para uma tarefa específica. Existem duas principais razões para isto:

a) A primeira razão é resumida em alguns princípios:

- O princípio da manipulação, onde o indivíduo que realiza a tarefa, toma decisões e executa julgamentos;
- A tomada de decisões e julgamentos são processos cognitivos que requerem atenção e criatividade;
- A capacidade cognitiva humana é limitada diante destes processos;
- As atividades humanas variam de acordo com a quantidade de consciência cognitiva alocada para cada tarefa;
- Analisando como a informação está sendo utilizada na realização da tarefa, é possível criar interfaces que podem ser manipuladas com um mínimo de capacidade cognitiva. Isto permite que maior atenção seja dispensada para os julgamentos e tomada de decisões inerentes à tarefa.

b) A segunda razão é que humanos por si mesmos, criam e padronizam ferramentas especiais e métodos que são adaptados para a realização da tarefa (tabelas, formulários, cartões). É importante entender estas adaptações e como elas trabalham, de maneira a estar hábil à integrá-las ao novo sistema.

3.4.2 CARGA COGNITIVA DE TRABALHO E A HABILIDADE DO USUÁRIO

Pesquisas realizadas por Gulliksen [Gulliksen, 96], demonstram que durante a interação com alguns sistemas informatizados, 80% do tempo de trabalho é gasto manipulando a interface. O processo de execução da tarefa é constantemente interrompido pela necessidade de rearranjo da interface, abrindo, minimizando e movendo janelas, inicializando diferentes aplicativos e localizando e interpretando informações. O resultado é a baixa eficiência na realização da tarefa e a baixa aceitação do sistema por parte do usuário. O computador é uma ferramenta que é usada e apreciada somente quando promove a eficiência no escopo da tarefa. Então, a interface deve ser projetada sobre as bases da otimização das atividades de trabalho, justificando a utilização do computador. A informação necessária para a interação usuário-computador-tarefa deve ser apresentada de uma maneira percebível ao invés de legível (informação que não necessita decodificação), refletindo os objetivos e resultados da análise e minimizando a carga cognitiva do usuário durante a execução da tarefa.

O maior parte das ações realizadas durante a execução de uma tarefa, podem ser descritas como julgamento da informação e tomada de decisões sobre o que fazer no próximo passo da interação. A performance de uma tarefa e o controle da interface do usuário pode ser considerado como duas tarefas concorrentes competindo pelos recursos cognitivos do usuário. O principal objetivo quando construir uma interface do usuário deve ser de projetar visando o julgamento e a tomada de decisões, situações onde a interface deve assegurar que valores mínimos de processo cognitivo (interpretação da informação) e ações físicas (manipulação da interface) serão necessários para a conclusão da tarefa.

3.4.3 O MÉTODO AIU

O método AIU descreve a tarefa de julgar e decidir, ações que são parte de um certo tipo de tarefa e como os dados são usados para resolver estas tarefas. Esta abordagem pretende envolver representantes do domínio da tarefa com especialistas em projeto de interfaces, onde a análise da tarefa e a formulação da descrição dos resultados sejam realizados em conjunto. Uma AIU é realizada com um ou alguns poucos representantes da tarefa, realizando efetivamente seu trabalho, sendo este (s) observado (s) por analistas que adquirem algum entendimento do propósito e do conteúdo da tarefa. Desta maneira, o analista adquire experiência e conhecimento que podem ser úteis quando este projetar a interface do usuário. A descrição do AIU deve ser baseada sobre o que é realizado, como é realizado e como os resultados são documentados.

3.4.4 PRÉ-REQUISITOS PARA O MÉTODO AIU

A base para a utilização do método AIU é a realização de uma típica análise da tarefa, onde as atividades da tarefa são descritas e analisadas com respeito ao seu conteúdo, a organização da tarefa é apresentada e os problemas existentes são declarados. A AIU não tem por objetivo descrever as regras de decisão que são usadas em julgamentos e tomadas de decisão, mas de como a informação é apresentada, usada, manipulada e organizada concretamente pelos usuários em cada tarefa. Em um ambiente computadorizado, isto será a descrição de, por exemplo, como o usuário que move-se entre telas está hábil a fazer julgamentos, o que causa interrupções na linha de pensamento, quais informações são de origem não informatizada e são usadas simultaneamente com o material informatizado, e como tudo isto é utilizado.

Um modelo (modelo de dados orientado a objetos ou um modelo funcional) pode ser desenvolvido, representando os dados com seus respectivos comportamentos e relacionamentos. O modelo de dados coloca limites no espaço do projeto e se a AIU necessitar modificações, o projeto da interface do usuário pode ser alterado, assumindo que o modelo de dados foi especificado de uma maneira modificável.

3.4.5 METODOLOGIA DO MÉTODO AIU

Durante a definição do domínio da tarefa é realizado uma detalhada análise das informações que são manipuladas no decorrer da tarefa, na tentativa de reduzir a carga cognitiva definida em relação aos indivíduos e seus comportamentos. Esta análise é realizada seguindo os seguintes passos:

- Primeiro, selecionar representantes que são habilidosos na realização da tarefa e para os quais o novo sistema informatizado está sendo desenvolvido;
- Realizar uma descrição do conteúdo da tarefa visto da perspectiva do usuário;
- Descrever e coletar cópias de todos os tipos de informações que são movimentadas diariamente, por exemplo, formulários, telas impressa ou anotações;
- Descrever todas as rotinas existentes para a manipulação da informação atualmente, por exemplo, arquivos portáteis, caixas de correio, pastas;
- Descrever o julgamento e as rotinas de tomada de decisões que são envolvidas na tarefa. As tarefas que o usuário realiza enquanto trabalha devem ser distinguidas e categorizadas;
- Descrever o conjunto de informações (variáveis) que são usadas em cada tarefa e como esta informação se relaciona no modelo de dados. Estas variáveis devem ser simultaneamente apresentadas pela interface quando determinada tarefa estiver sendo realizada;
- Descrever como cada variável é atualmente usada para cada tarefa;
- Analisar o material de cada tarefa em termos de:
  - Demanda sobre a simultaneidade dos dados;
  - Faixa de valores e características de cada variável;
  - O que deve ser realizado, por exemplo, como as decisões são documentadas. Esta informação ajuda o projetista a escolher conteiners representativos para as variáveis;
- Analisar o material em termos das tarefas que devem ser realizadas simultaneamente. Isto define a situação do trabalho.

A AIU deve ser suficientemente completa para cobrir todas as atividades relevantes no domínio, visando descrever qual informação está sendo usada, por quem e quando. Esta análise forma as bases para as decisões de projeto da interface do usuário. Baseado nesta análise, um conjunto de componentes de interface da possível aplicação são definidos e

projetados. O projeto é feito para que os componentes possam ser identificados e usados com um mínimo de carga cognitiva. A especificação dos componentes básicos e mais complexos da interface é um procedimento que se desenvolve passo a passo. Para construir o projeto, o modelo da tarefa deve incluir:

- Uma lista de tarefas realizadas em relação às decisões tomadas e quais tarefas podem ser realizadas simultaneamente;
- Uma lista das variáveis, em relação ao modelo de dados, que podem possivelmente ser usadas na realização de cada tarefa, suas prioridades e características;
- Uma lista de ações necessárias para manifestar cada decisão;
- Uma lista de situações de trabalho que ocorrem naturalmente e são relacionadas com diferentes usuários, definindo conjuntos de tarefas que são usualmente ou possivelmente realizadas concorrentemente;
- Um cenário descrevendo uma dia típico de trabalho. Este modelo de tarefa pode então ser usado pelo projetista na criação de objetos de interação adequados para a interface.

3.4.6 COMO REALIZAR UMA AIU

As palavras-chaves para a realização de uma AIU são a humildade e o respeito. O analista deve ser humilde para perguntar tudo o que for necessário a respeito da tarefa e deve respeitar a opinião da (s) pessoa (s) responsável (eis) pela tarefa. Isto significa, entre outras coisas, que para obter resultados satisfatórios em uma AIU, deve existir a cooperação entre analista e profissionais. O analista deve escutar, tentar entender e documentar cada fato que acontece durante uma sessão de trabalho. Os profissionais analisados devem realizar suas tarefas de rotina e através do diálogo descrever exceções e casos especiais que acontecem durante o desenrolar da tarefa. Estas características devem ser tratadas e avaliadas, pois a falta de flexibilidade encontrada em interfaces do usuário e de sistemas informatizados, são consequência do pouco valor dado a estes detalhes.

Outro aspecto da AIU é o chamado "Conhecimento Tácito" ou a automatização do comportamento, onde certas etapas da tarefa são automaticamente realizadas e não são transmitidas pelo profissional devido a sua estrutura complexa. A AIU não captura estes

processos cognitivos automatizados, apenas supõe a sua existência. Não existe um método ou técnica especializada para detectar estas situações. Experiências mostram que através da observação do usuário realizando suas tarefas é possível perceber o momento em que a tarefa atinge determinados estágios ou etapas são completamente concluídas. O início de cada tarefa ou sub-tarefa é então documentada, estabelecendo através de entrevistas os requisitos necessários para a continuidade da tarefa, relacionados com os resultados obtidos anteriormente.

3.4.7 DESCREVENDO AS VARIÁVEIS QUE SÃO USADAS

Através da descrição das ferramentas que os profissionais têm à sua disposição e discutindo as tarefas relacionadas à elas, é possível decidir sobre o número de entidades de informação (variáveis) que são essenciais para cada decisão. Os usuários tendem a descrever as propriedades gerais de suas tarefas, deixando de fora as exceções. Estas variáveis são nomeadas e identificadas baseando-se como elas são utilizadas ou por que são chamadas no contexto da tarefa. Na descrição dos resultados da análise, estas variáveis devem ser relacionadas no modelo de dados para serem organizados.

Variáveis importantes que são usadas, talvez sem o conhecimento do usuário devem ser documentadas durante a análise, pois no contexto da tarefa informatizada elas podem representar um campo de entrada de dados, um rótulo ou formulário. Categorizar estas variáveis demanda muita atenção do analista, e se estas variáveis forem negligenciadas o projeto corre o risco de ter a eficiência decrementada e agravando a performance do usuário. As variáveis devem ser listadas e uma descrição de como elas são usadas durante o processo de tomada de decisão e julgamento deve ser incluída. Cada variável identificada deve ser classificada considerando-se características que estas possam ter. O seguinte esquema pode ser usado para a classificação das variáveis [Foley apud Gulliksen, 97]:

- Variáveis contínuas;
- Variáveis discretas com um ordenado conjunto de valores (escala ordenada);

- Variáveis discretas com um desordenado conjunto de valores (escala nominal). Esta categoria deve ser distinta baseado na mutualidade exclusiva ou não dos valores.

O conjunto de valores possíveis de serem assumidos pela variável devem também ser descritos, bem como características que não são capturadas através do modelo de dados, e representam características que são específicas a um grupo de usuários ou tarefas. Estes dados são utilizados para definir a forma de apresentação e o refinamento do modelo de dados. A frequência de utilização das variáveis pode ser especificada por entrevista e observação. Um bom método é observar e fazer perguntas enquanto (ou após) a realização da tarefa.

3.4.8 ANÁLISE DO MATERIAL UTILIZADO EM CADA TAREFA

De maneira a agrupar toda a informação, a descrição deve ser realizada sob três diferentes óticas:

- Demanda sobre a apresentação simultânea de dados: a apresentação simultânea dos dados necessários para a tomada de decisões e julgamentos é de grande ajuda para o usuário;

- Baseado na informação coletada para cada situação, usuário e analista classificam cada variável em dois níveis: O nível 1 contém todas as variáveis que são usadas frequentemente na tomada de decisões e julgamentos, no nível 2 encontram-se as variáveis que são usadas com menos frequência. Estes níveis contém uma descrição resumida da demanda e simultaneidade da utilização;

- Setagem de valores e características das variáveis: para que a descrição realizada seja útil para o desenvolvimento de interfaces do usuário, cada variável deve ser relacionada no modelo de dados, isto é, a variável necessita ser descrita em termos da informação que será carregada no sistema. Os valores para cada variável devem ser discutidos junto com os analistas, para esclarecer quais valores são importantes mostrar e codificar;

- O que deve ser feito para cada situação de julgamento e tomada de decisão: as possíveis ações que são parte das situações de tomada de decisões necessitam ser supridas com uma aparência adequada (p.ex. botão, seqüência de teclas, menu) no processo de projeto da interface do usuário. Por isto o projetista necessita de uma lista com todas as ações que são necessárias para cada situação de tomada de decisão.

Frequentemente existem tarefas que logicamente não são relacionadas mas devem ser realizadas simultaneamente devido a condições externas, tais como estrutura organizacional, horário de telefone e outros. Tais condições são fáceis de especificar e de grande importância para o projeto da interface do usuário. Muito do que foi descrito anteriormente é válido quando o projeto em desenvolvimento é baseado em uma situação de trabalho existente e que será informatizada. Quando a AIU for realizada visando a criação de um sistema totalmente novo, a AIU deve ser realizada somente naquelas tarefas possíveis de serem visualizadas ou em sistemas semelhantes que apresentem características das futuras tarefas do novo sistema. Outros aspectos das novas situações de trabalho podem ser especificadas através de sessões de modelagem centrada no usuário. Aqui a cooperação com o usuário é essencial. O uso de diferentes técnicas baseadas em cenários pode ser a solução para a falta de situações de trabalho. Cenários detalhados contendo as novas situações de trabalho podem ser usadas como base para usuários experientes definirem como as rotinas de manipulação da informação devem ser realizadas sob as novas circunstancias.

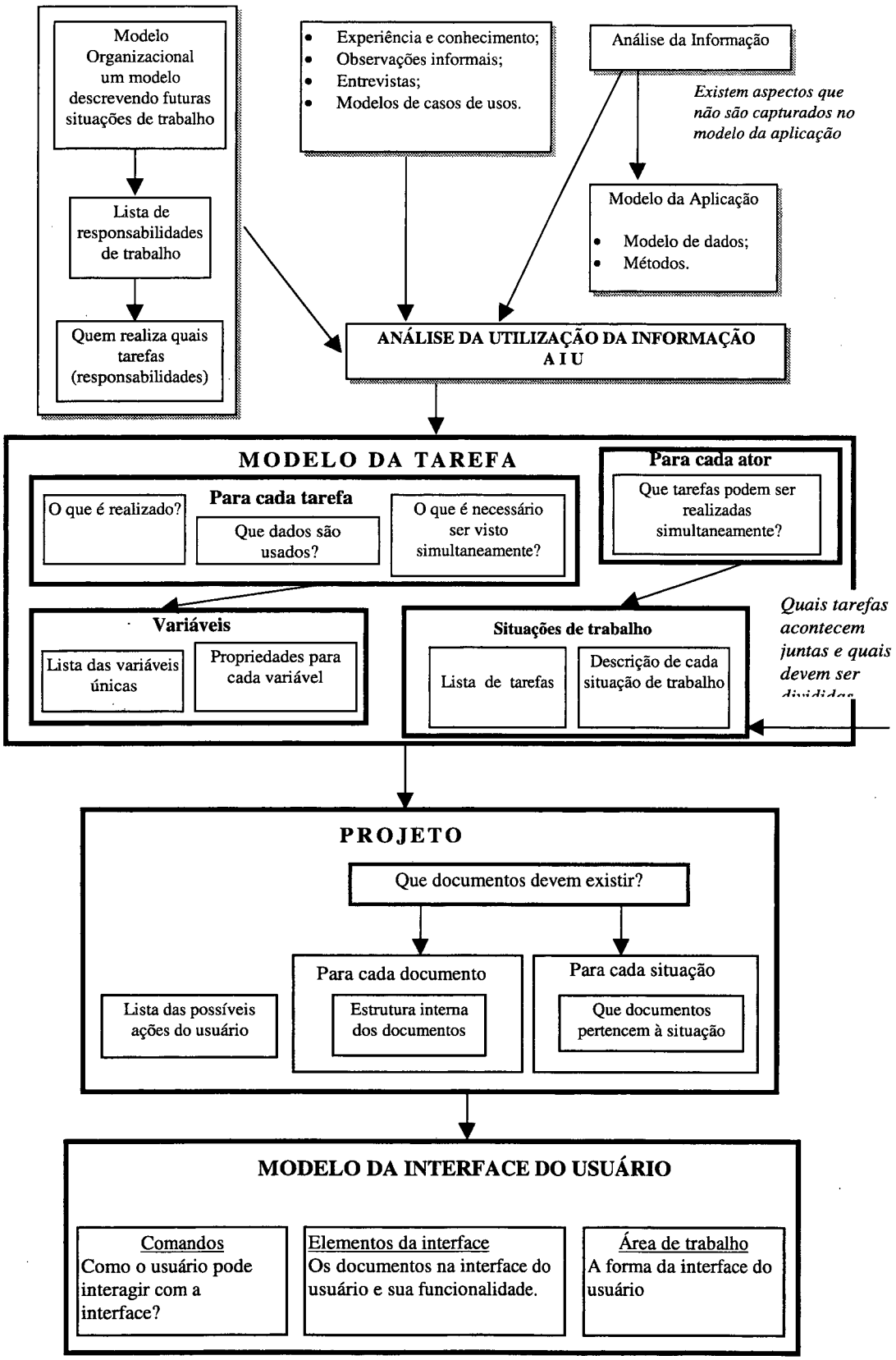

Figura 3.3 : Fases que compõem o método AIU. [Gulliksen, et. all., 97]

3.5 GENIUS - GERADOR AUTOMÁTICO DE INTERFACES USANDO REGRAS DE SOFTWARE ERGONÔMICO [Bellinger, et. all., 96]

Corresponde a um método e uma ferramenta para a geração de interfaces do usuário através de modelos de dados baseados em regras ergonômicas. A representação da interface do usuário é baseada em visões definidas pelo modelo de dados e a estrutura básica do diálogo é derivada da estrutura básica do modelo de dados. Isto assegura o desenvolvimento de interfaces apropriadas para a tarefa, através da transferência das características do domínio da aplicação e da tarefa do usuário refletidas no modelo de dados para a estrutura do diálogo. As regras ergonômicas utilizadas no processo de desenvolvimento garantem o uso consistente dos objetos de interação e uma estrutura de diálogo uniforme. O uso do modelo de dados como ponto inicial para a geração das interfaces, assegura a integração da engenharia de software e do projeto da interface, reduzindo os esforços de desenvolvimento e aumentando a qualidade da interface produzida.

3.5.1 O PROCESSO DE DESENVOLVIMENTO DE INTERFACE COM O AMBIENTE GENIUS

Em um primeiro passo, os dados e funções necessárias para a tarefa são especificadas. Esta informação é usada para a derivação da estrutura de diálogo e serve como base, no segundo passo, para a geração da interface do usuário com princípios ergonômicos. Esta descrição é transformada em especificações para serem utilizadas por algum UIMS (Sistemas de gerenciamento de interfaces do usuário), responsável pela implementação da interface. O uso deste modelo é ancorado nas seguintes vantagens em relação às notações comumente usadas de engenharia de software:

- É evitado a especificação redundante dos dados e é atingida e mantida a consistência entre o modelo de dados da aplicação e a interface desenvolvida;
- A interface permite a navegação completa pelos dados manipulados;
- O modelo de dados pode ser validado pelo usuário nos primeiros estágios do projeto, através da manipulação do protótipo desenvolvido;

- O mapeamento consistente dos elementos de dados e das funções dos objetos de interação em uma simples aplicação e através de diferentes aplicações é atingida pelo uso de regras de projeto;
- O projeto baseado em regras assegura a consideração de fatores humanos e apoia a maioria dos princípios dos guias de estilos existentes.

O ambiente da ferramenta GENIUS é ilustrado na figura a seguir. As visões são definidas sobre o modelo de dados. Um componente baseado em conhecimento gera a apresentação da interface do usuário e a estrutura de diálogo através da definição das visões. Para desenvolver o sistema um UIMS é utilizado. [Adaptado de Bullinger, et. all., 96].

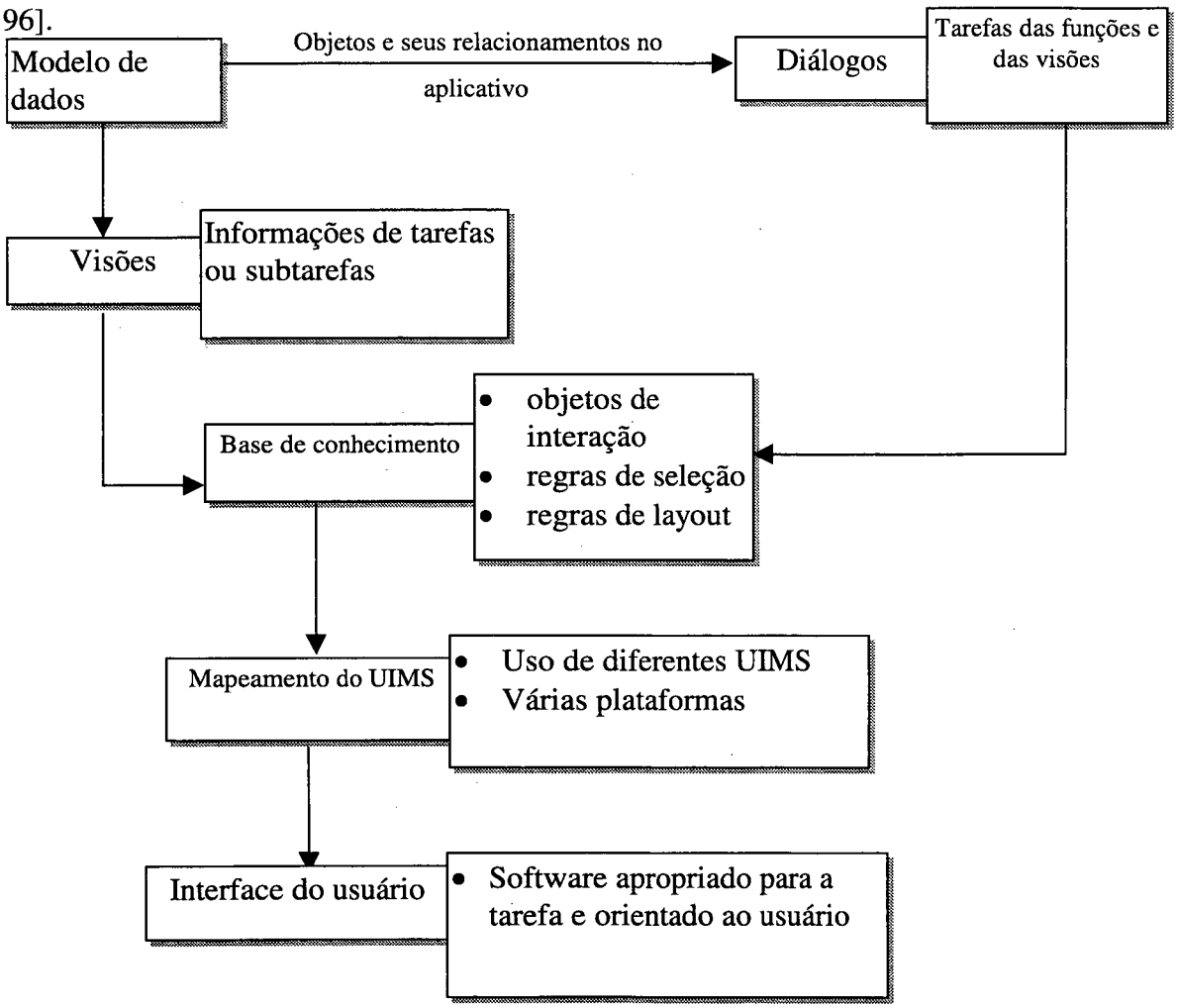

Figura 3.4 : Visão geral do ambiente do GENIUS.

3.5.2 DEFINIÇÃO DA INTERFACE DO USUÁRIO BASEADA NO MODELO DE DADOS

3.5.2.1 *EXTENSÕES PARA O MODELO ENTIDADE-RELACIONAMENTO*

O modelo entidade-relacionamento é consolidado e largamente utilizado na industria e suportado pela maioria das ferramentas CASE existentes. O modelo contém um conjunto completo de informações que o usuário pode desejar ver e manipular, de maneira a completar uma tarefa específica. Portanto, o modelo pode ser utilizado como base para o projeto da base de dados e para o projeto da interface do usuário e também pode ser usado para definir a apresentação dos elementos de dados e o mais importante a definição da estrutura de diálogo.

Entretanto, as metas do projeto da base de dados e da interface do usuário são diferentes, enquanto o esquema da base de dados é projetado para minimizar a redundância dos dados e aumentar a performance, a interface deve apoiar a completa realização da tarefa do usuário, facilitando o uso e o aprendizado das interações do sistema. Algumas características estão sendo acrescidas ao modelo entidade-relacionamento visando habilitá-lo a uma fácil e completa definição da informação necessária para a interface do usuário, são elas:

- Atributos complexos: são introduzidos para agregar atributos relacionados de uma entidade. Por exemplo a relação entre os atributos cidade, cep, rua e fone, são expressos pelo atributo complexo endereço;
- Atributos derivados: de modo a mostrar a informação que é necessária para uma tarefa especifica, mas não é explicitamente avaliada no modelo de dados. Por exemplo, a idade de uma pessoa pode ser calculada da data de aniversário e data atual;
- Grupos: Elementos de dados pertinentes a uma tarefa podem ser agrupados. Este agrupamento de informação é usado no processo de geração da interface do usuário quando os dados são estruturados na tela;

- Hierarquias: Para reduzir a complexidade da representação gráfica dos diagramas entidade-relacionamento, hierarquias são definidas e um conjunto de entidades e relacionamentos podem ser trocados por símbolos e editados separadamente.

3.5.2.2 *DEFINIÇÃO DE VISÕES*

São definidas visões específicas que providenciam uma representação da interface do usuário sem estar dependente do esquema da base de dados e para estruturar a informação contida no modelo de dados com respeito a tarefa do usuário. Uma visão consiste de um subconjunto de entidades, relacionamentos e atributos de um modelo de dados completo, que representam uma tarefa específica no conjunto de tarefas que são realizadas pelo sistema. Portanto, cada visão corresponde a uma janela ou sub-janela. De acordo com os elementos que formam a visão, dois diferentes tipos de visões são distinguíveis:

- Visão de agregação: É uma coleção de elementos de mesmo tipo, como todas as instancias de uma entidade. As funções associadas com uma agregação incluem funções de manipulação e de navegação, as quais habilitam o acesso a simples elementos contidos na agregação.
- Visão detalhada: Mostra os atributos simples de uma entidade. Pode apresentar todos ou somente alguns atributos de uma entidade, bem como atributos de entidades diferentes que são necessários para realização de uma tarefa específica. As funções de manipulação de dados podem ser acessadas implicitamente pela mudança do conteúdo do campo de dados ou explicitamente pela ativação da função que representa o objeto de interação, por exemplo, um item de menu ou botão. A estrutura do modelo de dados é preservado nas visões. Eles podem ser usados para determinar a apresentação dos atributos e para derivar a estrutura do diálogo.

3.5.2.3 *DEFINIÇÃO DE INFORMAÇÃO ADICIONAL COM TABELAS DE PROPRIEDADE*

Informações que não podem ser representadas graficamente, são formalizadas através de tabelas de propriedade, e consistem em três tipos de informação:

- Informação descritiva: Incluem uma definição para cada elemento e propriedades como tipo, faixa de valor, tipo de seleção e valores default armazenados serão utilizados para determinar o objeto de interação adequado para compor a interface;
- Informação estrutural: Mostra o relacionamento entre elementos individuais, são os elementos que formam uma visão e as funções associadas a esta visão. Existem as funções de manipulação de dados, que podem ser aplicados nos elementos de dados da visão e funções de navegação que definem a estrutura de diálogo para chamar outras visões. São usados para definir o comportamento dinâmico da interface do usuário;
- Informações sobre propriedades orientadas a tarefa: Incluem frequência, tipo de acesso e prioridades para a tarefa. Esta informação é utilizada para mapear os elementos da aplicação sobre os objetos de interação apropriados para a composição da interface.

3.5.3 OBTENDO A INTERFACE DO USUÁRIO ATRAVÉS DO MODELO DE DADOS

São usados no projeto da interface do usuário os elementos do modelo de dados, as visões e suas propriedades e também a estrutura do modelo de dados com suas relações e cardinalidades. Tanto a apresentação quanto a estrutura de diálogo podem ser obtidas através do modelo de dados.

3.5.3.1 *APRESENTAÇÃO DOS ATRIBUTOS*

Para a geração da interface do usuário, os elementos de uma visão são mapeados, visando selecionar o objeto de interação adequado para sua representação, considerando características dos atributos como tipo, faixa e número de valores (Tabela 3.1). O relacionamento entre as entidades também influenciam a apresentação e a estruturação dos elementos da visão.

Os atributos pertencentes a uma mesma entidade ou relação, são implicitamente agrupados, de modo a refletir sua conexão lógica para o usuário. Nem todos os atributos de uma entidade devem estar presentes em uma visão, somente aqueles que são necessários na especificação do contexto. Os atributos em uma visão simples não devem necessariamente

pertencer a uma entidade, mas em muitos casos eles pertencem a entidades próximas umas das outras.

	Objetos de interação abstratos	
Classificação	Elementares	Complexos
Objetos de diálogo		Janelas, caixas de diálogo
Objetos de ação	Seleção de operação	Menu
Objetos de apresentação	Campos permanentes	Grupos
	Campos de dados	Campos de dados com rótulo
	Escolha exclusiva	Lista
	Escolha não-exclusiva	Tabela
		Escala

Tabela 3.1: Objetos de interação abstratos para interfaces gráficas do usuário

3.5.3.2 *TRANSFORMAÇÃO DOS RELACIONAMENTOS*

A cardinalidade dos relacionamentos entre as entidades também influencia a apresentação dos atributos. Um relacionamento um-para-um entre duas entidades não possui efeito adicional sobre a apresentação dos atributos, porque eles são agrupados sempre de acordo com sua relação lógica. Em um relacionamento um-para-muitos, uma visão de agregação é necessária para mostrar mais que uma instancia de muitos relacionamentos (Figura 3.4).

Um relacionamento muitos-para-muitos pode ser representado por duas visões de agregação, por exemplo, listas que dependem uma da outra. Se um item de uma lista é selecionado, a segunda lista apresenta as instancias associadas ao elemento e vice-versa.

3.5.3.3 *OBTENDO A ESTRUTURA DE DIÁLOGO DO MODELO DE DADOS*

Uma interface do usuário baseada em objetos assim como o modelo de dados, são estruturados de acordo com os objetos de dados. Portanto, a estrutura do modelo de dados pode ser transferida para a estrutura de diálogo, que consiste em três principais partes: a primeira parte são as entradas da aplicação que apresenta todos os objetos relevantes da interface. Na segunda parte, uma seleção do diálogo é apresentada de maneira a identificar

o objeto desejado. A última parte mostra a informação detalhada do objeto desejado. Então, em um típico diálogo orientado a objetos, a navegação até o objeto desejado deve primeiro ser executado de modo a realizar uma tarefa específica.

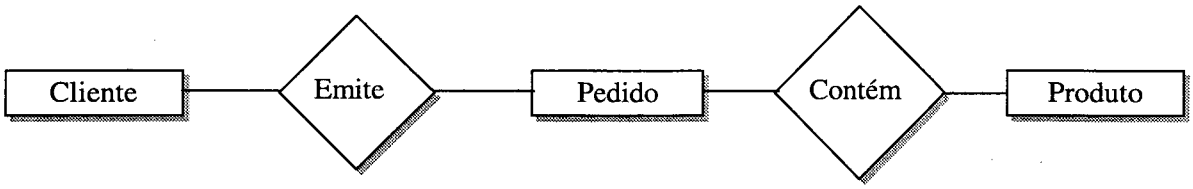

Figura 3.5 : Definição de uma visão para um diálogo de entrada de dados [Adaptado de Bullinger, et. all., 96].

A geração da interface do usuário através do modelo de dados produz a apresentação da informação em janelas e as seqüências de diálogo em aplicações orientadas a dados. Isto reduz o esforço e simplifica o projeto das interfaces. Além disso, tais seqüências de diálogo promovem a padronização da parte dinâmica da interface, que não é coberta pelos guias de estilo e ferramentas de desenvolvimento de interface existentes.

3.5.4 GERAÇÃO AUTOMÁTICA DAS INTERFACES DO USUÁRIO

As visões contém os elementos de dados para uma tarefa específica, formando uma descrição lógica das janelas do aplicativo. Esta descrição é usada como entrada para um sistema especialista que controla o processo de geração da interface. A base de conhecimento inclui os tipos de objetos de interação abstratos, as regras para a seleção e o layout dos objetos interação.

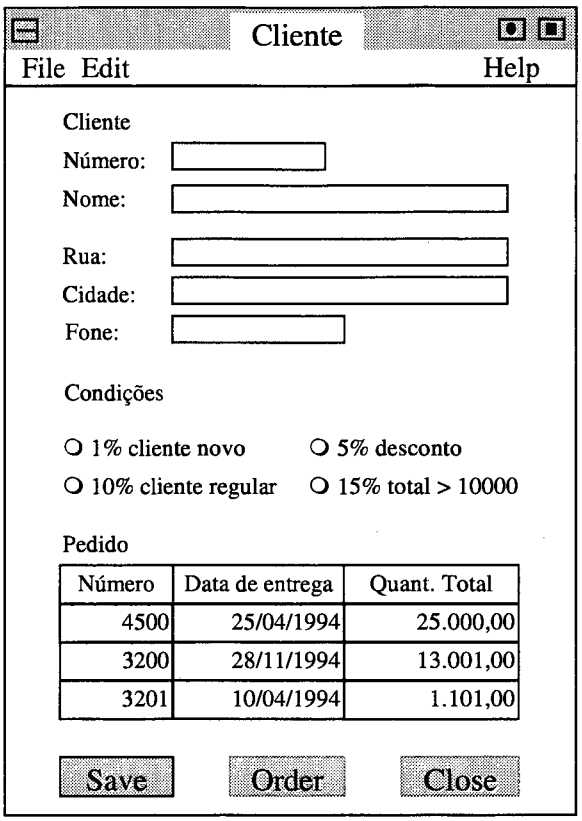

Figura 3.6 : Interface desenvolvida a partir da visão definida na figura 3.5 [Adaptado de Bullinger, et. all., 96].

3.5.4.1 *TIPOS DE OBJETOS DE INTERAÇÃO ABSTRATOS*

Os objetos de interação são chamados abstratos porque são independentes da implementação física, do ambiente e da plataforma. A definição destes objetos é baseada em diferentes guias de estilo (OSF/Motif, IBM 1992, Sun Microsystems e Microsoft) e de acordo com sua estrutura são classificados em elementares e complexos (Tabela 3.1). Os objetos de interação possuem atributos que devem ser preenchidos com os valores específicos do aplicativo como nome e valor, assim como atributos de layout como alinhamento, cor e tamanho. Alguns objetos possuem valores default para certos atributos, sendo que estes valores são carregados fora da base de conhecimento, permitindo que o projetista influencie no processo de desenvolvimento, aplicando mudanças de acordo com seu conhecimento ou padrão da empresa.

3.5.4.2 *REGRAS DE SELEÇÃO DE OBJETOS DE INTERAÇÃO ABSTRATOS*

As regras utilizadas são derivadas de guias de estilo, que apresentam regras para definir formato, localização e aparência. A tabela 3.2 mostra as regras para a seleção de objetos de interação.

A posição dos objetos de interação é determinada em dois passos. Primeiro, os objetos são colocados dentro de um grupo e então os grupos são posicionados. Para o cálculo da distância entre os objetos, um grid de layout é usado, o qual especifica através de unidades de medida (pixel) as distâncias corretas entre os diversos tipos de objetos. Aplicação dos objetos	Objetos de interação abstratos
Aplicação: • Apresentação • Entrada e apresentação • Apresentação e seleção	Objetos de diálogo • Janela • Janela com ícones • Janela com critérios de seleção
Funções • Funções padrão - usadas frequentemente • Funções específicas da aplicação – usadas frequentemente • Outros	Objetos de ação • Menu e operações • Menu e operações, operações de controle • Operações de controle
Objetos de dados • Escolha exclusiva – alfanumérico - 2 = número de escolhas = - número de escolhas < 6 • Escolha não – exclusiva – alfanumérico - número ≤ 6 - número > 6 - numérico - valores máximos e mínimos - números ≤ 60 • Outros – faixa de entrada e saída desconhecida – acesso para leitura – acesso para escrita	Objetos de controle • Controle de escolha exclusiva • Listas de seleção simples • Escolhas não – exclusivas • Listas de múltipla seleção • Escala • Campo para leitura de dados • Campo para entrada de dados

Tabela 3.2 : Regras para a seleção de objetos de interação abstratos

3.5.4.3 *REGRAS DE LAYOUT*

As regras de layout são definidas considerando a estrutura específica dos dados do aplicativo e as leis de percepção e cognição para o sistema visual humano, tais como proximidade, similaridade, consistência, continuidade e simetria. São definidos três diferentes algoritmos para o layout e são usados dependendo da sua conveniência. Baseando-se no tamanho da janela, os objetos de interação serão arranjados. As proporções da janela influenciam no layout, no balanço horizontal e na simetria esquerda/direita. A quantidade de objetos é maior que o possível de ser apresentado em uma janela, um terceiro algoritmo de layout é usado para produzir uma janela com barras de rolamento, permitindo a visualização de toda a informação e mostrando para o desenvolvedor a necessidade de redefinição das visões.

3.5.4.4 *O PROCESSO DE GERAÇÃO*

A geração da descrição da interface do usuário é realizada em três passos. No primeiro passo, os objetos apropriados para representarem a informação contida em uma visão são selecionados nas tabelas de propriedade de acordo com o tipo de dado, faixa e tipo de acesso (tabela 3.2). Uma janela é criada para cada visão e os atributos de layout considerados são os default. Para funções, o seu tipo, escopo e freqüência de uso determinam como estas serão representadas, como uma barra de menus ou um conjunto de botões ou lista de seleção. A barra de menu padrão é especificada de acordo com o estilo desejado e funções adicionais serão automaticamente integradas.

Um segundo passo é utilizado para determinar os valores dos atributos para os objetos de interação selecionados. Os atributos que dependem apenas do conteúdo da aplicação (tais como rótulos, tamanho de campo, formato e valores default) são copiados das tabelas de propriedades. Um segundo tipo de atributo é obtido pelos valores default que existem em cada objeto de interação. O terceiro tipo de atributos, principalmente os geométricos e de informação de layout, dependem da interdependência entre os objetos de interação. Neste passo, o tamanho de cada objeto e a posição relativa de um objeto dentro de um grupo é calculada.

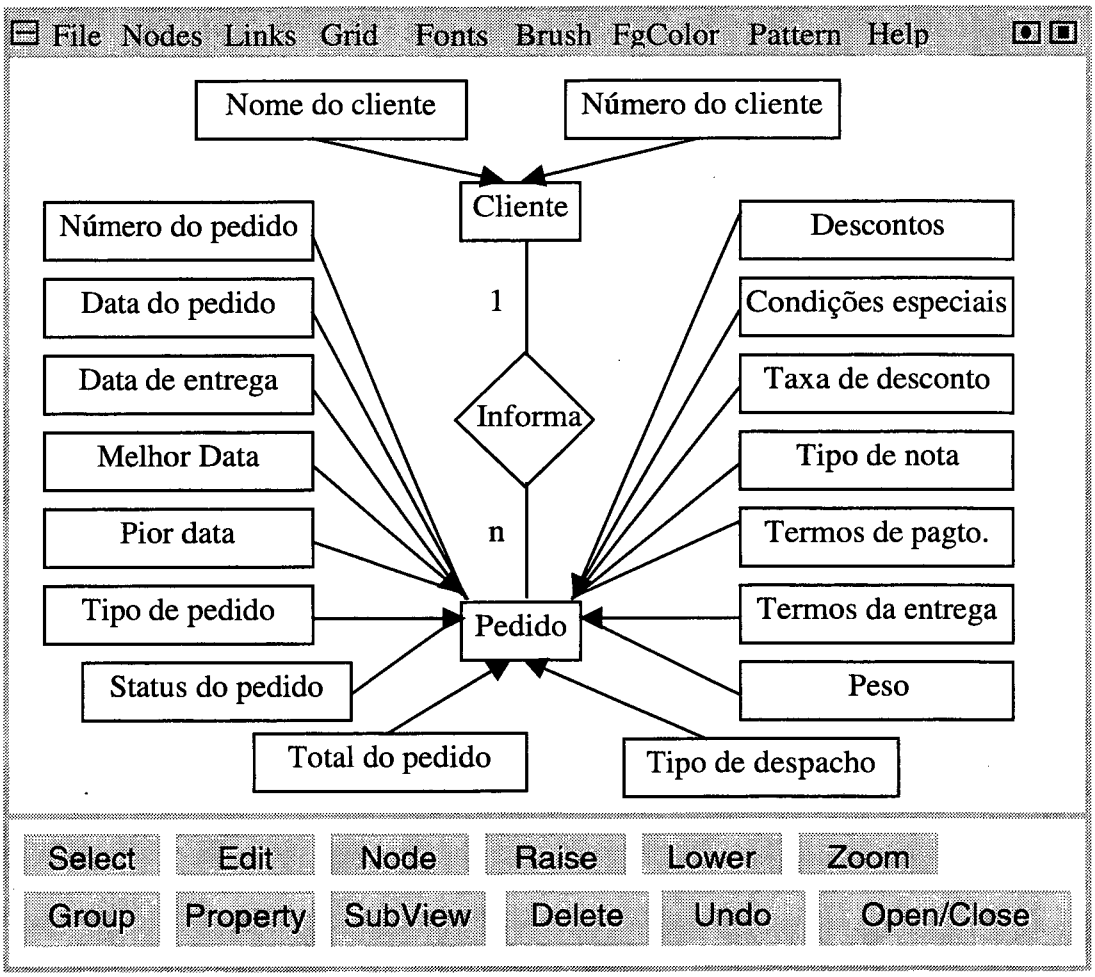

Figura 3.7 : Geração da descrição da interface (visão) do usuário no ambiente Genius

Os elementos são arranjados sobre o espaço avaliado da tela, de acordo com sua prioridade ou a uma seqüência determinada na visão. O sistema especialista libera uma descrição da interface que é transformada em uma especificação interpretada pelo UIMS escolhido. Esta transformação permite a utilização de vários UIMS's para diferentes ambientes.

3.5.4.5 *A IMPLEMENTAÇÃO DO AMBIENTE GENIUS*

A ferramenta GENIUS roda em estações Sparc SUN sob o sistema X Windows e é formada pelos seguintes componentes:

- Um editor gráfico para desenvolver o modelo de entidade-relacionamento e a definição das visões. O editor fornece as tabelas de propriedades. Este editor foi implementado usando o InterViews toolkit.

- Uma base de dados para carregar as informações das tabelas de propriedade. Para a implementação das tabelas da base de dados é utilizado o SGBD Oracle.

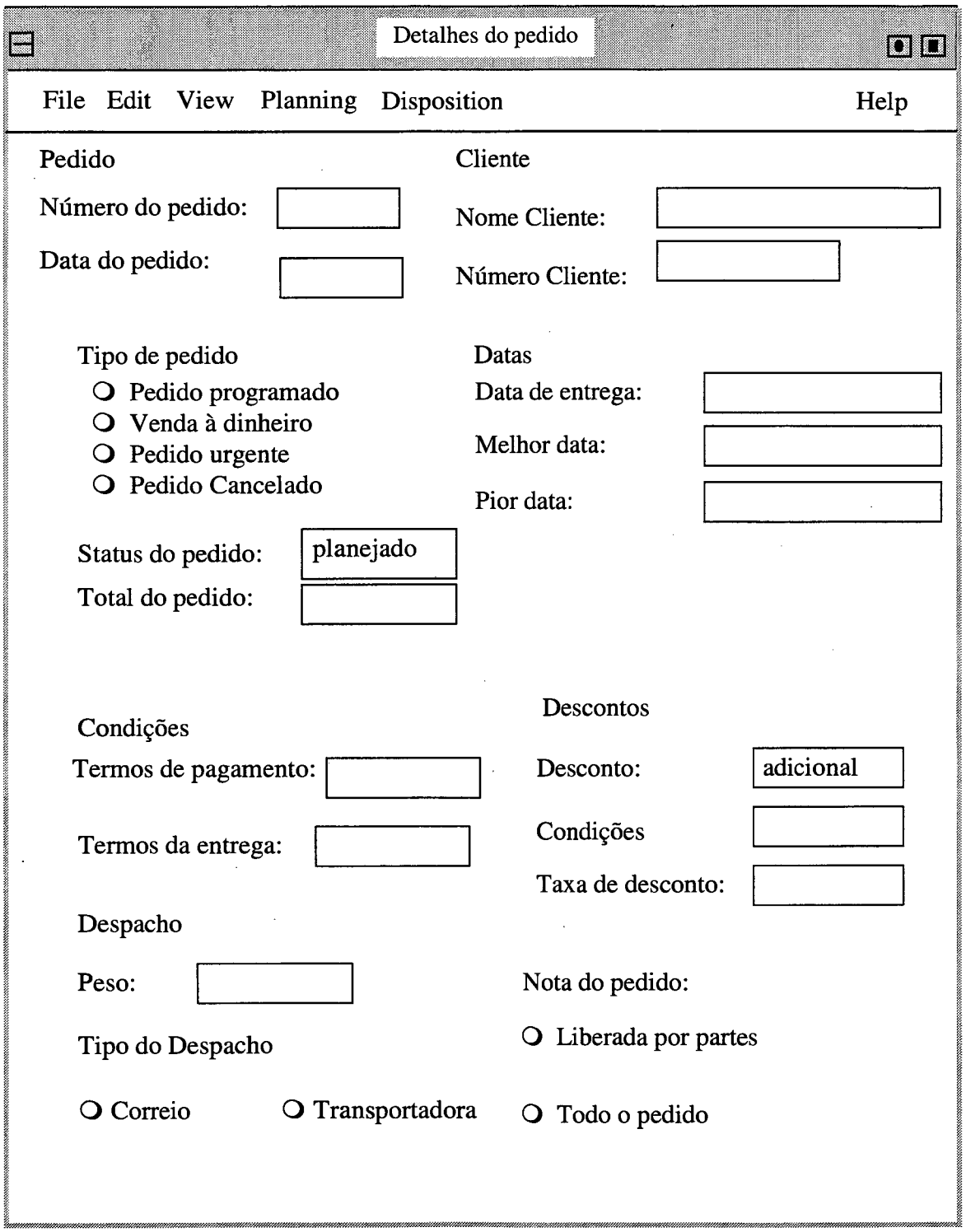

Figura 3.8 : Uma visão definida para um pedido e a janela automaticamente gerada [Adaptado de Bullinger, et. all., 96].

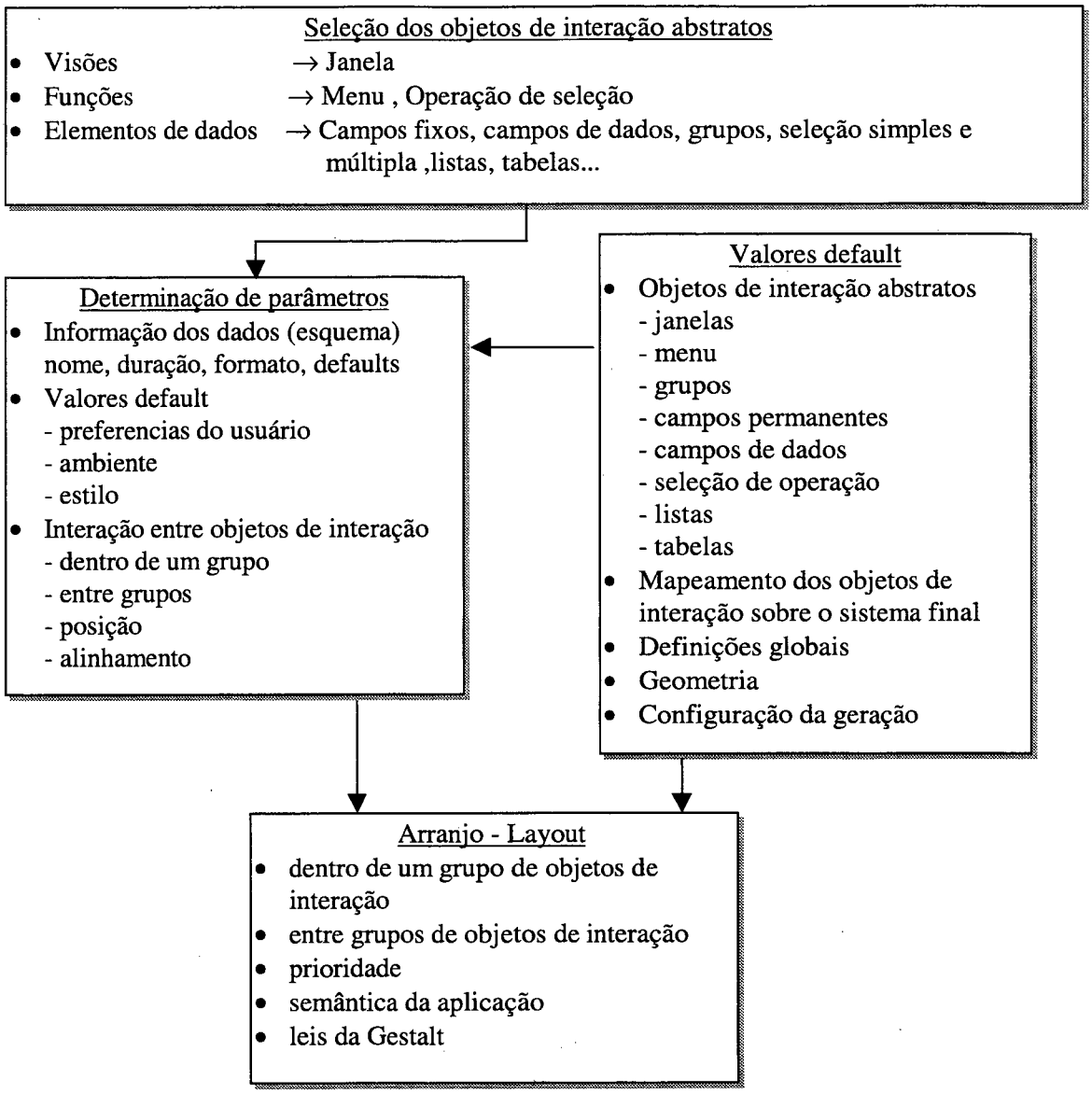

Figura 3.9 : Esquema representativo dos passos para a geração da especificação da interface [Bullinger, et. all., 96].

- Um sistema baseado em regras, implementado usando o shell híbrido do sistema especialista Nexpert Object. Os objetos de interação são definidos em classes e para a seleção e regras de layout, regras de produção são usadas.
- Um editor gráfico para a definição e manipulação da interatividade dos valores default, de modo a influenciar no processo de desenvolvimento.
- Um UIMS, para a apresentação e execução da geração da interface.

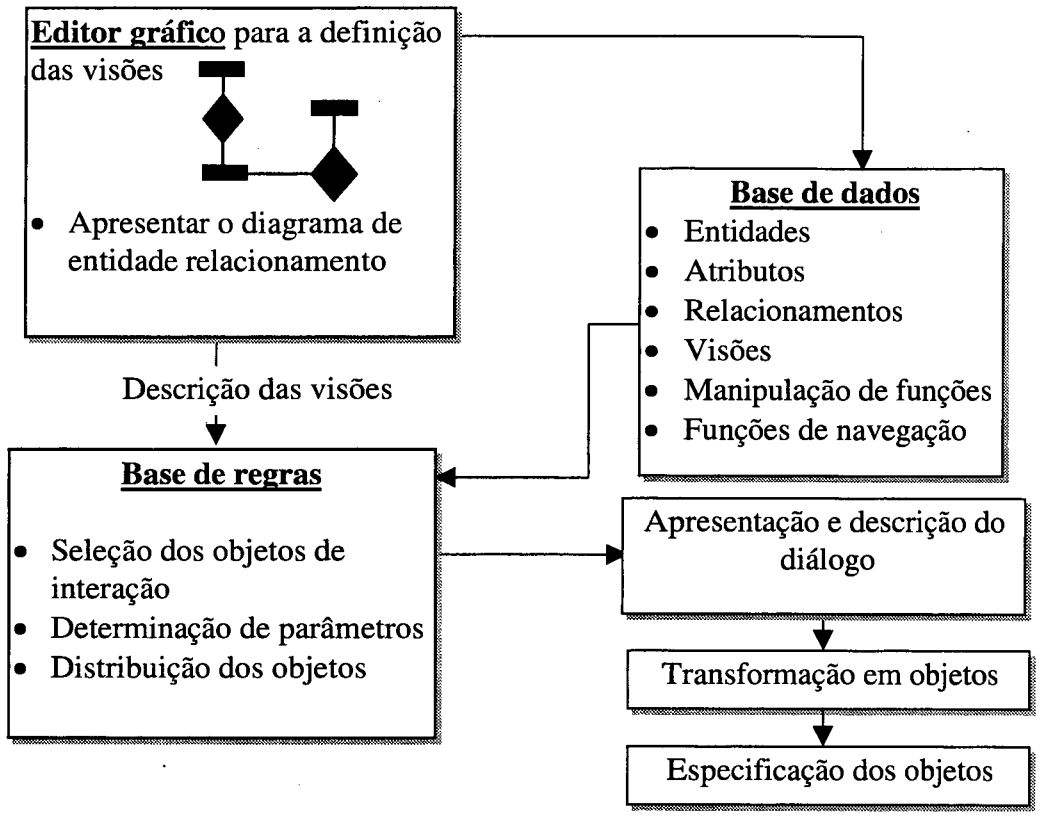

Figura 3.10 : A arquitetura do ambiente da ferramenta GENIUS.

3.6 CONCLUSÕES

As considerações finais a respeito das técnicas apresentadas anteriormente, são afirmações de seus autores e pesquisadores que efetivamente utilizaram cada técnica respectiva, para conduzir um experimento de caráter científico.

Cybis [Cybis, 1997], argumenta que a *Análise Ergonômica do Trabalho* pode ser usada para conduzir a especificação de um novo sistema. A abordagem é de conhecer a tarefa para modificá-la. Com a descrição formal do estado de coisas existentes o analista pode avaliar quais objetos e tarefas deveriam estar no futuro sistema. Ela também pode ser empregada para esclarecer os projetistas sobre o que se pretende com novas funcionalidades. A partir da AET, pode-se também fazer predições sobre o uso do novo sistema. Pode-se investigar como as partes a ser incluídas irão interagir com os procedimentos existentes, e de como as informações irão entrar e sair do futuro sistema.

Isso pode ser realizado através da análise de um sistema que tenha certas características análogas com o sistema pretendido. A estrutura genérica assim produzida pode servir de modelo interno do novo sistema. Ela será modificada para acomodar as novas funcionalidades, mantendo-se compatível com as expectativas reais dos usuários.

Lim [Lim, 96], esclarece que o método *MUSE* reduz a necessidade de reprojeto durante o processo de desenvolvimento, pois evita divergências entre as equipes defensoras dos fatores humanos e aqueles que defendem as metodologias da engenharia de software puramente. A possibilidade de discussão das questões de projeto, logo nos primeiros estágios do processo de desenvolvimento, resulta na produção de interfaces do usuário de alta qualidade, visto que o projeto de alto nível é apropriadamente especificado antes de iniciar o projeto detalhado. Também assegura que o sistema de documentação especificado pelo método *MUSE* facilita o reuso do trabalho de projetos anteriores, reduzindo a necessidade de duplicação de análise para aplicativos semelhantes. Cada produto *MUSE* possui uma tabela, que contém detalhes adicionais dos diagramas gerados, bem como observações e especulações concernentes às idéias surgidas durante o projeto. O método sugere que os documentos gerados pela sua especificação e os gerados pela técnica de engenharia de software sejam comparados e checados antes que se inicie a implementação do projeto final.

Gulliksen [Gulliksen et all, 97], explica que a *AIU* serve como um complemento para metodologias de projeto existentes e como um método independente para capturar o conhecimento do domínio da tarefa para o projeto de interfaces do usuário. Captura também os principais aspectos de como a informação está sendo usada e certas características da informação que não foram identificadas no processo de modelagem dos dados e podem constituir melhores bases para o projeto de interfaces do usuário, servindo para construir um primeiro protótipo da interface. Ainda, o método incrementa a eficiência do desenvolvimento da interface, reduzindo o tamanho da equipe e consequentemente os custos e é incrementada a consistência da interface, devido a participação de usuários finais.

Bullinger [Bullinger, et. all., 96], assegura que comparando com os sistemas de gerenciamento de interfaces, o ambiente GENIUS pode incrementar a produtividade e a qualidade da interface produzida, devido principalmente à possibilidade de seleção automática dos objetos de interação, da definição de seus atributos e o posicionamento destes dentro da interface que está sendo construída. A geração pelo princípio de regras ergonômicas assegura a consistência no uso dos objetos de interação assim como a consistência no projeto do diálogo, ainda, as regras ergonômicas garantem a aplicação de fatores humanos às interfaces. O uso de modelos de dados garante a integração do desenvolvimento da aplicação e o projeto da interface do usuário através do uso consistente dos dados relativos à aplicação em ambas as áreas. O ambiente GENIUS oferece ainda a possibilidade de gerar protótipos nas primeiras fases do projeto, apoiando o projeto ergonômico do software e da interface relacionada.

CAPÍTULO IV

4. FERRAMENTAS DE DESENVOLVIMENTO DE INTERFACES

4.1 UMA VISÃO GERAL

Até o ano de 1984, apenas engenheiros de software eram os profissionais capazes de desenvolver interfaces gráficas do usuário (GUI's). Neste tempo as ferramentas disponíveis eram o Macintosh toolbox, o Digital Research's GEM, as intermináveis versões beta do Microsoft Windows 1.0 e o novíssimo VAX – Window System X. Estas ferramentas eram primitivas e exigiam perícia de um complexo GUI's – API (Interface Gráfica do Usuário - *Aplication Program Interface*). Além disso os aplicativos eram geralmente escritos em C, uma linguagem complexa e de difícil aprendizado, onde os erros dos programadores eram difíceis de serem solucionados.

Anos mais tarde (1989), durante o desenvolvimento do projeto "*One computer for all*", engenheiros da Apple Corporation verificaram a necessidade de um ambiente de desenvolvimento de GUI'S "Para todos nós", como resultado surgiu o HyperCard, o primeiro ambiente de desenvolvimento completamente projetado para não programadores. Desde então, muitas outras ferramentas de programação destinadas a profissionais responsáveis pelo desenvolvimento de interfaces e aplicativos foram desenvolvidas, como por exemplo o SuperCard e o Toolbook, os Builders e os UIMS, e também Visual Basic, Borland Delphi e mais recentemente C++ e PowerBuilder, entre outras ferramentas conhecidas hoje como Linguagens de Quarta Geração (4GL), caixas de ferramentas que oferecem um conjunto de objetos de interação (widgets), com aparência específica e comportamento particular (look and feel), que podem ser manuseados pelo usuário e um rico conjunto de aplicações chamadas rotinas, usadas para especificar e manipular estes objetos.

Com a chegada destas ferramentas, tornou-se possível dividir o esforço de desenvolvimento de GUI's entre os diversos componentes de uma equipe, facilitando também a interação de profissionais com diferentes especializações. Devido ao fato de

estas ferramentas atuarem no início do ciclo de vida de um aplicativo e independente do resto da aplicação, elas podem promover o aumento da produtividade e da qualidade das interfaces produzidas obstante do tipo de profissional que às produziu.

4.2 A IMPORTÂNCIA DE USAR FERRAMENTAS PARA DESENVOLVER INTERFACES GRÁFICAS

Uma vez que o desenvolvimento de interfaces do usuário é uma tarefa de difícil realização devido aos diversos fatores que envolvem a sua concepção, é de se esperar que muitas pesquisas sejam realizadas no intuito de progredir e manter atualizadas as ferramentas de desenvolvimento de GUI's. Os sistemas evoluem rapidamente, e os modelos que foram populares a poucos anos atrás, estão hoje obsoletos devido às novas tecnologias, às mudanças no mercado de computadores, o surgimento de novos estilos de interface e as novas maneiras de comunicação entre homem e máquina.

Segundo Myers [Myers, 96b], existem muitas vantagens no uso de ferramentas de desenvolvimento de GUI's, estas podem ser classificadas dentro de dois grupos principais:

a) Aquelas que melhoram a qualidade da interface. Principalmente porque:

- O projeto da interface pode ser rapidamente prototipado e implementado, permitindo avaliações prévias, antes que o código da aplicação seja escrito;
- Torna-se mais fácil incorporar mudanças ao projeto, através de testes com o usuário;
- Diferentes aplicativos são mais prováveis de possuir interfaces consistentes se eles são criados utilizando a mesma ferramenta;
- A redução do esforço de desenvolvimento das interfaces é alcançado devido a seqüência de utilização da ferramenta na implementação de diferentes aplicativos;
- A ferramenta permite que uma variedade de especialistas estejam envolvidos no projeto da interface, ao invés de a interface ser totalmente desenvolvida por programadores. Artistas gráficos, psicólogos cognitivistas e especialistas em fatores humanos podem ser envolvidos. Em particular, projetistas profissionais de interfaces,

que não são programadores, podem realizar todo o projeto de desenvolvimento da interface.

b) Aquelas que facilitam e otimizam o desenvolvimento do código da interface. Devido principalmente aos seguintes fatores:

- A especificação da interface pode ser representada, validada, e avaliada mais facilmente;
- Existe menos código a escrever, porque grande parte é implementado pela ferramenta;
- Existirá melhor modularização devido a separação dos componentes da interface dos componentes do aplicativo. Isto permite mudar a interface sem afetar o aplicativo, e uma grande classe de mudanças no aplicativo (algoritmos internos) podem ser realizadas sem afetar a interface;
- O nível de experiência em programação dos projetistas da interface pode ser baixo, pois a ferramenta esconde a maioria das complexidades de desenvolvimento da interface;
- A confiabilidade da interface será maior, uma vez que seu código é criado automaticamente numa especificação de alto nível;
- A portabilidade do aplicativo para diferentes ambientes e plataformas será facilitada uma vez que a dependência de dispositivos é isolada pela ferramenta.

4.2.1 FUNÇÕES DESEJÁVEIS EM FERRAMENTAS DE DESENVOLVIMENTO DE GUI's

Durante o processo de seleção de uma ferramenta de desenvolvimento de interfaces, alguns critérios devem ser contemplados de maneira a assegurar que esta ferramenta suprirá as necessidades exigidas pelo projeto, permitirá o aumento do desempenho da equipe e produzirá interfaces com maior qualidade. Tais critérios são comentados a seguir:

USABILIDADE: Uma ferramenta para desenvolvimento de GUI's deve simplificar o processo padrão de desenvolvimento da equipe. A ferramenta deve ser fácil de instalar, aprender e usar. Deve também providenciar diferentes níveis de ajuda, adequados aos diferentes níveis de habilidade e conhecimento dos integrantes da equipe. Quando uma ferramenta não contempla algumas destas qualidades acaba reduzindo a produtividade e frustrando as expectativas dos projetistas.

FUNCIONALIDADE: A funcionalidade básica da ferramenta deve equiparar-se com os requisitos da interface. A ferramenta deve possuir os objetos de interação (widgets) necessários para o desenvolvimento da interface ou permitir que conjuntos de objetos externos sejam anexados à sua biblioteca de objetos. É interessante que a ferramenta suporte diversas linguagens, padrões e plataformas, facilitando a integração entre ambientes, a reutilização de código fonte e a portabilidade. A geração automática de código fonte é preferível pois incrementa a produtividade.

FLEXIBILIDADE: Os requisitos do sistema podem rapidamente mudar durante o projeto e o desenvolvimento da aplicação, portanto, a ferramenta deve suportar diferentes metodologias de projeto, arquiteturas de software, linguagens de programação, plataformas de hardware e diferentes línguas (humanas). A ferramenta deve estar preparada para gerenciar possíveis alterações no projeto mantendo a integridade do aplicativo.

PORTABILIDADE: Uma ferramenta para desenvolvimento de GUI's deve permitir a produção de aplicações que sejam portáveis facilmente através de múltiplas plataformas ou de plataformas com características semelhantes. Se um aplicativo deve rodar sobre várias plataformas, a abordagem API (Aplication Program Interface) em particular deve ir de encontro aos requisitos e conceitos operacionais do aplicativo.

SUPORTE: Uma ferramenta deve ser acompanhada de documentação, treinamento e ajuda on-line. Sendo que a qualidade destes elementos revelam a maturidade, a confiabilidade e a estabilidade da ferramenta.

CUSTO: Elemento difícil de ser avaliado, pois dependem da ferramenta, suporte, treinamento, manutenção etc., é um critério relativo, mas decisivo no momento da escolha

de comprar uma ferramenta. Deve ser criterioso pois a escolha de uma ferramenta pelo menor custo pode levar ao desastre todo um projeto.

4.3 CLASSIFICAÇÃO DAS FERRAMENTAS DE DESENVOLVIMENTO DE INTERFACES

Existem diversas propostas de classificação para as ferramentas de interfaces e estas podem ser classificadas de acordo com diversos aspectos. Um aspecto importante das ferramentas, é para que tipo de profissional ela se destina, se para analistas, projetistas, programadores, avaliadores ou usuários finais. Um outro aspecto importante a considerar-se no estudo das ferramentas[1] é a etapa do desenvolvimento em que ela se insere. Ela pode ser utilizada na fase de projeto, de implementação ou de execução. Com base nestes critérios, pode-se classificar as ferramentas em duas variedades básicas:

- A mais convencional é simplesmente uma coleção de procedimentos que podem ser chamados por programas aplicativos (*callbacks*), ou seja, bibliotecas de rotinas usadas por programadores para a implementação de detalhes de baixo nível. Em geral, quando se emprega uma *toolkit* na implementação de uma interface, o controle reside no componente computacional ou aplicação, que gerencia os eventos resultantes da manipulação dos objetos de interação incluídos na interface. São exemplos de *toolkits* o Xt Intrinsics, o Motif Widgets, o Xview, o Macintosh Toolbox, o SunTools, o InterViews, o Andrew e o SunTools *Toolkits*.

- A outra variedade usa um estilo de programação orientada a objetos, o que torna mais fácil para o projetista customizar as técnicas de interação. Como exemplo deste estilo temos o Smalltalk, Garnet, InterViews, Amulet e o Java *Toolkits* AWT. A vantagem de usar os princípios da orientação a objetos, é que esta é a maneira natural de pensar sobre objetos (os objetos da interface são vistos como objetos), e os próprios objetos podem manipular algumas tarefas que em outros ambientes seriam realizadas pelo programador. Também é mais fácil a criação de novos objetos através de instanciação ou criação de sub-

[1] As ferramentas citadas neste texto, são algumas produto de pesquisas acadêmicas e outras produzidas por empresas especializadas. O anexo 1 deste trabalho traz mais referências a respeito destas ferramentas.

classes de objetos já existentes e a anexação de bibliotecas de objetos desenvolvidas em outros sistemas é facilitada.

Entretanto, se considerar a classificação das ferramentas segundo diferentes funcionalidades e por sua adequação para determinadas aplicações pode-se utilizar uma taxonomia representada por quatro grandes grupos de ambientes. As *toolkits,* as ferramentas CASE, os *Builders* e os UIMS. Apresenta-se a seguir considerações levantadas durante avaliações técnicas de algumas ferramentas que fazem parte destes grupos:

4.3.1 *TOOLKITS* ESPECIALIZADAS

Geralmente uma *toolkit*, é uma coleção de bibliotecas para desenvolvimento de GUI's utilizada em conjunto com apropriadas ferramentas de programação (programas aplicativos), para a implementação de interfaces gráficas. A interface é manipulada e implementada pela *toolkit*, enquanto que os dados e seus relacionamentos possuem ferramentas específicas para sua programação. O controle global da aplicação desenvolvida por uma *toolkit* está sempre no componente computacional.

Existem, entretanto, problemas com as *toolkits*, primeiro, são os estilos de interação que ficam restritos àqueles oferecidos pela ferramenta e que limitam a criatividade e a evolução dos aplicativos desenvolvidos e, segundo, é que a maioria são difíceis de usar, uma vez que são formados por centenas de procedimentos, e muitas vezes não está claro como usá-los para criar a interface desejada. Ainda, segundo Cardelli [Cardelli, 85], a*s toolkits oneram o custo final do produto: "As primitivas que compõem as toolkits a princípio não são vistas como complexas, mas os programas que as implementam são surpreendentemente confusos".*

O tipo básico é formado por *widgets*, que são objetos que encapsulam as técnicas de interação, da aparência e do comportamento, usada em aplicativos informatizados. O *widget* contém informações a respeito de objetos gráficos, e a interface é constituída de instâncias desses objetos. Quase toda *toolkit* contém algum tipo de *widget* de posição, que serve para agrupar e dar ordem a outros *widgets*, como por exemplo formulários e *frames*.

As *toolkits* interagem com o processo computacional através de chamadas (*callbacks*), que são invocadas através de procedimentos definidos pelo programador em resposta a um evento causado pelo usuário em tempo de execução. Quando o usuário interage com um *widget*, o código da interface chama o código computacional para dar a resposta ao usuário.

As funções de uma *toolkit* são bastante limitadas, sem o apoio de um *Builder* ou *UIMS*, não é possível a prototipação rápida da interface, entretanto, permitem uma grande flexibilidade para o programador, apesar de apenas ajudar na construção da interface. Geralmente são construídas em cima de sistemas de janelas, e muitos sistemas de janelas conhecidos provêm uma *toolkit*, como por exemplo o X WindowSystem, o VisualBasic, o Borland Delphi e o C++ Builder. Os sistemas de janelas tornaram-se tão populares, que muitas das ferramentas provêm para eles o gerenciador de janelas, onde o usuário pode ter diferentes contextos em janelas diferentes. Através do gerenciador de janelas ele pode pular de contexto em contexto, sem que um interfira no outro. O mais conhecido exemplo é a série Windows, da Microsoft. O X WindowSystem apresenta uma *toolkit* um pouco diferente, pois além do sistema não apresentar sistema de gerenciador de janelas, ele provém uma *toolkit* com *widgets* básicos que devem ser agrupados a fim de formar *widgets* mais complexos, tudo programado por código puro, sem manipulação direta.

Atualmente o desenvolvimento de *toolkits* têm se especializado em produzir conjuntos de objetos para aplicações específicas ou para específicas classes de programadores. Por exemplo, o sistema SUIT que contém uma *toolkit* e um construtor de interfaces, é especialmente projetado para ser fácil de aprender e é dirigido para sistemas de ensino em sala de aula; o Amulet [Myers, 96a], providencia apoio de alto nível para manipulação direta de interfaces composta por elementos gráficos e manipula as entradas como comandos hierárquicos sobre os objetos; o Rendezvous e o Visual Obliq são projetados para tornar mais fácil a criação de aplicativos que suportem múltiplos usuários ou múltiplas máquinas trabalhando sincronizadamente.

4.3.1.1 *CONSIDERAÇÕES SOBRE OS WIDGETS ESPECIALIZADOS*

Caracterizam-se como um conjunto de objetos de interface que possuem funcionalidades próprias. Produzem interfaces baseadas num determinado estilo e padrão. Dois exemplos bastante conhecidos são o X/Motif e o Open Look, que seguem o padrão da Apple. Como os padrões de guias de estilo comerciais, essas ferramentas auxiliam o desenvolvedor a saber como o objeto de interação parecerá e, às vezes, como se comportará, apesar de não indicar quando nem como utilizá-los.

Como seguem um estilo padronizado, as ferramentas não permitem, muitas vezes, que o desenvolvedor altere os objetos de interação. Essa padronização muitas vezes serve como motivo para a não realização de testes e interações com os usuários, pois a interface segue um padrão bem definido e testado. O padrão garante uma interface consistente, mas não garante a usabilidade de um sistema, sendo ainda necessários os testes e refinamentos da interface.

O objetivo de se utilizar essas ferramentas é produzir uma interface consistente que se pareça com algo que o usuário esteja acostumado. Ao se comparar as diferentes ferramentas desse estilo, bem como seus padrões, encontra-se mais semelhanças que diferenças. A maioria é baseada em janelas, e mantém componentes e controles análogos e utiliza-se de menus *pull-down* e *pop-up*. A maior diferença entre elas é geralmente a aparência dos objetos de interação que elas contêm, e raramente no comportamento. Objetos individuais ou bibliotecas de objetos de diferentes fabricantes podem ser integradas para prover o projetista com as necessárias ferramentas de desenvolvimento. Exemplos incluem o GroupKit e o INT Widgets.

4.3.2 FERRAMENTAS CASE

Ferramenta CASE (Computer-Aided Software Engineering), é qualquer ferramenta que auxilia analistas, projetistas e programadores durante todo o ciclo de desenvolvimento de sistemas informatizados (análise, projeto, implementação, teste e manutenção). Há também ferramentas CASE para apoiar a gerência dos projetos de desenvolvimento,

proporcionando ao desenvolvedor a capacidade de automatizar atividades manuais e de melhorar a manipulação da informação, além de apoiar as metodologias e métodos que vão surgindo.

As ferramentas CASE agregam maior qualidade aos produtos desenvolvidos, reduzindo a probabilidade de erros, uma vez que controlam a consistência dos dados e proporcionam maior eficácia aos produtos, quando auxiliam as fases de análise e teste do produto pelo usuário. A produtividade na construção de aplicativos fica favorecida, através da realização automática de algumas tarefas, reduzindo erros sintáticos e liberando os projetistas para a tomada de decisões em fatores de maior importância no projeto.

As ferramentas CASE são flexíveis em relação à mudanças, permitindo que sejam alterados dados e diagramas durante o desenvolvimento do sistema, eliminando ou reduzindo a programação necessária para implementar estas modificações. Por armazenarem dados e diagramas, as ferramentas também contribuem para uma melhor documentação do sistema, agilizando relatórios, busca de informações e alterações, facilitando por consequência a manutenção do sistema (correção, atualização ou expansão).

4.3.2.1 *CLASSIFICAÇÃO DAS FERRAMENTAS CASE*[2]

As ferramentas CASE podem ser classificadas por função, pela atuação como instrumento de controle para os gerentes, pela utilidade etapas do processo de engenharia de software, pela arquitetura do ambiente (hardware e software) ou custo delas. Aqui adotou-se a classificação por função.

- Ferramentas de Análise e Projeto : desenvolvem um modelo do sistema que será construído. Ex. BX: GUI Builder for C, C++ & Java;

- Ferramentas SA/SD: utilizam a análise estruturada (SA) e projeto estruturado (SD). Estão nesta categoria ferramentas como a Axiom-SA: Structured Analysis with Real-Time Extensions [STG, Inc.], TurboCASE: SA & SD [StructSoft, Inc.];

- Ferramentas PRO/SIM: oferecem ao projetista um modo de criar modelos funcionais e comportamentais de um sistema. Como exemplo cita-se: DEBUG/sw and DESIGN/sw: Development Information System, Common Debugger Interface;

- Ferramentas de Projeto e Desenvolvimento de Interfaces: os sistemas de desenvolvimento de interfaces do usuário UIDS (User Interface Development Systems) combinam ferramentas CASE individuais para interação humano-computador com uma biblioteca de componentes de programa que possibilitam ao projetista a criação interativa da interface. As novas linguagens de programação como Visual-Basic [Microsoft], Delphi [Borland] e Borland-C [Borland] já trazem embutidas estas ferramentas. Entretanto, estas ferramentas ainda devem ser aperfeiçoadas para permitir a criação de interfaces dentro de padrões ergonômicos internacionalmente aceitos.

- Ferramenta de Programação: abrange compiladores, editores, depuradores e ambientes de programação orientados a objetos (O-O), as linguagens de quarta geração, os geradores de aplicações e as linguagens de consulta a banco de dados situam-se nesta categoria. Como exemplo tem-se : Rational Rose: object-oriented analysis and Design [Rational Software Corporation.];

- Ferramentas de Codificação Convencionais de Quarta Geração: aparecem embutidas nas ferramentas CASE disponíveis. Cita-se como exemplo, GOOEY: GUI [LTG, Inc.], Informix 4GL, Object-Oriented Designer: OOAD, Freeware, OMT [LTG, Inc.];

- Ferramentas de Programação Orientadas a Objeto: são novas ferramentas para desenvolvimento de software, ligadas a uma linguagem de programação específica (C++, Eiffel, Object-C ou Smalltalk), onde incorpora características de interface de terceira geração (mouse, janelas, menus suspensos, operações contextuais e multitarefas) com funções especializadas como o browser. Nesta classe de ferramentas temos Objecteering: UML tool, C++, Java.

---

[2] Maiores informações sobre ferramentas CASE são encontradas em http://www.quais.queensu.ca/software-engineering/tools.html .

4.3.3 OS GUI's BUILDER

Também conhecidos como ferramentas de desenvolvimento de interfaces, são úteis para o desenvolvimento de interfaces complexas, incrementam a velocidade de geração das telas gráficas e automaticamente geram o código fonte da interface. Ainda, em aplicativos que são susceptíveis a mudanças da interface tais como aplicações de comando e controle, o uso de *Builders* incrementa a habilidade de adicionar e modificar a funcionalidade da interface em tempos mínimos para apoiar mudanças da tarefa ou novos requisitos.

Estas ferramentas trabalham melhor quando usadas no desenvolvimento de novos aplicativos ou na reengenharia de sistemas existentes. Para assegurar todas as vantagens de se usar os *Builders*, a arquitetura de software selecionada, deve permitir a distinção entre o código da interface e código do aplicativo (camadas). Esta separação simplifica grande parte da manipulação do software como um todo, permitindo que mudanças na interface sejam realizadas facilmente e que o esforço para o desenvolvimento e a manutenção do aplicativo sejam reduzidos.

Com os *Builders* o projetista pode criar e manipular os objetos da interface, através de um mecanismo de aponta e clica, em um ambiente "WYSIWYG" "o que você vê é o que você tem", permitindo a criação rápida de protótipos, que facilitam experimentos iniciais com a interface. Uma vez que a interface tenha sido satisfatoriamente projetada e todas as chamadas de rotinas estejam registradas, o GUI Builder gera o código para a interface sobre o *toolkit* nativo no qual ele foi construído (ex. OSF/Motif). Não existe suporte para especificação de comportamento para os objetos criados na interface. Algumas ferramentas deste grupo também se destinam a realizar a avaliação automática de interfaces. Como exemplo de *Builders* cita-se o Builder Xcessory, o X-Designer, o PV-Wave, a AVS/Express, a XVT - DS, as ADK's, a Galaxy, o RtWorks, o Dialog Editor e o Next Interface Builder Prototyper. Maiores informações sobre estas ferramentas são encontradas em [Sastry, 95].

4.3.3.1 *O BUILDER X-DESIGNER*

O X-Designer é um GUI Builder para os ambientes OSF/Motif e Windows. Apresenta facilidade de manipulação e aprendizagem. A interface é desenvolvida diretamente na tela, depois, o X-Designer gera código C++, com definições de classe reutilizáveis e completamente portável para a plataforma C. Uma única interface pode ser criada para vários ambientes, é gerado código para X/Motif e para Microsoft Foundation Class Library (MFC). O X-Designer assegura a portabilidade de seu código sem o uso de bibliotecas0 proprietárias.

4.3.3.2 *O BUILDER Xcessory*

É uma ferramenta orientada a objetos que possibilita a manipulação direta dos objetos d0a interface e a geração do código fonte em C, C++ ou código Java ocorre automaticamente, em um ambiente de desenvolvimento de interfaces do modo WYSYWYG. Possui uma barra extensível com objetos de interface que são utilizados para a construção das interfaces. As interfaces são desenvolvidas usando widgets Motif ou componentes ViewKit ou componentes definidos pelo usuário ou ainda classes Java.

4.3.3.3 *A FERRAMENTA PV-Wave*

O PV-WAVE acelera o desenvolvimento de aplicações, através de funções integradas de gerenciamento de dados, visualização de dados e funções integradas de matemática e estatística, como também tecnologia para visualizaç00ão de dados da *Web*. O PV-WAVE possui uma *toolkit* para o processamento de imagens, com um conjunto de filtros, efeitos e ferramentas para suprir as necessidades principais de manipulação de imagens.

O PV-WAVE acelera o desenvolvimento de aplicativos. É formado por uma linguagem de quarta geração, que devido a sua estrutura, reduz em 80% o tempo de codificação. As linguagens Fortran e C são suportadas e integradas às funções de gerenciamento dos dados. Possui um ambiente do tipo "aponta e clica", com objetos

reusáveis, proporcionando um ambiente de trabalho que permite a rápida criação de interfaces. O PV-WAVE possui um módulo chamado "*Navigator*", onde é possível construir interfaces padrão, acessando facilmente uma grande variedade de componentes interativos e com a vantagem de as interfaces serem compatíveis com UNIX, Windows e OpenVMS.

4.3.3.4 *A FERRAMENTA AVS/Express*

A ferramenta AVS/Express é um ambiente de desenvolvimento de aplicativos multi-plataformas, que apresenta ampla capacidade de desenvolver aplicativos em combinação com atuais componentes tecnológicos. A ferramenta AVS/Express foi desenvolvida para criar aplicativos que requerem visualizações avançadas e características especiais de apresentação. Usando um ambiente de programação orientada a objetos visuais, chamado *Visual Network Editor*, é possível rapidamente prototipar e construir aplicativos com gráficos interativos. Os benefícios deste ambiente de desenvolvimento são a facilidade de construção e manutenção de aplicações multi-plataformas e a integração de componentes tecnológicos para interface do usuário, gráficos, imagens e visualizações.

4.3.3.5 *A FERRAMENTA XVT DEVELOPMENT SOLUTION FOR C AND C++*

A ferramenta XVT caracteriza-se por apresentar um avançado suporte para desenvolvimento de aplicativos orientados a objetos. Possui uma ferramenta de desenvolvimento visual que orienta o desenvolvedor interativamente durante o processo de construção de aplicativos orientados a objetos, incluindo todos os passos de arranjo e codificação da interface e do programa. Este suporte acelera a produção do código, tornando-o reusável e de fácil manutenção, oferecendo soluções para todos os sistemas baseados em janelas.

O XVT suporta as características das GUI's tais como janelas, menus, controles e caixas de diálogo e também atributos específicos da plataforma e dos objetos de GUI's.

Esta abordagem assegura a manutenção da aparência e do comportamento nativos da interface, alta performance, baixo custo e inter-operalidade com outros aplicativos.

4.3.3.6 *FERRAMENTAS PARA DESENVOLVIMENTO ENTRE-PLATAFORMAS (ADK's)*

Os *Application Development Kits* (ADK's) são essenciais para aplicações que devem ser portáveis. A aparência e o comportamento ("look and feel") comum através de diferentes plataformas podem ser criados usando um simulador de API (*Application Program Interface*), que adiciona uma camada de software entre a aplicação e o ambiente ou plataforma nativa. Esta camada providencia uma interface programada comum através de todas as plataformas existentes. Um API faz chamadas diretamente da biblioteca nativa de objetos gráficos de interação. Um subconjunto de funcionalidades, as quais são comuns para todas as plataformas existentes, são tipicamente implementadas. Aplicativos que podem ser desenvolvidos com o apoio dos ADK´s, ente outros, cita-se:

- Painéis de controle para máquinas de uso diário;
- Aplicações administrativas (preenchimento de formulários);
- Monitoramento de plantas e processos, sistemas de segurança críticos (aviões, trens);
- Simuladores aéreos;
- Aplicações CAD/CAM;
- Suporte à diagnóstico e decisões (projetos arquitetônicos);
- Gerenciamento de sistemas de informação (bancos);
- Sistema de informações geográficas - GIS (indústrias da defesa).

De maneira a apoiar os requisitos destes aplicativos, as ferramentas ADK providenciam um ambiente robusto e integrado específicos para:

- Desenvolver objetos de interface específicos, com múltiplas visões e animações;
- Especificar a manipulação de dados dinâmicos (de fontes/processos externos);
- Acesso a base de dados;
- Implementar apresentação de dados e atualizações em tempo real;

- Implementar a tomada de decisões, a aprendizagem baseada em regras e a lógica *fuzzy*;
- Manipular as entradas e a manipulação do usuário, com suporte para controle de tempo de execução entre o usuário.

4.3.3.7 *O PACOTE GALAXY (VISIX SOFTWARE)*

Esta é uma biblioteca de interfaces do usuário em C, escrita em um estilo orientado a objetos. Este pacote inclui um *Builder WYSIWYG*, onde as ferramentas são bastante completas e relacionadas, e os itens da interface do usuário têm extensiva abstração (por exemplo, eles possuem um dialogo para setar se um botão será *OK, Cancel, Apply*). Os objetos podem ser posicionados relativamente uns aos outros (não possuindo posição absoluta na tela).

Os erros são manipulados com um suporte de manipulação de exceções e as fontes suportam internacionalização (inclui japonesa) e os dados podem ser formatados (moeda). Algumas características extras incluem a detecção de falta de memória e objetos para texto da linguagem C, com multi-estilos, multi-fontes de texto e gráficos embutidos, listas (planilha de cálculo) e processamento de gráficos.

4.3.3.8 *O AMBIENTE RTWorks*

É uma família de módulos de software independentes, desenvolvidos para adquirir e monitorar em tempo real dados inteligentes, analise de dados, distribuição de dados e mensagens e a apresentação de dados e mensagens. O Rtworks oferece um grande número de estratégias sofisticadas para resolução de problemas, incluindo sistemas baseados em conhecimento, interfaces do usuário do tipo aponta e clica, razões estatísticas e temporais e a habilidade para distribuir uma aplicação sobre uma rede heterogênea. É incluído no Rtworks uma máquina de inferência (RTie) de alta velocidade, a qual é usada para analisar os dados usando objetos, classes, procedimentos e regras.

As telas podem ser criadas na ferramenta DrawProgram, onde existe mais de 60 formatos diferentes de objetos para representar dados, como barras gráficas, controles, *dials* e campos de dados. Estes objetos gráficos têm os seus atributos manipulados dinamicamente, como cor, escala, rotação, movimento, animação entre outros. É um ambiente para a plataforma UNIX.

Como parte de um projeto de pesquisa do Laboratório de Utilizabilidade - LabIUtil que busca pesquisar, projetar e desenvolver uma ferramenta de avaliação automática de interfaces baseada em requisitos ergonômicos, apresenta-se a seguir o estado da arte de *Builders* (sistemas especialistas) que avaliam interfaces gráficas do usuário automaticamente.

4.3.3.9 *KRI/AG - KNOWLEDGE-BASED REVIEW OF USER INTERFACES*

O Kri/ag é um *Builder* conectado diretamente à um UIMS (System Management Interface User) o TeleUse. TeleUse é um ambiente X Window que avalia os arquivos com formato UIL, gerados pelo UIMS. O sistema Kri/ag é capaz de avaliar a interface gerada pelo UIMS. Cada arquivo contém a representação estática da interface. A avaliação é realizada durante o desenvolvimento da interface e fornece informações úteis para que se possa realizar correções no projeto.

a) Estrutura do sistema : o Kri/ag é composto por três elementos: um analisador sintático, uma base de conhecimento e uma base de regras.

- O analisador sintático : tem por função transformar a descrição estática da interface gerada no UIMS em um formato utilizável pelo sistema especialista durante a avaliação.
- A base de conhecimento : contém as regras ergonômicas disponíveis na literatura e o guia de estilo OSF/MOTIF. A base de conhecimento contém cerca de 70 regras ergonômicas (60% de recomendações de guidelines e 40% do guia OSF/MOTIF). As regras ergonômicas são selecionadas em função de sua possibilidade de avaliação. O nível da avaliação é limitada pelo conteúdo dos arquivos que descrevem a apresentação estática da interface. As regras ergonômicas são agrupadas de acordo com os objetos da interface, permitindo que a avaliação seja realizada em diferentes etapas e sob aspectos específicos.

- A base de regras : contém as recomendações ergonomicas existentes nos guias de estilos citados anteriormente.

b) A estruturação das regras ergonômicas : as regras ergonômicas são estruturadas de acordo com uma taxonomia de componentes de interfaces mais utilizados.

c) Os passos da avaliação são os seguintes :

- A descrição da interface : é automaticamente recuperada através dos arquivos de recursos do UIMS conectado ao KRI/AG.
- A avaliação : é em parte guiada pelo avaliador que especifica os elementos que serão avaliados. O sistema avalia as regras ergonômicas referentes aos objetos selecionados. Esta avaliação pode ser realizada durante a concepção da interface.
- Os resultados : são fornecidos na forma de uma lista de comentários que o avaliador pode consultar. São fornecidas também as recomendações ergonômicas que não foram respeitadas.

4.3.3.10 *SYNOP*

O Synop é um *Builder* de avaliação automática da apresentação estática de sistemas de controle em tempo real. No Synop os arquivos binários gerados pelo editor gráfico IMAGIN são tratados, via um módulo específico, para recuperar a descrição da interface.
O Synop é composto por três módulos:

- A interface : entre o Synop e o editor gráfico IMAGIN tem por função interpretar os arquivos binários de descrição da interface a avaliar.
- A base de conhecimentos : é constituída de regras ergonômicas e de meta-regras que permitem selecionar as regras ergonômicas a usar na avaliação. As regras ergonômicas são de nível lexical e sintáxico.
- O módulo de controle : permite modificar a semantica (interface) em função dos erros detectados.

a) Estruturação das regras ergonômicas: as regras ergonômicas são agrupadas por

problemas ergonômicos. A base de recomendações é formada por pacotes de recomendações relativas a um mesmo problema, como por exemplo a legibilidade dos caracteres ou o contraste das cores. Através da ativação de critérios, são selecionados os pacotes que são formados por meta-regras que em conjunto formam a base do meta-conhecimento.

A estruturação da base de metas-regras permite restringir o número de regras executáveis entre as regras efetivamente ativadas. A estruturação permite reduzir os tempos de tratamento e os tempos de seleção das regras ativadas.

b) O proceso de avaliação ocorre através da seguintes fases :

- A descrição da interface : é automaticamente recuperada dos arquivos binários gerados pelo editor gráfico conectado ao Synop.
- A avaliação : é realizada pelo sistema que seleciona as regras e verifica a validade de sua execução.
- Os resultados : são fornecidos na forma de uma lista de comentários que o avaliador poderá consultar. A interface avaliada é melhorada em função dos erros detectados.

4.3.3.11 *A FERRAMENTA CHIMES*

CHIMES (Computer Human Interaction Models) é um *Builder* desenvolvido pela NASA-Goddard Space Flight Center . A ferramenta Chimes foi criada para demonstrar a facilidade da avaliação automática de interfaces homem-computador na perspectiva de guidelines sobre fatores humanos e heurísticas. A meta inicial desta ferramenta é checar objetivamente as características da interface tais como tamanho de botões, rótulos, localização, fontes e o uso de cores em conformidade com as normas. É utilizada na avaliação de interfaces desenvolvidas pelo TAE plus (Transportable Application Environment) sob OSF/MOTIF. Avalia automaticamente a conformidade da interface conforme a norma padrão do OSF/MOTIF e particularmente a respeito das recomendações ergonômicas utilizadas nos objetos da interface. O especialista em interfaces homem-computador é então liberado das tarefas monótonas e demoradas da avaliação para se

concentrar em questões específicas da interface, tais como funcionalidade e comportamento das interações.

A ferramenta Chimes é habilitada a comutar entre dois modos, dependendo se uma interface alfanumérica ou uma interface gráfica está sendo avaliada. No modo modelagem por demanda, usado para avaliar interfaces alfanuméricas, a ferramenta Chimes estima a demanda que a interface colocará sobre um operador experimentado de múltiplos recursos cognitivos. No modo GUI (Graphical User Interfaces) baseado em guidelines, a ferramenta Chimes checa sua conformidade com guidelines de fatores humanos e os requisitos de estilo da toolkit. A abordagem baseada em guidelines para avaliação de GUI é conceitualmente similar à de um corretor ortográfico ou um corretor gramatical para um documento de texto.

Tendo projetado um componente da GUI, o projetista invoca a ferramenta Chimes para checar se o projeto está em conformidade com os vários conjuntos de GUI guidelines. Ainda, a ferramenta Chimes permite ao usuário customizar estas regras e focar sobre componentes particulares da interface (p.ex. botões).

A ferramenta Chimes providencia uma pequena lista de todos os conselhos produzidos durante a avaliação e então o avaliador pode escolher qualquer item da lista e visualizar mais detalhadamente seu conteúdo. Um aviso também é providenciado durante a avaliação, quando um conselho é pertinente para um objeto em particular na interface (um apontador é mostrado perto do objeto em questão).

Depois, o projetista pode fazer mudanças nos componentes da GUI, ou, em alguns casos, pode requisitar que a ferramenta Chimes faça as modificações apropriadas. Após fazer as modificações, o projetista pode reinvocar a ferramenta Chimes para uma reavaliação, e este processo continua até que o projetista esteja satisfeito com o projeto em relação as regras (guidelines).

a) Estrutura da ferramenta Chimes

A ferramenta Chimes é composta por cinco módulos principais: um modulo de

aquisição de informações, um verificador de conformidade , uma base de conhecimento, um gerador de respostas (conselhos) e uma interface do usuário. Abaixo apresenta-se a arquitetura da ferramenta Chimes com seus cinco principais módulos e existem outros três módulos suplementares : a biblioteca de projeto, que contém os exemplos de projeto que verificaram as regras ergonômicas, o módulo que permite imprimir os relatórios da avaliação e um módulo que permite detectar mudanças realizadas na interface.

- O módulo de aquisição de informações : recupera a descrição da interface (através do arquivo de recursos do TAE) e transforma estas informações em domínios de entrada para a avaliação.

- O módulo verificador de conformidade : possui várias funções. Recupera as informações do módulo de aquisição, prepara a base de conhecimento, converte as informações em fatos e inicia a avaliação.

- A base de conhecimentos : contém as regras que representam o projeto de avaliação. As regras estão em um nível léxico e sintáxico.

- O gerador de respostas (conselhos) : apresenta o resultado da avaliação, na forma de uma lista de conselhos referente a cada objeto da interface avaliado.

- A interface : permite o diálogo entre o avaliador e o sistema, gerencia a avaliação e apresenta os resultados.

b) A estruturação das regras ergonômicas

As regras ergonômicas utilizadas na ferramenta Chimes não são especialmente estruturadas, estão todas em um mesmo nível na base de conhecimento.

c) Observações sobre a ferramenta Chimes

As investigações iniciais mostraram que a ferramenta Chimes foi mais rápida do que os especialistas em IHC na avaliação de características da interface, tais como: número de

diferentes objetos por tela, tamanho dos objetos, estilos de linhas, uso de cores e a consistência destes através das múltiplas telas.

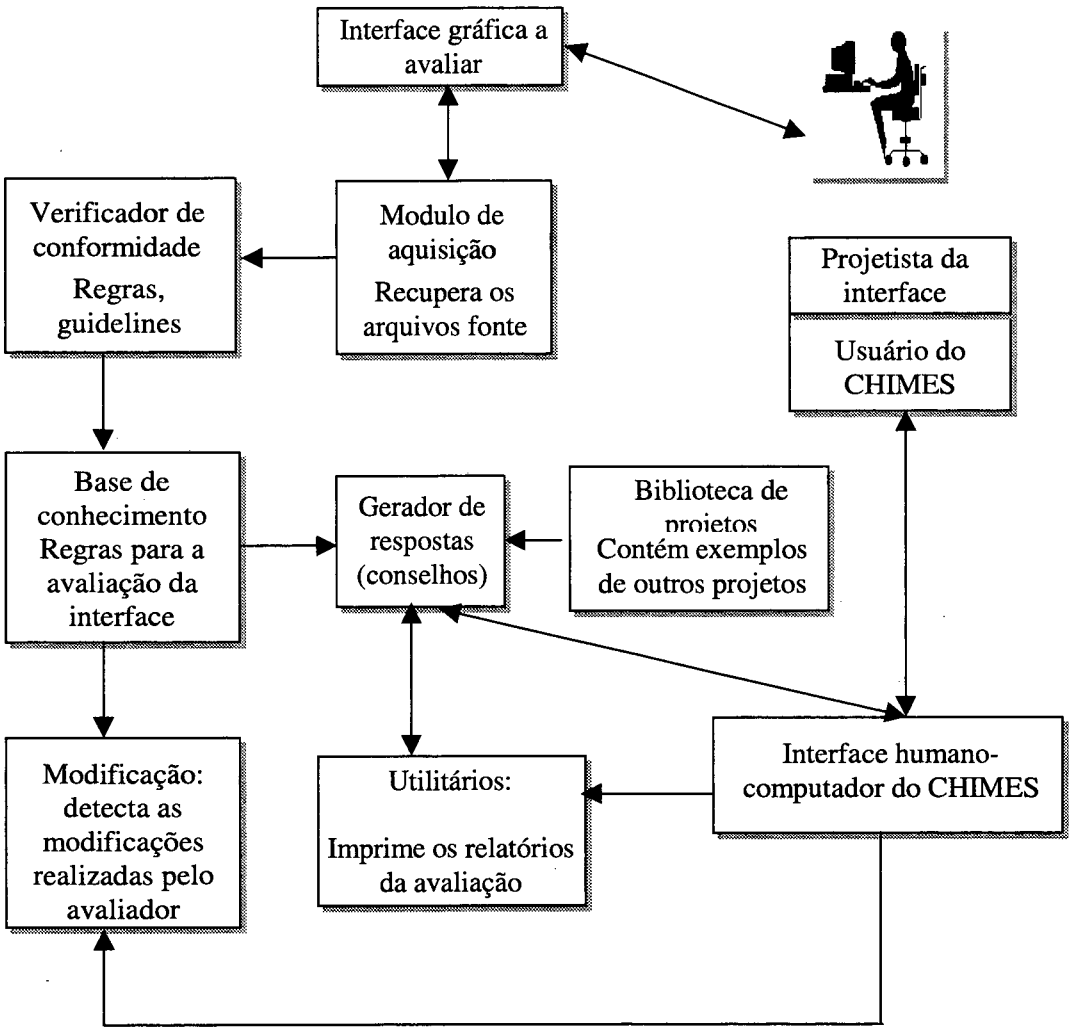

Figura 4.1 : Esquema gráfico da arquitetura do Chimes

4.3.4 OS SISTEMAS DE GERENCIAMENTO DE INTERFACES DO USUÁRIO - UIMS

UIMS é uma ferramenta ou série de ferramentas integradas, que servem para dar suporte ao desenvolvimento do projeto, da interface e da aplicação abaixo dela. Auxiliam na especificação, projeto, construção de protótipos, implementação, execução, avaliação, modificação e manutenção de interfaces [Hix, 90].

Apresentam um editor interativo para desenvolvimento de GUI's, acrescidos de uma linguagem de alto nível com a capacidade de definir e especificar o comportamento dos objetos da interface. O nível desta linguagem de especificação de comportamento é maior que no ambiente C, com uma sintaxe simplificada para que programadores com pouca experiência possam utilizar. Opcionalmente, também é providenciado um conjunto de aplicações chamadas "rotinas de conveniência" que são de um nível mais alto que as tradicionais bibliotecas de programação. Juntos, estes utilitários permitem especificar o comportamento da interface, as respostas da aplicação, a ajuda, a substituição dos valores default, etc. Os UIMS têm utilitários que:

- Compreendem a especificação do comportamento dos objetos de interação e traduzem estes para o domínio da *toolkit*;
- Geram os arquivos modelo (make files);
- Compilam o código da aplicação junto com adicionais bibliotecas que podem ser necessárias;
- Criam o arquivo executável;
- Em tempo de execução, lê os arquivos de dados nos quais a apresentação da interface foi salva e implementa-os na tela.

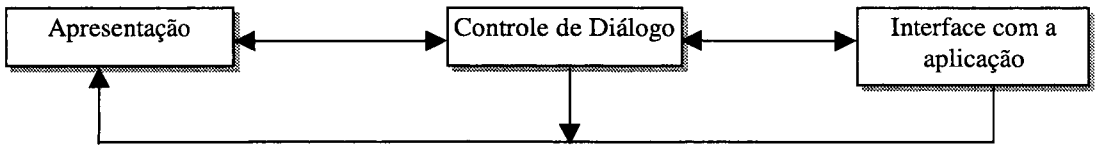

Figura 4.2 : Modelo abstrato de um UIMS . Modelo de Seeheim [Green, 85]

O ponto forte dessas ferramentas é que elas permitem que o projetista veja a interface gerada da mesma maneira que o usuário e também permite que ele ligue corretamente os objetos à suas rotinas específicas. Desse modo, percebe-se que os UIMS são bons para apresentar elementos estáticos. Por outro lado, caso as ferramentas não apresentem os objetos de interação que o projetista necessita, ele terá que criá-los através de código, e além disso, UIMS são deficitários com o comportamento de objetos dinâmicos, sendo eles controlados por responsabilidade do projetista. Isso requer que os objetos e seus comportamentos sejam programados em linguagens comuns. Geralmente as

ferramentas de apresentação também não separam entre a dependência de mídia e o núcleo funcional.

Um UIMS também pode gerar código para construir uma versão da interface que irá executar independente da ferramenta. Como exemplos temos o UIM/X e o TeleUSE. Maiores informações sobre estas ferramentas são encontradas em [Sastry, 95].

4.3.4.1 *A FERRAMENTA UIM/X*

O UIM/X é uma ferramenta de desenvolvimento visual de terceira geração, com uma avançada funcionalidade para manipulação do layout da interface, através de um sofisticado sistema de edição visual, com características de interatividade, que adicionam lógica e comportamento para a interface do usuário durante o projeto. O UIM/X contém um interpretador C++ embutido, permitindo ao desenvolvedor produzir a interface dentro dos padrões de outras linguagens de programação, sem no entanto abandonar o uso da ferramenta. As modificações na GUI podem ser realizadas em tempo de execução, permitindo ganhos de produtividade e evitando o consumo de tempo com os processos de "compilar - editar - debugar".

O UIM/X oferece um editor visual para adicionar comportamento aos objetos de interação da interface. O Editor de Conexões habilita o desenvolvedor a graficamente conectar componentes e adicionar sofisticados comportamentos através de simples técnicas visuais, tais como arrastar e soltar. O UIM/X permite definir graficamente hierarquias de classe de objetos de interação, com definição de herança, encapsulamento e polimorfismo. Este modelo visual evidencia a propriedade de reutilização destes objetos.

4.3.4.2 A FERRAMENTA TeleUSE

Com a ferramenta TeleUSE , utiliza-se apenas um conjunto de código fonte para desenvolver, portar e manter as aplicações nos ambientes UNIX e Windows, incluindo os componentes da interface, as mensagens entre estes componentes e os componentes da aplicação. Os arquivos da aplicação são completamente compatíveis, e é permitido

modificar e adicionar novas características em uma plataforma e imediatamente aplicar estas mudanças em outra plataforma.

O TeleUSE/Win apresenta uma biblioteca de objetos de interação baseado no Motif, a biblioteca "WinTif", capaz de renderizar elementos de GUI's com aparência e comportamento iguais ao Motif, WindowsNT ou Windows95. Um componente da interface é então mostrado no Windows como um objeto da plataforma Windows e como um elemento do UNIX na plataforma UNIX.

O TeleUSE/Win oferece um completo conjunto de widgets com estilo Windows, com características de fonte e cores manipuláveis, tornando o aplicativo herdeiro do comportamento e da aparência do ambiente Windows. O TeleUSE inclui ainda um módulo especial com específicos widgets Windows que o desenvolvedor pode utilizar em seus aplicativos, tais como barra de ferramentas, caixa de atribuição, botão de variação e botões de comando pré-rotulados.

O TeleUSE/Win apresenta poderosos utilitários para apoiar a portabilidade do código da aplicação, oferecendo aos desenvolvedores maneiras simples de portar aplicativos da plataforma UNIX para o Windows. As características principais do desenvolvimento orientado a objetos, como abstração, encapsulamento, herança e polimorfismo, a geração de classes C++ e a criação de componentes reutilizáveis são suportados pelo TeleUSE, tornando o código fácil de ser criado, reutilizado e corrigido.

4.3.4.3 *A FERRAMENTA SL-GMS*

O SL Graphical Modeling System (SL-GMS), é um poderoso sistema de desenvolvimento de gráficos dinâmicos, apresentando um conjunto de distintas mas integradas ferramentas de desenvolvimento, bibliotecas de objetos e componentes de interface para monitorar e desenvolver sistemas de controle, automação de processos, monitoração de redes de computadores e controle de tráfego inteligente. Apresenta a representação e a visualização dinâmica dos dados. O projetista pode definir e animar objetos ou cenas de objetos. Usando um editor de desenhos, o projetista cria interfaces com objetos padronizados, então um comportamento dinâmico é ligado a cada objeto,

permitindo a avaliação de sua funcionalidade, característica interessante para desenvolvimento de sistemas de tempo-real. O pacote SL-GMS é formado pelas seguintes ferramentas:

- O SL-GMSDraw Graphical Model Editor é utilizado pelos desenvolvedores ou usuários finais para construir elementos gráficos dinâmicos, que são usados junto com uma interface gráfica de algum tipo, serve para Windows/MFC e UNIX/Motif;
- O SL-GMS ActiveX Control habilita o SL-GMS com gráficos de funcionalidade dinâmica que podem ser acessados de qualquer aplicativo ActiveX, tais como Visual Basic ou Internet Explorer;
- O SL-GMS Code Generator para Java, produz classes Java que definem uma simbologia complexa para gráficos dinâmicos. Para grandes sistemas integrados SL-GMS esta simbologia permite alta performance em uma web ou intranet com aplicação Java cliente;
- O SL-GMS Native Run-Time Library é uma biblioteca de classes e funções que podem ser embutidas ou linkadas dentro de um sistema de controle para gerenciar uma interface gráfica do usuário. O SL-GMS providencia toda a funcionalidade necessária para apresentar a monitoração dinâmica dos gráficos e controlar as telas conectadas aos dados da aplicação específica. O C/C++ API providencia muitas funções para manipular objetos, acessar dados e customizar comportamentos.
- SL-GMS Network Management and Geographic Mapping Library é uma biblioteca especial para aplicações que necessitam gerenciamento de rede e funcionalidades para manipular elementos (mapas) geográficos, tais como zoom, mapas em camadas.

4. 4 CONCLUSÕES

As ferramentas para desenvolvimento de GUI's têm se tornado imprescindíveis no trabalho de criação de aplicativos dos mais diversos tipos. Estas ferramentas permitem ao desenvolvedor criar a aparência da interface simplesmente arrastando objetos de interação de uma paleta de objetos e posicionando-os sobre a interface (janela). Elas possibilitam que com uma quantidade muito pequena de programação de baixo nível seja produzida grande parte do aplicativo, liberando os projetistas para tarefas mais específicas como os processos e os relacionamentos dos objetos da interface.

As ferramentas atuais de desenvolvimento de GUI's, refletem a tendência para a reutilização do esforço de desenvolvimento para uma variedade de domínios de aplicativos, sistemas operacionais, plataformas de hardware e tipos de usuários. Cada um destes tipos de reusabilidade é um exemplo de independência, porque permitem que um mesmo aplicativo possa ser usado sobre muitas plataformas por muitos usuários e permitem também que um mesmo usuário use muitos aplicativos sobre muitas plataformas. Para permitir o desenvolvimento de GUI's para muitos aplicativos, para muitas plataformas e para muitos usuários, estas ferramentas devem possuir muitas funcionalidades e independências, tornando difícil que se agrupe todas estas características em uma única ferramenta.

Através da análise prática em laboratório e de documentos relacionados às ferramentas apresentadas neste capítulo, pode-se perceber a falta de recursos auxiliares para a especificação do layout dos objetos de interação durante a fase de concepção da interface. Verificou-se a carência de características operacionais automáticas nas ferramentas, do tipo, avaliação automática de concordância à requisitos ergonômicos da interface que está sendo produzida, gerenciamento automático de posição de objeto na interface, validação automática de mnemônicos e teclas de atalho e outros. Encontrou-se deficiência em recursos encontrados em ambientes gráficos como aqueles para a manipulação de figuras, formas geométricas e desenhos. Tipo zoom, rotação, redimensionamento, preenchimento.

No próximo capítulo propõe-se um conjunto de recursos que buscam minimizar as deficiências encontradas nas ferramentas analisadas e também considerações sobre o ambiente de autoria, buscando especificar uma ferramenta com características de usabilidade em sua interface, que ofereça um conjunto de objetos de interação com especificações de aparência e comportamento de acordo com critérios ergonômicos permitindo efetivamente o desenvolvimento de interfaces ergonômicas.

CAPÍTULO V

5. PROPOSTAS DE RECURSOS ERGONÔMICOS PARA AMBIENTES DE AUTORIA E BIBLIOTECAS DE OBJETOS DE INTERAÇÃO REUTILIZÁVEIS

Atualmente, os usuários de sistemas informatizados possuem uma grande quantidade de recursos tecnológicos a sua disposição. Estes recursos, vão desde equipamentos providos de hardware de última geração, conectados em rede e à *Internet*, e chegam a pacotes de software que automatizam todas as tarefas e controlam os principais processos presentes no cotidiano. Com a evolução da utilização do computador nas atividades humanas e, em particular, nas organizações, a preocupação geral entre os técnicos e usuários é a utilização inteligente e eficiente dos sistemas de computadores. Este potencial tecnológico fica frequentemente difícil de ser explorado devido a baixa qualidade das interfaces humano-computador presentes nos sistemas informatizados. A interface têm como papel principal realizar a interação entre o usuário e o aplicativo, possuindo portanto, vários requisitos que devem ser contemplados visando assegurar que a interface será agradável ao uso e efetivamente permitirá a realização da tarefa a que se destina.

Apresentou-se nos capítulos anteriores, estudos buscando a aplicação de metodologias e ferramentas, especialmente na área da ergonomia de interfaces, para projetar e desenvolver interfaces humano-computador em conformidade com padrões ergonômicos. Alguns dos objetivos destes estudos são concernentes à padronização da aparência, comportamento e aumento da qualidade ergonômica dos objetos de interação da interface, o projeto do diálogo entre usuário e aplicativo, o controle da interação pelo usuário e a documentação do sistema. A padronização ergonômica tem um importante papel no aperfeiçoamento da usabilidade destes sistemas, melhorando a consistência da interface e dos componentes. Neste sentido, no estudo realizado sobre ferramentas que se destinam a desenvolver interfaces para o usuário, verificou-se que estas, buscam principalmente criar automaticamente o código da interface projetada, assegurando consistência do código, das mensagens, reduzindo erros (*bugs*), preocupando-se em criar

um modelo padrão para a aparência e para o comportamento dos objetos (*widgets*) que compõem a interface.

Segundo Martin [Martin,97] *" As atuais ferramentas para construção de GUI habilitam os desenvolvedores à construir aplicações mais facilmente que a alguns anos atrás. Entretanto, as ferramentas não asseguram que as aplicações são automaticamente bem projetadas".*

Propor qualidades ergonômicas para uma ferramenta de desenvolvimento e para bibliotecas de objetos de interação, têm como objetivos principais, permitir a afinada integração do projeto da interface, dos princípios da usabilidade, das metodologias de desenvolvimento de software em um *framework* formalizado ergonomicamente. As propostas apresentadas a seguir, surgiram de três fontes distintas, entretanto relacionadas:

- A primeira, um CheckList de validação de objetos de interação, desenvolvido pelo LabIUtil[1], que unifica um conjunto de recomendações ergonômicas, oriundas de diversos autores, para os principais objetos de interação que compõem as interfaces gráficas do usuário. As propostas originadas do checklist LabIUtil serão marcadas com o símbolo: 🕮 ;
- A segunda, através do estudo do estado da arte das ferramentas para desenvolvimento de interfaces, que permitiu verificar e validar as características das diferentes abordagens e filtrar funcionalidades julgadas interessantes. As propostas originadas de outras ferramentas serão marcadas com o símbolo: 💻;
- A terceira, são empirismos, algumas, adaptações de funcionalidades encontradas em sistemas informatizados, outras, observou-se válidas durante discussões em laboratório de pesquisa – LabIUtil/UFSC, onde julgou-se utilizáveis em um ambiente para construção de interfaces. As propostas originadas destes empirismos serão marcadas com o símbolo: ✪.

[1] LabIUtil – Laboratório de Utilizabilidade da Informática – Centro Tecnológico – Departamento de Informática e Estatística – Universidade Federal de Santa Catarina – Florianópolis – Brasil. http://labiutil.inf.ufsc.br

5.1 QUADRO DE PROPOSTAS PARA CAIXAS DE FERRAMENTAS

Além de propor uma biblioteca de objetos de interação, lista-se algumas considerações relacionadas com recursos auxiliares, bibliotecas de rotinas, repositórios de objetos e recursos pertinentes ao ambiente de desenvolvimento, essenciais para a manipulação da ferramenta e a criação de interfaces com usabilidade certificada. Os recursos auxiliares listados adiante, são recursos observados em diversas ferramentas, algumas com outra finalidade senão construir interfaces, mas julgadas interessantes para um ambiente de autoria:

- Uma ferramenta de manipulação de bitmaps, para criação de ícones;
- Uma ferramenta gráfica que permita a edição dos objetos de interação existentes em suas bibliotecas. (ex. Iconizar botões, modificar aparência e forma de objetos);
- Uma ferramenta para verificar interativamente a ortografia de rótulos, menus e itens de menu;
- Uma ferramenta assistente de digitação, para correção e colocação de maiúscula/minúscula em qualquer elemento de texto utilizado na interface;
- Uma ferramenta para avaliação ergonômica automática da interface produzida;
- Uma ferramenta para monitorar a criação de teclas de atalho e mnemônicos evitando que comandos diferentes possuam conjuntos de teclas iguais;
- Uma ferramenta para monitorar a criação de menus e seus derivados,
- Uma ferramenta para formatação da fonte utilizada nos rótulos, menus, campos de dados/texto.
- Uma ferramenta para gerenciar a especificação do controle de diálogo;
- A ferramenta deve fornecer linhas guias, utilizadas como elementos auxiliares no layout da interface;
- Os arquivos executável e modelo (*make files*) devem ser criados automaticamente.
- A ferramenta deve possuir recursos de auxilio a criação de arquivos de ajuda;
- Uma ferramenta para a criação de hipertextos, hiperlinks de acordo com o padrão

Web

- Uma ferramenta para editar as características gráficas dos objetos de interação (padrões, imagens, textura, cor, preenchimento);
- Um editor interativo de layout, com características WYSIWYG;
- Desenvolver o código da interface independente de plataforma, permitindo a portabilidade da interface;
- A ferramenta deve permitir a criação de barras de ferramentas personalizadas;
- A ferramenta deve possuir recursos para criar ou personalizar novos motivos para o cursor do mouse;
- No ambiente de autoria, é interessante existirem recursos para a manipulação de figuras, formas geométricas e desenhos. Tipo zoom, rotação, redimensionamento, preenchimento;

5.2 BIBLIOTECAS RESIDENTES NA FERRAMENTA

Uma ferramenta para desenvolvimento de interfaces deve oferecer para o desenvolvedor, objetos de interação prontos para o uso e adaptáveis aos diversos tipos de interfaces que podem ser criadas, agilizando o processo de concepção e reduzindo a carga de trabalho do desenvolvedor. Algumas bibliotecas foram selecionadas e são listadas a seguir, outras são apresentadas juntas com os respectivos objetos de interação:

- A ferramenta deve possuir bibliotecas com modelos editáveis de caixas de mensagem;
- A ferramenta deve possuir bibliotecas com modelos editáveis de mensagens;
- A ferramenta deve possuir bibliotecas com elementos de desenho reutilizáveis;
- A ferramenta deve possuir bibliotecas com botões de comando rotulados com bitmaps com vários motivos e com as funções mais comuns encontradas em aplicativos, tipo OK, Cancelar, Salvar;
- A ferramenta deve possuir bibliotecas de ícones que possam ser utilizados em

botões, caixas de mensagens e barras de ferramentas;

- A ferramenta deve possuir bibliotecas com painel de menu horizontal, vertical, em cascata, pop-up, pulldown e dropdown;
- A ferramenta deve ter bibliotecas de objetos de interação relacionados com a internet;
- A ferramenta deve disponibilizar bibliotecas editáveis de janelas ou formulários, com as funções básicas de aplicativos, tais como abrir, salvar, imprimir;
- A ferramenta deve possuir bibliotecas com modelos de gráficos (de barras, pizza...) e suporte para a sua manipulação, permitindo anexar este recurso aos aplicativos gerados;
- A ferramenta deve dispor de bibliotecas de objetos de interação com aparência 3D;
- A ferramenta deve apresentar bibliotecas de objetos com funções específicas para manipulação de banco de dados (*dbnavigator*);
- A ferramenta deve disponibilizar bibliotecas com exemplos de interfaces utilizáveis;

5.3 A INTERAÇÃO DO PROJETISTA COM A FERRAMENTA

A interação entre desenvolvedor e ferramenta deve ocorrer da forma mais eficaz possível, permitindo o desenvolvimento de interfaces dentro de padrões de usabilidade, em um ambiente concebido sob estes padrões. Estas considerações buscam contemplar a usabilidade do ambiente proposto:

- Durante o desenvolvimento da interface, é interessante a ferramenta apresentar feedback, mostrando ao desenvolvedor alternativas de concepção;
- A ferramenta deve possuir um gerenciador de projeto, facilitando a navegação entre os componentes lógicos do sistema;
- A ferramenta deve possuir tutoriais de treinamento e programas de ajuda para auxiliar os desenvolvedores;
- A ferramenta deve permitir modificar a configuração de sua interface de acordo

com a experiência do desenvolvedor;

- A ferramenta deve permitir configurar sua barra de ferramentas de acordo com o tipo de interface que está sendo criada ou com os controles mais utilizados;
- Os ícones da ferramenta devem possuir aparência que permita o fácil reconhecimento da sua função;
- A ferramenta deve disponibilizar as funções de desfazer, refazer e repetir as últimas ações realizadas;
- Os objetos de interação disponíveis, devem ser agrupados conforme sua lógica de utilização. (p.ex. Caixa de agrupamento + *checkbox* + lista + botão)

5.4 A BIBLIOTECA DOS OBJETOS DE INTERAÇÃO

A biblioteca de objetos de interação especificados para a ferramenta, são classificados como elementos mínimos que uma ferramenta de desenvolvimento de interfaces deve oferecer, permitindo a criação de interfaces em uma ampla faixa aplicativos. Estes objetos compõem a *Lista de Verificação LabIUtil*, os quais são agrupados em classes, conforme mostrado a seguir:

5.4.1 PAINÉIS DE CONTROLE

5.4.1.1 *Formulário*: É uma metáfora para janelas baseado na idéia dos formulários e fichas em papel. Permite a entrada de informação predefinida através de campos de dados rotulados.

- A área útil do formulário deve ser dividida em "uma área de identificação", "uma área de entrada e saída de informação" e "uma área de controle", onde somente é permitido aos objetos específicos para estas áreas se localizarem;
- O formulário deve possuir, por padrão, na barra de título, objetos de controle automaticamente anexados, como botão minimizar, maximizar, fechar, redimensionador de janela;

 - O formulário deve permitir a seleção da posição de alinhamento do título da janela, na barra de títulos; (centralizado ou a esquerda);

 - O formulário deve fornecer automaticamente um dispositivo de rolamento quando a interface que está sendo desenvolvida assim exigir (algoritmo);

 - O formulário deve monitorar a densidade de objetos de interação em cada interface, buscando manter a clareza, a percepção e o reconhecimento da informação (utilizando algoritmos de layout);

 - O formulário deve monitorar o espaçamento entre os objetos de interação, facilitando a legibilidade da interface (grid de layout);

 - O formulário deve formatar um tamanho mínimo e máximo para o redimensionamento da janela;

 - O formulário deve promover o distanciamento dos objetos de interação em relação a borda em distâncias definidas pelas normas e guias de estilo;

 - Em um mesmo formulário, objetos "devem se perceber" visando auto-alinhar-se e auto-distanciar-se (eqüidistantes);

 - O formulário deve monitorar o posicionamento de objetos do tipo lista, não permitindo a colocação de outros objetos abaixo ou acima desta, evitando que outros objetos ou informações sejam encobertos;

 - O formulário deve possuir grid milimétrico para posicionamento de objetos;

 - O formulário deve possuir uma função de fixar (chavear) o formato da interface não permitindo a movimentação dos objetos de interação, quando esta for considerada concluída;

 - O formulário deve apresentar a posição do objeto de interação em relação ao eixo horizontal e vertical, na forma de um mostrador digital, com recurso de incremento e decremento do valor relativo da posição;

5.4.1.2 *Caixas de Mensagem*: São janelas auxiliares que informam o usuário da ocorrência de erros do sistema ou problemas com o aplicativo ou que fornecem ajuda contextual ao usuário na realização da tarefa em foco, dando sugestões, avisos concretos e respostas para questões simples. Também apresentam-se como janelas auxiliares que devem ser usadas para receber uma resposta do usuário à uma questão ou apresentar

mensagens que chamam a atenção do usuário para uma operação que pode ter consequências indesejáveis ou se o sistema não está seguro que o processo em andamento se encerrará normalmente.

- A caixa de mensagem deve ser dividida em "uma área de identificação", "uma área de símbolo", "uma área de mensagem" e a "uma área de controle";
- É obrigatório a existência de barra de título em todos os objetos da classe Painéis de Controle, para identificar o aplicativo e ou processo;
- As caixas de mensagem de erro, por padrão, devem apresentar obrigatoriamente um botão de ajuda, um botão de repetir/cancelar e um botão de confirmação e serem modais;
- As caixas de mensagem de informação, por padrão, devem apresentar obrigatoriamente um botão de ajuda e um botão de confirmação do recebimento da mensagem;
- As caixas de mensagem de alerta, por padrão, devem apresentar obrigatoriamente botões para aceitar/desistir /cancelar e um botão de ajuda;
- As caixas de mensagem de confirmação, por padrão, devem apresentar obrigatoriamente um botão para confirmar (Sim), um botão para cancelar (Não) e um botão de ajuda;
- As caixas de mensagem de informação, por padrão, devem apresentar obrigatoriamente um botão para confirmar e um botão de ajuda;
- As caixas de mensagem de status, por padrão, devem apresentar obrigatoriamente um botão para abortar o processamento em andamento;
- As caixas de mensagem de erro, de alerta devem ser acompanhadas de sinal sonoro quando da sua apresentação;
- As caixas de mensagem, por padrão, devem ser apresentadas no centro da tela;

5.4.2 CONTROLES COMPLEXOS

5.4.2.1 *Menus*: Corresponde a uma organização das diversas características que vão fazer parte da interação com menus. Seu estilo de interação simplificado reduz a margem de erros ao mesmo tempo que estrutura a tarefa guiando o usuário novato ou intermitente.

Não exige do usuário treinamento e memorização de complexas seqüências de comando. Corresponde a um conjunto de títulos de menu, arranjadas horizontalmente, localizadas no topo da janela, logo abaixo da barra de título. Tem como função separar em conjuntos as opções disponíveis na aplicação. Cada opção permite acesso a um painel de menu.

- As opções da barra de menu, por padrão, devem ser separadas automaticamente por 04 espaços em branco;
- Os grupos de opções da barra de menu, por padrão, devem ser automaticamente limitadas à 08 itens;
- Os itens de menu devem ser formatados, por padrão, na forma: 1ª letra maiúscula e demais letras em minúscula;
- A ferramenta deve separar automaticamente com linhas separadoras, os grupos de itens de menu quando este atinge o valor máximo permitido (09);
- As opções de menu devem possuir três tipos de aparência: "disponível, não_disponível e selecionada";
- As opções de menu devem automaticamente ser alinhadas e distanciadas da borda do painel ou da barra de menu;
- A ferramenta deve permitir anexar dinamicamente, informações, às opções de menu, relacionadas à tarefa; (ex.: menu pop-up - Salvar figura como <nome_figura>)
- A ferramenta deve inserir nos painéis de menu os elementos padronizados (▸) que apresentam outros painéis;
- A ferramenta deve monitorar a seleção de teclas de atalhos (*hotkeys*) e/ou mnemônicos para opções de menu e itens de menu, evitando repetições.

5.4.2.2 *Tabela*: Objeto usado na apresentação de uma coleção de itens que são organizados em forma de tabela (linhas e colunas), suportando a seleção de um ou mais itens.

- A ferramenta deve apresentar recursos para a configuração da aparência gráfica de tabelas, as linhas, colunas, separadores, cabeçalhos e células;
- A ferramenta deve realizar a formatação da fonte utilizada nos cabeçalhos e células da tabela;
- A ferramenta deve formatar a aparência dos valores das células de tabelas, diferenciando valor selecionado e valor não selecionado;
- A ferramenta deve *chavear* os campos de cabeçalhos das tabelas, evitando alteração em tempo de execução;
- A ferramenta deve permitir configurar como se realizará a navegação entre as células, verticalmente e horizontalmente;
- O comprimento padrão de apresentação de uma tabela deve ser de no máximo oito linhas;
- Quando a tabela possui mais que oito itens (linhas), a ferramenta deve disponibilizar dispositivos de rolagem;
- A ferramenta deve formatar a largura da tabela de acordo com o maior item e/ou no máximo com 40 caracteres.

5.4.2.3 *Lista de Seleção*: Uma lista de seleção é um objeto usado na apresentação de uma coleção de itens, suportando a seleção de um ou mais itens. O uso de listas de seleção é apropriado na entrada de dados numéricos, alfanuméricos, booleanos, horários, de calendário e gráficos sempre que os valores são conhecidos e a quantidade de valores possíveis é maior do que 8 itens. São classificadas em listas de seleção simples, descortinamento e seleção múltipla.

- O conjunto "rótulo / lista de seleção" deve ser representado por um objeto único;
- O comprimento padrão da lista de seleção, quando aberta, deve apresentar no máximo oito itens;
- A ferramenta deve dispor de recursos (separadores) que permitam criar agrupamentos de itens nas listas de seleção;
- Quando a lista possui mais que oito itens, a ferramenta deve disponibilizar dispositivos de rolagem para a apresentação dos itens encobertos;

- A lista de seleção deve possuir, como padrão, a largura mínima de 20 caracteres;
- A ferramenta deve formatar a largura da lista de seleção de acordo com o maior item da lista e/ou no máximo com 40 caracteres;
- Os itens da lista de seleção, por padrão, devem ser obrigatoriamente alinhados a esquerda;
- O objeto lista de seleção deve apresentar visualmente diferenciados, os itens selecionados, daqueles não-selecionados;
- A aparência padrão da lista de seleção é fechada, apresentando apenas um item default;
- A lista de seleção é um objeto que deve permitir a seleção de apenas um item;
- A lista de seleção é um objeto que não deve permitir a inclusão de itens.

5.4.2.4 *Caixa de Combinação*: Representa a agregação entre um campo de dados e uma lista de seleção. Permite em tempo de execução a manipulação dos dados da lista.

- O conjunto "rótulo / caixa de combinação" deve ser representado por um objeto único;
- Por padrão a caixa de combinação, quando aberta, deve apresentar no máximo oito itens;
- A caixa de combinação deve, por padrão, possuir a largura mínima de 20 caracteres;
- A ferramenta deve formatar a largura da caixa de combinação de acordo com o maior item da lista e/ou no máximo com 40 caracteres;
- Os itens da caixa de combinação devem ser obrigatoriamente alinhados a esquerda;
- O objeto caixa de combinação deve apresentar visualmente diferenciados, os itens selecionados dos itens não-selecionados;
- Quando o conjunto de itens for maior que oito, deve ser providenciada barra de rolamento;

- A caixa de combinação é um objeto que deve permitir a inclusão/ edição de itens;

5.4.2.5 *Paleta*: São painéis modais que apresentam um conjunto de controles para a seleção de valores gráficos como cores, padrões ou imagens.

- A ferramenta deve apresentar recursos para definição e inclusão de novas elementos à paleta;
- O conjunto "rótulo / paleta" deve ser representado por um objeto único;
- A ferramenta deve permitir que a paleta se desloque para qualquer lugar na tela ou seja anexada à uma barra de ferramentas;
- O conjunto de cores contidas na paleta deve ser de um mesmo padrão CMYK, Pantone, RGB ou outros;
- A ferramenta deve permitir a modificação no modo de apresentação da paleta.

5.4.2.6 *Calendário*: Objeto de interação em forma de tabela, onde se fixam os dias do ano correspondentes a determinados acontecimentos. Contem uma área de apresentação de dia e uma área de seleção de dia, mês e/ou ano.

- O conjunto "rótulo / calendário" deve ser representado por um objeto único;
- A ferramenta deve possuir bibliotecas de calendários prontos para o uso;
- A ferramenta deve possuir bibliotecas de calendários de vários países;
- O objeto calendário deve informar ano / mês / dia e nome do dia da semana;
- O calendário deve possuir configurações que permitam seu deslocamento para qualquer lugar na tela ou ser anexado à uma barra de ferramentas;
- O calendário deve possuir configurações que permitam as principais operações de edição com seus dados (copiar, imprimir, exportar);
- O calendário deve ser compatível com o calendário do sistema, permitindo visualizar datas em outros períodos.

5.4.3 GRUPOS DE CONTROLES

5.4.3.1 *Grupo de Botões de Rádio*: São usados para setar atributos ou valores, sendo a maneira mais comum de apresentar escolhas mutuamente exclusivas. O usuário realiza uma seleção clicando em um botão de radio do grupo, consequentemente desabilita o item previamente selecionado. Cada botão é rotulado de acordo com sua função e o grupo generaliza o conjunto.

- O conjunto "rótulo / botão de rádio" deve ser representado por um objeto único;
- A ferramenta deve possuir o objeto botão de rádio sem rótulo;
- A ferramenta deve permitir que o objeto botão de rádio seja agrupado com outros objetos, quando a interface assim exigir; (ex. botão de rádio + caixa de agrupamento (configuração de tcp/ip no windows));
- O conjunto "rótulo / botão de rádio" deve se distanciar automaticamente da borda do formulário ou da caixa de agrupamento, a uma distância de 4 pixels;
- Os botões de rádio devem, quando agrupados, se auto-distanciar (eqüidistantes) em valores definidos em normas ou guias de estilo;
- A ferramenta deve monitorar automaticamente o alinhamento de grupos de botões de rádio, selecionando o alinhamento vertical ou horizontal;
- O conjunto "rótulo / botão de rádio" devem ser formatados com a mesma altura (tamanho);
- A ferramenta deve gerenciar o layout de grupos de botão de rádio, visando facilitar a leitura (Z);

5.4.3.2 *Grupo de Caixas de Atribuição*: Representa um conjunto de caixas de atribuição (*check boxes*), que permitem a seleção de opções e valores de forma não exclusiva. São utilizados para apresentar os vários atributos de objetos os quais o usuário pode modificar.

- O conjunto "rótulo / caixa de atribuição " deve ser representado por um objeto

único;

- A ferramenta deve monitorar automaticamente o alinhamento de grupos de caixas de atribuição, selecionando o alinhamento vertical ou horizontal;
- A ferramenta deve possuir o objeto caixas de atribuição sem rótulo;
- A ferramenta deve permitir que o objeto caixas de atribuição seja agrupado com outros objetos, quando a interface assim exigir
- O conjunto "rótulo / caixa de atribuição" deve estar posicionado a uma distancia padrão de 4 pixels da borda do formulário ou da caixa de agrupamento;
- A ferramenta deve monitorar o alinhamento de grupos de caixas de atribuição, selecionando o alinhamento vertical ou horizontal;
- As caixas de atribuição devem, quando agrupados, se auto-distanciar (eqüidistantes) em valores definidos em normas ou guias de estilo;
- A ferramenta deve gerenciar o layout de grupos de caixa de atribuição, visando facilitar a leitura (Z);
- O conjunto "rótulo / caixa de atribuição" devem ser formatados com a mesma altura (tamanho);

5.4.3.2 *Grupos de Botões de Comando*: Quando várias ações podem ser realizadas em uma janela, esta geralmente apresenta um conjunto de botões para dispará-las. O conjunto de botões é disposto abaixo ou em uma coluna à direita, quando relativos a um mesmo conjunto lógico de dados.

- A ferramenta deve agrupar botões em um número máximo de 7 itens;
- O objeto botão quando agrupado deve automaticamente distanciar-se (eqüidistantes) e alinhar-se na vertical ou horizontal;
- Grupos de botões de comando devem localizar-se quando na horizontal, na base do formulário e/ou abaixo do conjunto de dados e quando na vertical, devem localizar-se à direita dos objetos e/ou alinhados com a borda direita do formulário;
- Um botão, dentre os botões do conjunto, deve ser indicado como default. A

posição deste botão deve ser a mais alta se os botões estiverem alinhados na vertical e mais a esquerda se estivem alinhados na horizontal.

5.4.4 CONTROLES SIMPLES

5.4.5.1 *Botão de Comando*: É um objeto de controle utilizado para disparar uma ação descrita pelo rótulo. Um botão contém um rótulo textual, gráfico (ícone) ou misto.

- O conjunto "rótulo / botão de comando" deve ser representado por um objeto único;
- O conjunto "ícone / botão de comando" deve ser representado por um objeto único;
- O objeto botão de comando deve ajustar-se proporcionalmente na largura e no comprimento ao tamanho do rótulo;
- O objeto botão de comando deve ter sua aparência diferenciada conforme o estado default, habilitado, não-habilitado, selecionado e acionado;
- A rótulos textuais de botões de comando, obrigatoriamente devem ser disponibilizados mnemônicos ou hotkeys;
- O objeto botão deve permitir acrescentar imagens, figuras ou bitmaps como rótulo;
- A ferramenta deve associar às funções dos botões, teclas com características funcionais padrão;
- Um botão de comando deve distanciar-se (eqüidistantes) quando posicionado próximo à outro botão de comando;
- Devido às diversas formas dos botões, a ferramenta deve permitir a especificação de "áreas sensitivas" também em diferentes formatos.

5.4.5.2 *Interruptor*: Objeto de controle (checkbox) com características de interruptor, usado para setar atributos de comandos.

- O conjunto "rótulo / interruptor" deve ser representado por um objeto único;
- Quando o interruptor é o objeto default ele deve apresentar-se diferenciado visualmente;

5.4.5.3 *Controle Deslizante*: É um objeto de controle utilizado para calibrar comandos que não possuem restrição de precisão. Geralmente possui um rótulo na forma de uma escala de valores definida pelo contexto da aplicação.

- A ferramenta deve permitir que o controle deslizante seja acionado via mouse e teclado;
- O objeto Controle Deslizante deve ajustar-se proporcionalmente ao tamanho da escala.

5.4.5.4 *Barra de Rolagem*: Corresponde a uma barra retangular geralmente posicionada a direita e/ou abaixo de uma janela. É composta por botões em suas extremidades, rotulados com setas indicando o sentido do movimento e um botão (slider) deslizante que indica a progressão e direção do movimento. Permitem ao usuário visualizar qualquer parte da área de trabalho disponibilizada pelo aplicativo.

- A ferramenta deve possuir bibliotecas com barras de rolagem com orientação vertical e horizontal;
- A ferramenta deve possuir bibliotecas com barras de rolagem simples (composta por slider, e botões de progressão ▲ ▼)com orientação vertical e horizontal, para serem utilizadas em conjunto com os objetos lista de seleção, lista de combinação e outros;
- A ferramenta deve possuir bibliotecas com barras de rolagem compostas (composta por slider, e botões de progressão ▲ ▼ , e botões para progressão por páginas ⬇ ⬆) com orientação vertical, para serem utilizadas em conjunto com os objetos formulário e janelas;

- A barra de rolagem deve permitir o acionamento com mouse e teclado.

5.4.5.4 *Caixa de Atribuição*: É um objeto composto por dois botões rotulados com setas e um campo de dado. Permitem a apresentação e seleção de opções ou valores predefinidos de forma não exclusiva. Possui um rótulo textual que expressa o valor ou efeito da seleção. Os botões rotulados com setas servem para incrementar ou decrementar os valores das opções.

- O conjunto "rótulo / caixa de atribuição" deve ser representado por um objeto único;
- A ferramenta deve possuir o objeto caixa de atribuição sem rótulo, para ser utilizado em conjunto com outros objetos(caixa de agrupamento);
- O objeto caixa de atribuição deve permitir a formatação de preenchimento, tamanho, apresentação (selecionado, não selecionado, indisponível);
- O conjunto "rótulo / caixa de atribuição" deve se distanciar automaticamente da borda do formulário ou da caixa de agrupamento, a uma distância de 4 pixels;
- A caixa de atribuição deve ser formada pela união do objeto campo de dado e setas up-down formando um objeto único;
- As caixas de atribuição devem, quando agrupadas, se auto-distanciar (eqüidistantes) em valores definidos em normas ou guias de estilo.

5.4.5.6 *Botão de Variação*: Corresponde aos botões rotulados com setas, utilizados nas caixas de atribuição, têm como função incrementar ou decrementar os valores associados ao campo de dado. São usados quando os itens de uma lista são apresentados ordenadamente.

- O Botão de Variação deve possuir funções de agregação com o objeto campo de dados, de maneira a produzir um objeto único.

5.4.5.7 *Cursor de Mouse*: Corresponde as diversas formas assumidas pelo cursor do dispositivo de apontamento de acordo com as funções disponíveis na aplicação. O estilo padrão é uma seta apontando para esquerda para seleção de objetos, uma ampulheta para informar processamento, uma barra vertical para seleção de componentes de textos , etc.

 - A ferramenta deve disponibilizar os recursos (aparência) padrão para o mouse (barra vertical, seta, ampulheta, link) de acordo com a função do objeto de interação.

5.4.5 CAMPOS DE ENTRADA DE DADOS

5.4.5.1 *Campo de Dados*: Corresponde a uma área retangular que permite a introdução e manipulação de caracteres na forma textual através de recurso de edição uni-linear. São utilizados recursos do sistema para a manipulação dos dados tipo seleção por mouse e edição por teclado.

 - O conjunto "rótulo / campo de dados" deve ser representado por um objeto único;

 - O objeto campo de dados deve monitorar o distanciamento em relação a borda da janela ou da caixa de agrupamento;

 - O objeto campo de dados deve ser formatado apenas com fontes sem serifa;

 - A ferramenta deve possuir o conjunto “rótulo / campo de dados / rótulo ” para quando um campo de dados apresenta unidade de medida, peso, volume ou valor;

 - A ferramenta deve formatar a aparência do objeto campo de dados, como bordas, cor do fundo, fonte, alinhamento;

 - A ferramenta deve formatar o tamanho do campo de dados de acordo com seu valor lógico e no máximo em 40 caracteres;

 - O objeto campo de dados deve permitir a formatação dos dados que serão apresentados;

 - A ferramenta deve possibilitar a formatação dos métodos de navegação entre os campos de dados;

- A ferramenta deve monitorar o alinhamento e a separação de conjuntos de campos de dados;
- A altura dos campos de dados deve ser de no mínimo 12 pixels.

5.4.5.2 *Campo de Texto*: Corresponde a uma área retangular onde o usuário pode digitar e manipular caracteres na forma textual, através de recursos de edição multi-linear. São utilizados recursos do sistema para a manipulação dos dados tipo seleção por mouse e edição por teclado.

- O conjunto "rótulo / caixa de texto" deve ser representado por um objeto único;
- A largura mínima de apresentação de um campo de texto deve ser de 04 linhas;
- A ferramenta deve permitir anexar ao campo de texto barras de rolagem, vertical e horizontal, quando necessário;
- Ao objeto campo de texto deve ser fornecido apenas fontes com serifa, para facilitar a leitura;
- O comprimento do campo de texto deve ser configurável e apresentar no máximo 50 caracteres;
- A ferramenta deve possibilitar a formatação e edição do texto inserido em uma caixa de texto;
- Por padrão o texto inserido em um campo de texto deve ser alinhado pela esquerda;
- O campo de texto deve realizar automaticamente a quebra de palavras, mantendo monossílabos inteiros.

5.4.5.3 *Linha de Comando*: Campo de entrada específico para digitação de caracteres que disparam algum tipo de processamento.

- O conjunto "rótulo / linha de comando" deve ser representado por um objeto único;

- O comprimento de uma linha de comando deve ser configurável e possuir no máximo 40 caracteres;
- A ferramenta deve disponibilizar para a linha de comando meios para reutilizar os últimos comandos inseridos.

5.4.6 MOSTRADORES DE DADOS ESTRUTURADOS

5.4.6.1 *Lista de Apresentação*: Objeto utilizado para apresentar um conjunto de itens para seleção ou visualização. Pode ser composta de uma área para apresentar elementos de texto ou elementos gráficos. Pode opcionalmente possuir barras de rolagem vertical e horizontal, possibilitando mostrar diferentes visões dos elementos da lista.

- O conjunto "rótulo / lista de apresentação" deve ser representado por um objeto único;
- Deve ser permitida a configuração da aparência física da lista de apresentação (as linhas separadoras, cor de fundo, cabeçalhos);
- A lista de apresentação deve acrescentar barras de rolagem vertical e horizontal, quando necessário;
- A lista de apresentação deve permitir a seleção do alinhamento dos itens (direita ou esquerda);
- A formatação da lista de apresentação deve permitir que itens ocupem mais de uma linha.

5.4.6.2 *Tabela*: Objeto de interação utilizado para agrupar elementos de informação organizados em linhas e colunas.

- A ferramenta deve ter recursos de configuração da aparência física da tabela, as linhas, colunas, células e cabeçalhos (cor de fundo);
- A ferramenta deve permitir a formatação da fonte utilizada nos cabeçalhos e nas células da tabela;

- A ferramenta deve definir tipos de separadores (linhas e colunas com formato diferenciado) para blocos de informação relacionadas;
- A ferramenta deve possuir formas diferenciadas para a apresentação dos dados, selecionados ou não;
- A ferramenta deve permitir a formatação dos cabeçalhos, com aparência diferenciada dos campos de dados;
- A ferramenta deve permitir configurar como se realizará a navegação entre as células, verticalmente e horizontalmente.

5.4.6.3 *Locução*: Apresentação narrativa de um dado ou informação.

- A ferramenta deve possuir recursos para a gravação e a execução de frases contínuas;
- A ferramenta apresentar controles para o ritmo da apresentação, volume e balanço;
- Toda locução deve ser apresentada em conjunto com texto.

5.4.6.4 *Listas Simples*: Objeto de interação destinado a apresentação de um conjunto de itens enfileirados verticalmente um em cada linha.

- O conjunto "rótulo / lista simples" deve ser representado por um objeto único;
- A ferramenta deve dispor de recursos que permitam criar agrupamentos de itens nas listas simples;
- Quando a lista simples possui mais que oito itens, a ferramenta deve disponibilizar dispositivos de rolagem;
- A ferramenta deve ter recursos de configuração da aparência física da lista simples;
- A lista simples deve possuir largura mínima de 20 caracteres como padrão;

- A ferramenta deve formatar a largura da lista simples de acordo com o maior item da lista e/ou no máximo com 40 caracteres;
- Os itens da lista simples devem ser obrigatoriamente alinhados a esquerda ou a direita;
- O objeto lista simples deve apresentar visualmente diferenciados, os itens selecionados dos itens não-selecionados;
- A aparência padrão da lista simples é sem nenhum item selecionado;
- A lista de seleção é um objeto que deve permitir a seleção de um ou vários itens;
- A lista simples é um objeto que deve permitir a inclusão de itens.

5.4.6.5 *Diagrama*: Representação gráfica de objetos. Mostra hierarquia, processos, valores e suas relações por meio de linhas, esboço.

- A ferramenta deve apresentar acessórios de manipulação de diagramas, tipo lupa ou zoom;
- A ferramenta deve oferecer bibliotecas com objetos gráficos comuns em diagramas (quadrados, elipses, retângulos);
- A ferramenta deve oferecer recursos para formatação dos objetos gráficos;
- A ferramenta deve oferecer barras de ferramentas com linhas, setas, usadas para ligar os elementos do diagrama;
- A ferramenta deve oferecer recursos para o processamento de imagens, com um conjunto de filtros, efeitos e ferramentas para suprir as necessidades principais de manipulação de imagens.

5.4.7 MOSTRADORES DE DADOS SIMPLES

5.4.7.1 *Mostrador de Dados*: São objetos utilizados na apresentação de dados numéricos. São adequados na apresentação de valores dentro de um intervalo definido.

- O conjunto "rótulo / mostrador de dados" deve ser representado por um objeto único;
- A ferramenta deve permitir editar a aparência do objeto mostrador de dados.

5.4.7.2 *Mostrador Analógico*: É um objeto de interação graduado numericamente através de um limite inferior e um limite superior com intervalos regulares, munido de uma agulha que aponta o valor dentro de um intervalo (relógio). Adequado na apresentação de valores numéricos oscilantes.

- O conjunto "rótulo / mostrador de dados analógico / rótulo p/ unidades" deve ser representado por um objeto único;
- O conjunto "rótulo / mostrador de dados analógico" deve ser representado por um objeto único;

5.4.7.3 *Mostrador Digital*: É um objeto de interação usado para a apresentação de dados numéricos, dinâmicos ou não, através de dígitos.

- O conjunto "rótulo / mostrador de dados digital" deve ser representado por um objeto único;
- O conjunto "rótulo / mostrador de dados digital / rótulo p/ unidades" deve ser representado por um objeto único;
- A ferramenta deve possuir recursos de configuração da aparência dos dados apresentados, tipo cor, forma.

5.4.7.4 *Escala*: Objeto de interação onde o usuário introduz valores por meio do ajuste de um indicador em uma posição específica de uma linha graduada.

- O conjunto "rótulo / escala" deve ser representado por um objeto único;
- O conjunto "rótulo / escala / rótulo" deve ser representado por um objeto único (qdo. a escala possui unidade);

- A ferramenta deve permitir que a escala seja formatada quanto ao conjunto de valores apresentados;
- A ferramenta deve possuir bibliotecas de escalas na orientação vertical e horizontal;
- A ferramenta deve permitir que a escala seja acionada via mouse e teclado;

5.4.8 ELEMENTOS DE LAYOUT

5.4.8.1 *Layout*: Corresponde à distribuição espacial dos objetos de interação na interface, levando-se em conta o equilíbrio, legibilidade, alinhamento, balanço, etc.

- A ferramenta deve municiar de recursos de gerenciamento do layout da interface durante o seu desenvolvimento, o objeto janela, formulário e caixa de agrupamento, para que estes em conjunto com os demais objetos que compõem a interface, possam se perceber, e a partir daí utilizando regras pré-definidas, se auto-alinhem, se auto-distanciem e se auto-localizem;
- Na interface da ferramenta, os atributos de layout dos objetos de interação devem ser apresentados dinamicamente, somente no momento da sua utilização.

5.4.8.2 *Fundo*: Atributo dos objetos utilizado para modificar sua aparência através da utilização de cores.

- A ferramenta deve possuir bibliotecas de cores que possam ser aplicadas nos objetos de interação que fazem parte da interface. Devem ser conjuntos de cores cromaticamente complementares, para garantir a legibilidade, puras ou saturadas, que apresentem contraste suficiente para assegurar a clareza;
- A ferramenta deve monitorar o uso de cores de acordo com conjuntos pré-estabelecidos de cores para o fundo (dos objetos) e dos rótulos, campos de dados / de texto.

5.4.8.3 *Caixa de Agrupamento*: Separador retangular destinado a marcar um agrupamento dentro da estrutura informacional de um objeto interativo por razões semânticas ou ergonômicas de apresentação.

- O conjunto "rótulo / caixa de agrupamento" deve ser representado por um objeto único;
- A ferramenta deve permitir editar o formato (irregular) da caixa de agrupamento, facilitando o layout;
- A caixa de agrupamento deve monitorar o posicionamento, número de elementos, alinhamento e distanciamento (da borda) dos objetos que são inseridos no seu interior;
- A ferramenta deve permitir a formatação da borda da caixa de agrupamento.

5.4.8.4 *Linha Separadora*: Linha reta gráfica com o objetivo de separar, agrupar ou isolar um ou mais objetos interativos que fazem parte de um objeto composto.

- Deve ser disponibilizada uma biblioteca com diversos tipos de linhas prontas para o uso, com características gráficas variadas;
- A ferramenta deve permitir que a linha separadora tenha suas características gráficas configuradas (espessura, cor, comprimento).

5.4.9 ELEMENTOS DE INFORMAÇÃO

5.4.9.1 *Mensagens*: Informação mostrada ao usuário, em forma de texto, em resposta a um evento não esperado, a uma situação onde algumas vezes podem acontecer situações indesejáveis, ou mostrando uma informação adicional de um processo que foi completado. Existem vários tipos de mensagens, tais como de erro, de ajuda, informacional, de status, de advertência e de confirmação.

- A ferramenta deve permitir a formatação do texto (fonte) que será apresentado nas mensagens;
- A ferramenta deve padronizar o formato para as mensagens, monitorando o número de palavras, comprimento da linha e largura do parágrafo.

5.4.9.2 *Barra de Status*: Área que mostra mensagens informativas ou fornece feedback, sobre o estado de uma tarefa interativa.

- A ferramenta deve apresentar recursos para a criação de barras de status.

5.4.9.3 *Indicador de Progressão*: Indicador visual ou textual, que mostra o progresso de um processamento demorado.

- O conjunto "rótulo / indicador de progressão" deve ser representado por um objeto único;
- A ferramenta deve possuir o conjunto, indicador de progressão gráfico e indicador de progressão numérico;
- A ferramenta deve permitir formatar a aparência do indicador de progressão (Fundo, cor da barra);
- A ferramenta deve permitir formatar o indicador de progressão numérico em termos de fonte, posição, alinhamento e aparência.

5.4.9.4 *Bolha de Ajuda*: É uma pequena janela local, com um texto descritivo e é apresentada quando o usuário move o cursor sobre um objeto de controle.

- A ferramenta deve possuir o recurso da inclusão de bolhas de ajuda nos objetos de interação;
- Deve ser permitida a configuração da aparência das bolhas de ajuda,

independente da configuração do sistema;

- A ferramenta deve permitir configurar o tempo mínimo ou máximo de apresentação da bolha de ajuda.

5.4.9.5 *Rótulo*: São elementos identificadores textuais (títulos) de janelas, caixas de diálogo, listas, tabelas, campos de dados, botões e cabeçalhos. O rótulo identifica e/ou descreve todo tipo de objeto e ação associada e convida o usuário à interação.

- A ferramenta deve disponibilizar para os rótulos, apenas fontes que não apresentam serifa;
- A ferramenta deve apresentar o objeto rótulo independente de qualquer objeto de interação;
- A ferramenta deve permitir rótulos compostos por somente letras maiúsculas (siglas);
- A ferramenta deve permitir a retirada do rótulo daqueles objetos formados por "rótulo + objeto de interação";
- O rótulo, apesar de existir em conjunto com outros objetos de interação, deve ser independente, em termos de formatação.

5.5 CONCLUSÕES

Percebeu-se com este estudo a importância dos métodos de engenharia de software, das técnicas e ferramentas para análise, projeto e implementação, como elementos auxiliares no desenvolvimento dos complexos sistemas informatizados atuais. Da mesma maneira, o projeto de interfaces do usuário, as quais têm se tornado muito mais complexas devido às chamadas GUI (Interfaces gráficas do usuário), deve ser apoiado por ferramentas e metodologias resultantes de pesquisas relativas à interação humano-computador.

O desenvolvimento de interfaces é ainda uma atividade que consome grande quantidade de tempo. Com as ferramentas disponíveis atualmente, cada objeto de interação

da interface necessita ser selecionado e distribuído na interface pelo desenvolvedor. Estas ferramentas não apresentam condições de que recomendações ergonômicas sejam eficientemente implementadas e também, a consistência entre interfaces e aplicativos é difícil de ser atingida devido à deficiências nas próprias ferramentas.

Verificou-se durante a concepção deste trabalho, que a Engenharia de Software e a Ergonomia de Interfaces são áreas que se sustentam e ao mesmo tempo se complementam quando, por um lado, são utilizadas técnicas que permitem gerenciar o processo de desenvolvimento de sistemas informatizados e, por outro, oferecem-se critérios ergonômicos que quando contemplados, asseguram a usabilidade da interface e padronizam a aparência e o comportamento dos objetos de interação da interface.

As propostas apresentadas anteriormente, a criação automática do código da interface, assegurando consistência do código, das mensagens, reduzindo erros (*bugs*), preocupando-se em criar um modelo padrão para a aparência e para o comportamento dos objetos (*widgets*), representam características oferecidas nas ferramentas estudadas e, consideradas apropriadas ao desenvolvimento de interfaces ergonômicas. A presença destas funcionalidades no ambiente de autoria proposto confirma a necessidade de integração das diferentes perspectivas observadas neste estudo.

Acrescentou-se à estas, funcionalidades para o ambiente de desenvolvimento tais como, linhas guias auxiliares, zoom, gerenciador de layout, oriundas de ferramentas gráficas utilizadas na criação de desenhos artísticos e técnicos, que usam estes recursos para aprimorar a qualidade do trabalho desenvolvido, refinando propriedades de layout, legibilidade e clareza, características básicas da usabilidade.

Os objetos de interação disponibilizados pelo ambiente, também receberam um tratamento ergonômico, buscando a padronização visual e comportamental e o aperfeiçoamento da usabilidade das interfaces produzidas, através da manutenção da consistência entre interfaces e seus componentes, permitindo aos futuros usuários a interação natural e fácil.

Algumas funções automatizadas que foram propostas, como a verificação interativa da ortografia de rôtulos, menus e itens de menu, a ferramenta assistente de digitação, a ferramenta para monitorar a criação de teclas de atalho e mnemônicos e a ferramenta para avaliação ergonômica automática da interface, destinam-se à auxiliar o desenvolvedor na concepção da interface e assegurar que os requisitos ergonômicos projetados para a interface estão sendo efetivamente implementados.

5.6 TRABALHOS FUTUROS

Espera-se obter recursos técnicos e administrativos que permitam dar sequência à estes estudos, implementando na prática as propostas aqui apresentadas que são passíveis de desenvolvimento, confirmando as conclusões apresentadas, colaborando com a comunidade científica que pesquisa os diversos campos de atuação da ergonomia e cooperando com aqueles desenvolvedores que buscam incorporar características ergonômicas às interfaces implementadas.

6. BIBLIOGRAFIA

[Ameritech, 93] AMERITECH GRAPHICAL USER INTERFACE , **Standards and Design Guidelines –** Disponível dia 20/11/98, em http://www.ameritech.com

[Bass & Coutaz, 91] BASS, L. & COUTAZ, J., **Developing Software for the User Interface**, SEI – Series in Software Engineering, Addison-Wesley, 1991.

[Shneiderman, 92] SHNEIDERMAN, B., **Designing the User Interfaces - Strategies for effective human-computer interaction,** Addison-Wesley Massachusetts, USA, 1992.

[Bullinger et. al., 96] BULLINGER H.J.; FAHNRICH K. P.; WEISBECKER A., **GENIUS: Generating Software-Ergonomic User Interface.**. Fraunhofer - Institut fur Arbeitswirtschaft und Organisation (IAO) – International Journal of Human-Computer Interaction 8(2) pg. 115 – 144, 1996.

[Bass et. al., 92] BASS, L. et al., **A Metamodel for the RunTime Architecture of an Interactive System**, SIGCHI Bulletin, vol. 24, N° 01, pg. 32-37, 1992.

[Bastien & Scapin, 93] BASTIEN, J. M. C & SCAPIN, D. L., **Ergonomic Criteria for the Evaluation of Human-Computer Interfaces.** (Report No. 156). Rocquencourt, France: Institut National de Recherche en Informatique et en Automatique, 1993.

[Cardelli, 85] CARDELLI, L., **Squeak: A Language for Communication With Mice.** Computer Graphics, pg. 199-204. Proceedings Siggraph. San Francisco, CA, 1985.

[Cybis, 94] CYBIS, W. A., **A identificação de Objetos de Interfaces Homem-Computador e seus Atributos Ergonômicos.**. Tese de Doutorado, Engenharia de Produção – UFSC, Florianópolis, 1994.

[Cybis, 97] CYBIS, W. A., **Ergonomia de Interfaces Homem-Computador.** Apostila para o Curso de Pós-graduação em Engenharia de Produção - UFSC, Florianópolis, 1997.

[Dix, 93] DIX, A. et.al., **Human-Computer Interation.** Prentice hall International UK, Limitet. I^{a} edição, 1993.

[Diaper, 89] DIAPER, D., **Task Analisys for Human-Computer Interaction.** University of Liverpool - Ellis Horwood limited, 1989.

[Eberts, 94] EBERTS, R. E., **User Interface Design.** Prentice-Hall, Inc. New Jersey, USA. 1994.

[Gulliksen, 97] GULLIKSEN, J, et.al., **Analisys of Information Utilization (AIU).** Center for humam-computer studies (CMD), Uppsala University, Sweden. International journal of human-computer interaction, 9 (3), pg. 255 – 282, 1997.

[Green, 86] GREEN, M., **Report on Dialogue Specification Tools.** In User Interface Management Systems, Springer-Verlag, pg. 9-20, 1986.

[Hartson, 89] HARTSON, H. R. & HIX, D., **Toward Empirically Derived Methodologies and Tools for Human-Computer Interface Development**, Int. Journal Man-Machine Studies, pg. 447-494, 1989.

[Helander, 97] HELANDER, M. G. et.al., **Handbook of Human-Computer Interation.** Edited by North-Holland - 2ª edição, 1997.

[Hix, 90] HIX, D., **Generations of User-Interface Management Systems.** IEEE Software, pg. 77-89, 1990.

[IHC, 98] **I workshop sobre fatores humanos em sistemas computacionais** - IHC98 – Maringá - Pr. PUC, Rio de janeiro, 1998.

[Ilog Views, 97] ILOG VIEWS 2.4., **Getting Started with High-Performance Graphics.** User manual – Illog, 1997.

[ISO 9241, 99] International Standards Organisation. ISO 9241., **Ergonomic Requirements for Office Work with Visual Display Terminals –**
Part 11 – Usability Statements; International Standard, 1999.

[Laird & Soraiz, 98] LAIRD, C. & SORAIZ, K., **An Overview of Today's Best-bet GUI Toolkits.** 1998 – Disponível dia 12/06/99 em Http://www.sunworld.com/swol-03-1998.

[Landay & Myers, 97] LANDAY, J. A. & MYERS, B. A., **Interactive Sketching for the Early Stages of User Interface Design.** Human-Computer Interaction Institute School of Computer Science Carnegie

Mellon University , ACM, 1997. Disponível dia 02/02/1999, em http://www.cs.cmu.edu:8001/Web/People/landay/home.html

[Leite, 97] LEITE, J. C., **Projeto de Interfaces de Usuário** – Monografia em Ciência da Computação, no. 11, Departamento de Informática, PUC-Rio de janeiro, 1997.2

[Liesemberg, 96] LIESEMBERG, H., **Por quê as Interfaces são Importantes**, Projeto Xchart, DCC/UNICAMP/ IMECC, São Paulo, 1996.

[Lim, 96] LIM, K. Y., **Structured Task Analysis: An Instantiaton of the MUSE Method for Usability Engineering.** Interacting With computers vol 8 n° 1, pg. 31-50, 1996.

[Mark, 85] MARK, G., **Report on Dialogue Specification Tools, UIMS**, Computer Graphics, pg. 9 – 20, Springer-Verlag, 1985.

[Mazzola, 98] MAZZOLA, V. B., **Apostila de Engenharia de Software**, Curso de Pós-Graduação em Ciência da Computação, UFSC, Florianópolis,1998.

[Microsoft, 95] MICROSOFT., **Windows Style Guide, the Windows Interface Guidelines** - A Guide for Designing Software, Microsoft Corporation, 1995.

[Minasi, 94] MINASI, M., **Segredos de Projetos de Interface Gráfica com o Usuário.** – Infobook - 1ª edição, 1994.

[Moran, 81] MORAN, T., **The Command Language Grammar: A Representation for the User Interface of Interative Computer Systems**, International Journal of Man-Machine Studies, 15, 3 - 50, 1981.

[Motif, 93] MOTIF., **OSF/Motif Style Guide** (Open Software Foundation) – Cambridge, MA 02142, 1993.
Disponível dia 12/09/98, em http://www.autarch.loni.ucla.edu

[Myers, 96a] MYERS, B. et.al., **The Amulet Environment: New model for Effective User Interface Software Development.**
HCI Institute – Carnegie Melllon University - Pittsburgh, PA, 1996.
Disponível dia 10/12/1998, em http://www.cs.crm.edu/~amulet

[Myers, 96b] MYERS, B., **UIMSs, Toolkits, Interface Builders.** – HCI Institute – Carnegie Melllon University , Pittsburgh, PA, 1996.

[Myers, 96c] MYERS, B. et.al., **Overview of the Amulet User Interface Toolkit.** HCI Institute – Carnegie Melllon University, 1996. Disponível dia 02/02/1999, em http://www.cs.crm.edu/~amulet

[Perlman, 90] PERLMAN, G., **A Vision of Universal Functionality for Tomorrow's User Interfaces.** Department of Computer and Information Science The Ohio State University Columbus, Ohio 43210, 1990.

[Perlman, 89] PERLMAN, G., **GUI Toolkits: What are Your Options? User Interface Development.** – Software engineerind Institute, Carnegie Mellon University, SEI-CM-17-1.1, 1989.

[Powell, 90] POWELL, J. E., **Designing User Interface.** – Microtrend Books. 1ª edição, 1990.

[Presmann, 95] PRESMANN, R., **Engenharia de Software**. Mackron Books, 3ª edição, 1995.

[Rosson & Carroll, 97] ROSSON, M. B. & CARROLL J. M., **Integrating Task and Software Development for Object-Oriented Applications.** Virginia Polytechnic Institute and State University Blacksburg, VA 24061, ACM, 1997.

[Salber et. all., 94] SALBER, D., et. al., **Extendeing the Scope of PAC-Amadeus to Cooperative Systems**, Proceedings of CSCW, pg. 22-26, 1994.

[Sastry, 95] SASTRY, L., **Graphical User Interfaces Development Tools.** Sponsored by the Joint Information Systems Committee. Visualization Group, Informatics Departament, DRAL. Engineering and Physical Sciences Research Council, 1995.

[Shackel, 91] SHACKEL, B. & RICHARDSON, S., **Usability - Context, Framework, Definition, Design and Framework**, In: Human Factors for Informatics in Usability, Cambridge University Press, 1991.

[Shafer, 95] SHAFER, D., **WindowBuilder Pro/V Tutorial.** 1995.

[Shneiderman, 92] SHNEIDERMAN B., **Designing the User Interface: Strategies of Effective Human-Computer Interaction**. Second edition, Addison-Wesley Publishing Company, 1992.

[Smith & Mosier, 86] SMITH S.L. & MOSIER J.N., **Guidelines for Designing User Interface Software.** Final Report Number MTR-10090 ESD-TR-86-278, The Mitre Corporation, 1986.

[Sommerville, 96] SOMMERVILLE, I., **Software Engineering**. 5ª edição, Addison-Wesley Publishing Company, 1996.

[Stork, 94] STORK, A., **Applying a Structured Method for Usability Engineering to Domestic Energy Management User Requirements - MUSE : a Successful Case-Study.** Copenhagen, Denmark: University of Copenhagen, 1994.

[Swan, 98] SWAN T., **Delphi 4 Bible**, IDG Books Worldwide, Inc. First edition, Foster city – USA, 1998.

[Thovtrup & Nielsen, 91] THOVTRUP, H. & . NIELSEN J., **Assessing the Usability of a User Interface Standard**, useit.com Papers and Essays User Interface Standards, 1991.

[Toleman & Welsh, 98] TOLEMAN M. A. & WELSH, J., **Systematic Evaluation of Gesign Choices for Software Development Tools.** – Tecnical Report - The University of Queensland, 1998.

[Valaer & Babb, 97] VALAER, L. A. & BABB, G. R., **Choosing a User Interface Development Tool,** Institute of Electrical and Eletronics Engineers –IEEE – Software, pg. 29 - 39, 1997.

[Vanderdonckt, 96] VANDERDONCKT, J., **Current Trends in Computer Aided Design of User Interface.** In Proc. Of 2th International Workshop on Computer-Aided Design of User Interfaces (CADUI'96), Namur, Belgique, 1996.

As seguintes ferramentas são comentadas e apresentadas nos endereços eletrônicos relacionados:

Artkit	http://www.cc.gatech.edu/
AVS/Express	http://www.avs.com/products/
Brown 3D	http://www.cs.brown.edu/research/graphics/
Dialog Editor	http://www.unix.digital.com/
IDL	http://www.java.sun.com/products/jdk/idl
InterViews	http://www. rd13doc.cern.ch/public/doc/Note10/
Java Toolkit AWT	http://www. java.sun.com/products/jdk/awt/
Motif Widgets	http://www.premier.sco.com/guide/MotifStyleGuide/
NeXTStep	http://www.next.com/
Open Look	http://www.darwinsys.com/
PV-WAVE	http://www.vni.com/products/wave/

Rendezvous.....	http://www.rv.tibco.com/
SL-GMS.....	http://www.sl.com
Silicon Graphics Inventor Toolkit...	http://www-europe.sgi.com/Technology/Inventor/
Smalltalk.....	http://www.smalltalksystems.com/references.htm
SunTools.....	http://www.sun.com/
SUIT	http://www.cs.virginia.edu/suit
TeleUSE.....	http://www.aonix.com/Products/UIMS/uims.html
Visual Basic.....	http://www.microsoft.com/vbasic
Visual Obliq.....	http://www.cc.gatech.edu/
Wind/U.....	http://www.bristol.com/
X WindowSystem.....	ftp://ftp.x.org/contrib/audio/Xaudio/xwindowsystem.html
X/Motif.....	http://www.cms.dmu.ac.uk/
Xt Intrinsics.....	http://www.thinkbank.com/jordan/
LabIUtil.....	http://www.labiutil.inf.ufsc.br (Neste site você encontra informações sobre ergonomia de interfaces, alguns trabalhos publicados na área e o Ergolist, uma ferramenta para avaliação ergonômica de interfaces gráficas.)

7. ANEXO

É apresentado neste anexo uma lista de ferramentas de desenvolvimento de interfaces descritas no artigo "User Interface Software Tools, ACM Transactions on Computer-Human Interaction vol. 2, no. 1, March, 1995. pp. 64-103. de Brad A. Myers". As informações são formatadas na seguinte ordem: **Nome da ferramenta, nome do fabricante, rua, cidade, estado, código postal, telefone, E-mail, preço, plataforma / ambiente e classificação**. Este anexo tem caráter informativo e espera-se que seja útil no processo de seleção de ferramentas para desenvolvimento de interfaces.

Action!, Macromedia, 600 Townsend St., San Francisco, CA 94103 - 415-252-2000, General Number 800-288-4797, $100, Multimedia Toolkit. **Action!**, ExperTelligence, 5638 Hollister Ave #302, Coleta, CA, 93117, (805) 967-1797, $595, Lisp builder for Mac, IB - Mac **ActivAda for Windows**, Thomson Software Products (formerly Alsys), 10251 Vista Sorrento Parkway, Suite 300, San Diego, CA 92121, 800-833-0085 x244, Fax: 619-452-2117, Activada@thomsoft.com, $995, Windows GUI IB. **Actor**, The Whitewater Group, 1800 Ridge Ave, Evanston, IL, 60201, $475, PC/Windows, OO prog lang, Toolkit – PC. **ActiveX Components**, ProtoView 26 Offington Drive Worthing West Sussex, BN14 9PN England chris.geiger@protoview.co.uk Windows, Component Library, Data Explorer, DataTable, TreeViewX and WinX Component Library. **Ada95 GUI Library**, Asterisk Business Solutions, 11130 Kingston Pike, Suite 1-191, Knoxville, TN 37922, info@asterisksolutions.com PC, Mac, Motif, Irix, Solaris -toolkit. **AdaSAGE**, Idaho National Engineering	**ILOG Views**, ILOG Inc., 2005 Landings Drive, Mountain View, CA, 94043, (415) 390 9000, fax: (415) 390 0946, info@ilog.com, $5,000 to $7,500, Unix, OS/2, Windows 3.1, Windows NT cross platform C++ library. **INT Widgets**, INT Corp., 2901 Wilcrest, Suite 300, Houston, TX 77042, 713-975-7434,info@int.com, $1750-$3000, X/Motif, charting and table widgets (toolkit). **InterMAPhics, Gallium Software Inc.** *(formerly Prior Data Sciences)*, 4000-303 Moodie Drive, Nepean, Ontario, CANADA K2H 9R4; (613) 721-0902; $45,000, Sun, DEC, and most Unix platforms, Aeronautics, C&C. UIMS – Unix. **InterViews**, Stanford University, interviews-bugs@interviews.stanford.edu, ftp:interviews.stanford.edu, free, C++/X/Unix, Toolkit/Structured Graphics. **ISA Dialog Manager**, ISA Informationssysteme GmbH, Azenbergstrasse 35, D-70174 Stuttgart, +49/711/22769-15, Fax: +49/711/22769-19, info-idm@isa.de, from DEM 9,000., UNIX/Motif, UNIX/ASCII, VMS/Motif, VMS/ASCII, OS/2, Windows (3.1, NT, '95), Virtual Toolkit, UIMS. **IVTOOLS** , Vectaport Inc. P.O. Box 7141 Redwood City,	**SUIT**, formerly: Randy Pausch, UVA, Dept. Comp. Sci, Thornton Hall, Charlottesville, VA, 22903, suit-admin@virginia.edu, free, portable, used for teaching, contains an IB, UIMS **SuperCard**, Allegiant Technologies, Inc., 6496 Weathers Place, Suite 100, San Diego, CA, 92121, (619) 587-0500;Fax (619) 587-1314, $299, (HyperCard-like), Cards - PC or Mac. **TAE Plus**, Century Computing, 8101 Sandy Spring Rd., Laurel, MD, 20707, (301) 953-3330 or 800-823-3228, tae-info@cen.com, v5.3 site license, $1,995, v5.3 & v6.0 Beta source code site license $9,495, X. IB. FREE to NASA users. **Aonix**, 595 Market Street, 12th Floor San Francisco, CA 94105 Phone: 415-543-0900 Fax: 415-543-0145 $7,500, Motif, IB – Motif. **Theseus++**, Computer Graphics Center ZGDV, Wilhelminenstrasse 7 64283 Darmstadt, Germany, Phone +49-6151-155-112, Fax +49-6151-155-450, theseus-support@igd.fhg.de, free,X, Motif, C++, High Level Toolkit. **Thistle**, Language Technology Group, University of Edinburgh 2 Buccleuch Place Edinburgh Scotland EH8 9LW, +44 131 650 4656, J.Calder@ed.ac.uk, Free for non-commercial purposes,

Laboratory (INEL), Lockheed Martin Idaho Technologies, P.O. Box 1625, Idaho Falls, ID, 83415-3779 (208) 526-0656, tech support: hotline@sage.inel.gov, DOS, Windows 3.1/95/NT, Unix, Sun, AT&T 3b2, IBM RS6000, IB - forms interface.

Alis Translator, Alis Technologies Inc. 100 Alexis Nihon Montreal, QC, H4M 2P2 CANADA epoirier@alis.com, Internationalisation (i18n) library win95, win98 and NT.

Alpha UIMS, LoneWolf Systems, P.O. Box 81188, Pittsburgh PA 15217-0688, (412) 242-5245, alpha@lonewolf.com, $7995, X and Windows NT, (successor to Serpent), UIMS.

AlphaWindow, Cumulus Technology Corp., 1007 Elwell Court, Palo Alto, CA, 94303, (415) 960-1200, $750. Unix, Alpha-numeric terminal windows, Window System.

Altia Design, Altia, 5030 Corporate Plaza Dr #300, Colorado Springs, CO, 80919, (800)653-9957 or (719)598-4299. UNIX or Windows.

Amulet, Brad Myers, Human-Computer Interaction Institute, Carnegie Mellon Univ, Pittsburgh, PA, 15213, (412) 268-5150, amulet@cs.cmu.edu, Note: Currently being developed as OpenAmulet , free, X, Mac, or MS Windows, portable toolkit, UIMS.

Andrew User Interface System (AUIS or ATK), Fred Hansen, Andrew Consortium, Carnegie Mellon Univ, Pittsburgh, PA, 15213, (412) 268-6710, wjh+@cmu.edu, free,X, UIMS.

AppMaker, Bowers Development, P.O. Box 9, Lincoln Center, MA, 10773, (508) 369-8175, $295, Mac, IB – MAC.

AppWare, Novell, 122 E. 1700 S., Provo, UT, 84606. 800-277-2717, postmaster@novell.com, $199.00, Windows & Mac, Visual Development Environment with

California, USA, 94063 650-368-8210, info@vectaport.com, free, C++/X/Unix, Toolkit/Application Framework: Structured Graphics, Flyweight Glyphs, Scripting Language.

JAM, JYACC, 116 John Street, New York, NY, 10038, (800) 458-3313, $6000+, All; Database access & graphical Uis, UIMS.

Jx, John Lindal, jafl@cheshire-cat.wonderland.caltech.edu, available free under the SYPP license, X windows, C++ application framework.

KEE, Intelli Corp, 125 Cambridge Park, Cambridge, MA, 02140, (617) 868-5611, $5,000, LISP for PC, UNIX, toolkit.

Knowledge Pro, Knowledge Garden, Inc., 12-8 Technology Drive, Setauket, NY 11733, jslade@kgarden.com, 516-862-0600, FAX 516-862-0644, $449, PC Cards + AI expert system, Cards – PC.

LabVIEW and LabWindows, National Instruments, 6504 Bridge Point Parkway, Austin, TX, 78730-5039, (512)794-1000, newsletter@natinst.com, DOS/Windows/Mac, Virtual Instrument IDT.

Layout, Objects, Inc., 99 Rosewood Drive, Danvers, MA, 01923, (800) 424-6644, $300, Windows, Visual Language.

Loox, Xanth Informatique, 4 boulevard des iles, 32137 Issy les moulineaux Cedex - France (33) 01 41 90 65 00 from France info@loox.com Unix, Windows, Java; C, C++, Java; A vector object library and editor for 2D and 3D animation and graphs.

LUIS, Lockhead, Austin Div., Austin, TX,, (512) 386-4171, $50,000, research system by Steve Heichen.

Lxb bruce.parkin-1@umn.edu, free, Unix/X/Motif, GUI Builder.

Java (web browser or standalone), Grammar-driven UIMS and UI specification system.

TIGERS, (part of "ROSE"), CAE Electronics, CP 1800 St. Laurent, Quebec, CAN, H4L4X4, (514) 341-6780, $2,500, Silicon Graphics.

Tigre Interface Designer, Tigre Object Systems, 3004 Mission Street, Santa Cruz, CA, 95060, (408) 427-4900, $1,500, Smalltalk; MS windows, Mac, Unix, IB.

Tilcon Real-Time Developer Tilcon Software Ltd. 20 Gurdwara Road, Unit #1 Nepean, Ontario, K2E 8B3, Canada (613) 226-3917, (800) 665-5928 infonews@tilcon.com Unix: QNX, Linux; Windows 95/98/NT, Windows CE toolkit for real-time systems.

Tk/Tcl, Scriptics Corporation 2275 East Bayshore Rd., Suite 101 Palo Alto, CA 94303 650-843-6900, 650-843-6909 (fax) sarah.daniels@scriptics.com $1.000.00 formerly: J. Ousterhout, UC Berkeley X/11, PC, Mac; contains an IB, scripting language & toolkit also see Python's Tkinter.

ToolBook, Asymetrix Corp., 100 100th Ave., NE, Bellevue, Washington, 98004, (206) 637-1500, $395, Hypercard for Windows 3.0, Cards – PC.

UIM/X, Visual Edge Software, LTD, 3950 Cote Vertu, Montreal, Quebec, CAN, H4R 1V4, (514) 332-6430, *(NOTE: Visual Edge develops UIM/X but the next entries are the distributors)*, IB for Motif, IB – Motif.

UIM/X, Bluestone, Inc., 1000 Briggs Road, Mt. Laurel, NJ, 08054, (609)727-4600, $5,000, sells UIM/X for all platforms except HP (mostly in the East), IB - Motif. Also available are db-UIM/X and dbViewer for relational databases, UIM/Export for Windows, and many other widgets and tools.

cross-platform Win<=Mac capability.

Aspect, Open Inc., 655 Southpointe Ct, Suite 200,Colorodo Springs, CO, 80906. (719) 527-9700, $800-5000, portable: Motif, OpenLook, Windows, OS2 PM, Virtual Toolkit.

Authorware, Macromedia, 600 Townsend St., San Francisco, CA 94103. 415-252-2000, General Number 800-288-4797, $995, Proto – PC.

AutoCode, Integrated Systems, 3260 Jay Street, Santa Clara, CA, 94054-1215, (408) 980-1500, $20,000, Unix, VMS. Real-time control, aeronautics, UIMS -X,VMS.

Builder Xcessory, Integrated Computer Solutions, Inc. (ICS), 201 Broadway, Cambridge, MA, 02139, (617) 621-0060, info@ics.com, $3,200, Unix/X/Motif, IB - Motif
Some other products by ICS:

- ViewKit - a C++ framework written by Doug Young and the standard for SGI developers;
- EnhancementPak - a collection of 27 widgets for Motif developers;
- EditTable - Table widget
- ChartObject - 2D and 3D charting;
- View3D - a C++ 3D visualization library;
- DX - Database Xcessory. BX with database capability.

CanAda, DAINA Engineering, 4111 Central Ave. NE, Minneapolis, Minnesota, 55421, 612-781-7600, pukite@daina.com, free, MS-Windows/Ada, class library and Interface Builder.

Case PM, Casework, 1 Durwoody Dr. Suite 130, Atlanta, Georgia, 30338. (404) 399-6236, $1,995, PC/0S2 PM, complete CASE tools, IB – PC.

Chimera, Richard Taylor, CS, Univ. California, Irvine, CA 92717-3425, chimera-local@ics.uci.edu, free. SunOS 4.1.X and X11R5, Ada and C clients supported, Hypermedia System.

MacA&D and WinA&D, Excel Software, P.O. Box 1414, Marshalltown, IA 50158, 515-752-5359, info@excelsoftware.com, MacAnalyst at $995, MacA&D at $1995, MacTranslator at $495 QuickCRC Macintosh at $395 *(above run on Macintosh, HP-UX and Sun Solaris.)* WinA&D at $1995 WinTranslator at $495 QuickCRC Windows at $395 *(above run on Windows 95 and Windows NT)*; client server, CASE, Analysis, Design, Modeling.

Macintosh Toolbox, Apple, 20525 Mariani Ave., Cupertino, CA, 95014, (408) 996-1010, bundled, standard Macintosh toolkit.

Menuet/CX, Atumn Hill Software, 1145 Ithaca Drive, Boulder, CO, 80303, (303) 494-8865, BBS (303) 494-8868, $400-$800, C++ framework .

MetaCard, MetaCard Corporation, 4710 Shoup Pl, Boulder, CO, 80303, 303-447-3936, FAX: 303-499-9855, info@metacard.com, $995, 14 UNIX/X11 platforms; multimedia/hypermedia, Cards - Unix, Windows NT, Windows 95.

Motif, Open Software Foundation,,,, (617) 621-8700, $500, Unix, X/11. Usually FREE with Workstations, toolkit.

MotifGUIDE, OlafBecker, 8503 Timber Court, Burnaby, BC V5A 4B6, CANADA, free, Unix, IB.

MrEd, Matthew Flatt, Department of Computer Science - MS 132, Rice University, 6100 Main Street, Houston, TX 77005-1892, (713)527-8101, mflatt@cs.rice.edu, free, Motif; Xview; MSWindows; Mac, toolkit.

New Wave, Hewlett Packard, 19310 Pruneridge Avenue, Cupertino, CA, 95014, (800) 554-1305 or (800) 752-0900, $195, Window Environments, Toolkit - PC
Next Interface Builder, Next, Inc., 900 Chesapeake Drive, Redwood

UIM/X, Black & White Software, 2155 S. Bascom Ave., Campbell, CA, 95008, 408-369-7400, $5000, Unix, GUI builder. (sells UIM/X mostly in the west.) Also available are: UIM/XMove: $7700, Unix/Windows, Interactive graphics/animation builder. UIM/Orbix: $TBD, Unix, CORBA-based GUI builder. UIM/Ada: $4000, Unix, Ada-based GUI builder. Integration Manager: $250, Unix, GUI integration tool. Visual Action Toolset: $2500, Unix, Live docs builder using Display PostScript. XRT widgets: $1495-2495, Unix, Widgets (graph, 3D, table). View.h++: $795, Unix/Windows, C++ class lib for Motif. RWCGEN: $695, Unix, UIM/X C++ code generator for View.h++.

UTAH, apparently, no longer available. See PowerCharger or ViewSoft.

V, Bruce E. Wampler, Department of Computer Science, University of New Mexico, bruce@objectcentral.com, free, X Windows using the Athena widgets and Microsoft Windows 3.1, C++ toolkit.

VAPS, Virtual Prototypes, 4700 de la Savane, Suite 300, Montreal, Quebec, Canada, H4P 1T7, (514) 341-3874, bennett@VirtualPrototypes.CA, $10,000 to $41,500, Builder, SGI, SUN, and HP, UIMS.

Vermont Views with Designer, Vermont Creative Software, P.O. Box 250, 140 Main St., Richford, VT 05476, Sales & Cust Svc 802-848-7731, Tech. suppt 802-848-7571, fax 802-848-3502, info@vtsoft.com, DOS or Unix (or OVMS), database access, IB.

ViewSoft Internet, ViewSoft Inc., 250 W. Center, Provo, UT 84601, (801) 377-0787, FAX (801) 377-0788, info@viewsoft.com, $2,995, Internet RAD tool for developing thin-client Internet applications. Develop in Visual C++, deploy user

Chiron, Richard Taylor, CS, Univ. California, Irvine, CA 92717-3425, arcadia-chiron@ics.uci.edu, free, X: Motif or OpenLook, UIMS.

Choreographer, Guidance Tech, 800 Vinial Street, Pittsburgh, PA, 15212, (412) 231-1300, $7,500, PC/0S2 PM, UIMS – PC.

CLIM, International Lisp Associates, 114 Mount Auburn St, 4th Floor, Cambridge, MA, 02138, (800) 477-CLIM, $?, toolkit and UIMS for CommonLisp, UIMS – Lisp.

COBOL sp2, Flexus, P.O. Box 640, Bangor, PA 18013-0640. Rtwolfe@flexus.com, $795.00(Win16), $1,195.0(Win32). User Interface Development System for COBOL programmers.Windows.

DataViews, DataViews Corporation 47 Pleasant St., Northampton, MA, 01006, (413) 586-4144, $17,700, UNIX, VMS, IB – X.

DEC Forms VDP, Digital Equip Corp, 146 Main Street, Maynard, MA, 01754-2571, $161-291, DEC.

Delphi from Borland.

Demo-II (Dan Bricklin), Lifeboat Publishing, 1163 Shrewsbury Ave, Shrewsbury, NJ 07702, 800-336-1166, 908-389-9227, $249, PC/Char Graphics, Proto – PC.

DesignInRealTime (DIRT), University of Kent-Canterbury, UK, rlh2@ukc.ac.uk, free, X, toolkit.

DevGuide, Sun, 2550 Garcia Avenue, Mountain View, CA, 94043, (415) 960-1300, $250, Open Windows Devel, IB – OpenLook.

Dialog Director, HyLight Limited, Christopher E.Hyde, free, drjekyll@hylight.demon.co.uk, GUI toolkit for AppleScript for Mac.

Director, Macromedia, 600 Townsend St., San Francisco, CA 94103 USA, macropr@macromedia.com, $598, Mac, PC/Windows, Proto.

City, CA, 94063, (800) 848-NEXT, bundled, UNIX/NeXT, IB – NeXT.

Next Step, Next, Inc., 900 Chesapeake Drive, Redwood City, CA, 94063, (800) 848-NEXT, bundled, UNIX/NeXT, Toolkit.

ObjectBuilder, ParcPlace Systems Inc.), Openware Technologies, 8000 Arlington Expressway, Suite 600, Jacksonville, Florida, 32211, (904) 725-7187 (X212), tom@Jax.openware.com, $5,400, Sun O/S, Solaris, HP, IBM, SCO, SGI, DEC, Windows95, NT.

ObjectVision, Borland, 1800 Green Hills Road, Scotts Valley, CA, 95067-0001, Sales: (800) 331-0877, Support: (408) 438-5300, Fax: (800) 408-0001, $64, Windows, Visual Language.

OI, *Solbourne and then ParcPlace Systems Inc.*OpenwareTechnologies, 8000 Arlington Expressway, Suite 600, Jacksonville, Florida, 32211, (904) 725-7187 (X212), tlovegr@jax.gttw.com or tom@jax.openware.com, $5,400, Sun O/S, Solaris, HP, IBM, SCO, SGI, DEC, Windows95, NT, toolkit.

OLIT, Sun, (800) 872-4786,Unix, X, OpenLook, toolkit.

Omnis7, Blyth Software, Incorporated, 989 East Hillsdale Blvd., Suite 400, Foster City, California, 94404, 415-571-0222, postmaster@blyth.com, $500+, Mac or PC, IB & DB, front end.

OpenAmulet, (new version of Amulet). free, X, Mac, or MS Windows, portable toolkit, UIMS.

Open Dialoque, HP/Apollo Computer, 330 Billerica Road, Chelmsford, MA, 01824, (800) 752-0900, $2,020, UIMS – declarative language, UIMS – X.

Open Interface, Neuron Data, 1310 Villa Street, Mountain View, CA 94041, 1-800-876-4900, sales@neurondata.com, $10,000, portable: Motif, OpenLook, PM, Windows, Mac, Virtual Toolkit.

interface remotely as Java applet or ActiveX control. Product handles all network connections visually.

Visaj, Imperial Software Technology, Berkshire House, 252 Kings Road Reading, Berkshire RG1 4HP, United Kingdom, +44 118 958 7055, fax: +44 118 958 9005; USA Address: Imperial Software Technology, 120 Hawthorne Ave, Suite 101, Palo Alto, CA 94301, (650) 688 0200, Fax: (650) 688 1054, Java, IB.

VISION, Unify Corporation, 181 Metro Drive 3rd Floor, San Jose, CA, 95110, 408-467-4500, customers@unify.com, US$4,995.00, HP-UX Sun Solaris Dec Unix IBM AIX MS-Windows Window-NT, Macintosh, virtual toolkit for database & client server.

Visual Basic, Microsoft, 10700 Northup Way, Bellevue, Washington 98004, (800) 426-9400, MS-Windows only, Proto or UIMS.

Visual C++, Microsoft, 10700 Northup Way, Bellevue, Washington 98004, (800) 426-9400, MS-Windows only, IB + programming environment.

Visual Designer, Intelligent Instrumentation, 6550 S. Bay Colony Drive, Tucson AZ 85706, 1-800-685-9911, sales@instrument.com, PC/WINDOWS, signal-processing application generator.

Visual Object Manager, ViVi Software Srl, corso Dogali 10/15, Genova, Italy, 16136, +39 10 213 513, sales@vivi.dsnet.it, $1600, WIndows NT/ObjectStore 3.0.x 4.0, Windows GUI OODBMS.

Visual/Recital, Recital Corporation, 85 Constitution Lane, Danvers, MA 01923, 800-USE-R4GE, Headquarters 508-750-1066, sales@recital.com, $4000, Motif/X11, IB & DBMS.

Visual Tcl, SCO, 400 Encinal St., Santa Cruz, CA 95061, 1-800-SCO-

Display Construction Set, AT&T, (614) 860-4357, Unix, X, OpenLook, IB –OpenLook.

Druid, Gurminder Singh, Institute of Systems Science, National University of Singapore, Heng Mui Keng Terrace, Kent Ridge, Singapore 0511, REPUBLIC OF SINGAPORE, +65 772-3651, FAX: +65 774-4998, druid-info@iss.nus.sg, $1250, X and Motif, IB.

Easel II, Easel, 25 Corporate Drive, Burlington, MA, 01803, (617) 221-2100, $10,000, DOS, 0S/2, UIMS – PC.

ECANSE Environment for Computer Aided Neural Software Engineering, Siemens Austria, Gudrunstrasse 11, A-1100 Vienna, Austria, (+43-1)-1707-45985, fax: (+43-1)-1707-56399, ecanse@siemens.at, $3500 - $21000, visual language.

ExoCODE, EXOC, 500 Hyacinth Place, Highland Park, Ill, 60035, (708) 926-8500, $1,500, Motif or OpenLook or SunView, IB – Unix.

EZX, Sunrise Software Sys, P.O. Box 329, Newport, RI, 02804, (401) 847-7868, $3,500, Motif, IB – Motif.

FaceSpan, Digital Technology, $200, Interface Builder for AppleScript on Mac.

Forms, Mark H. Overmars, Department of Computer Science, Utrecht University, PO Box 80.089, 3508 TB Utrecht, the Netherlands, markov@cs.ruu.nl, ftp://ftp.cs.ruu.nl//pub/SGI/FORMS/ free, SGI GL, toolkit and IB.

FormsVBT, Marc H. Brown, DEC Systems Research Center, 130 Lytton Ave., Palo Alto, CA 94301, (415) 853-2152, mhb@pa.dec.com, free, Modula-3 on either X or Windows, UIMS - Research System.

Fresco, X Consortium Inc, ftp://ftp.x.org/pub/R6untarred/

OpenUI, Open Software Associates Inc., 20 Trafalgar Square, Nashua, NH 03063, (603) 886 4330, $5,000, Windows (3.1, '95, NT), MAC, Presentation Manager, Motif (12 flavors of UNIX plus VMS), UIMS for INTERNET and distributed applications.

Oracle Tools, Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA, 94065, (800) 633-0583, $6,000, Database front ends for any platform, UIMS.

PLUS, Spinnaker, 201 Broadway, Cambridge, MA, 02139, (617) 494-1200, $495, Mac (HyperCard), Cards – PC.

PowerCharger for MFC, ViewSoft Inc., 250 W. Center, Provo, UT 84601, (801) 377-0787, FAX (801) 377-0788, info@viewsoft.com, $199, Visual C++/MFC, IB.

PowerPlant, Metrowerks, Inc. 2201 Donley Drive, Suite 310, Austin, TX 78758, 800.377.5416, fax: 512.873.4900, sales@metrowerks.com, Mac, C++ Application Framework.

PowerBuilder, Powersoft 561 Virginia Road, Concord, MA 01742, USA, 1-800-395-3525 or 1-508-287-1500, Fax 1-508-287-1600, PC, Unix and Mac, Interface Builder and Database front end.

POWERMEDIA, OmniSoft, 1723 Westbrook Avenue, Los Altos, CA, 94024, 415-917-1395, bchandra@omnisoft.com, US$79, MS-Windows Window-NT, MMtools; Toolkit for Multimedia.

Presentation Manager, Microsoft, One Microsoft Way, Redmond, WA, 98052-6399, (800) 426-9400, OS/2 toolkit, toolkit.

Prograph, Pictorius Incorporated, 2745 Dutch Village Road, Suite 200, Halifax, Nova Scotia B3L 4G7, Canada, (902) 455-4446, FAX: (902) 455-2246, Sales: (800) 927-4847,

UNIX, info@sco.com, free, UNIX, Motif Scripting Language.

Vsystem, Vista Control Systems, 134B Eastgate Dr, Los Alamos, NM, 87544, (505) 662-2484, FAX (505) 662-3956, sales@vistanm.com,, X/Motif, DEC VMS, Vaxeln, OSF/1, IB – Vdraw, Vscript.

Windowcraft, WindowCraft Corp., 6 New England Executive Park, Burlington, MA, 01803, (617) 272-0999, $295, PC/Hypercard, Cards - PC

WindowsMAKER, Blue Sky Software, 2375 E. Tropicana Ave. # 320, Las Vegas,NV, 89119, (702) 456-6365, $795, Windows 3.0, IB – Windows.

Windows Develop. Kit, Microsoft, One Microsoft Way, Redmond, WA, 98052-6399, (800) 426-9400, Windows toolkit, toolkit.

VXP -- Visual X windows Programming Interface, Yong Chen, 210 Wells Fargo Dr. #216, Houston, TX 77090, (713) 586-9089, stdyxc05@pip.shsu.edu, free, X: Motif, GUI Builder.

WINTERP, Niels P. Mayer 5214-F Diamond Heights Blvd., #635 San Fransisco, CA 94131 winterp-request@netcom.com, free, Unix/X/Motif, UIMS.

WxWindows, Julian Smart, 12 North Street West Uppingham, Rutland LE15 9SG, UK julian.smart@ukonline.co.uk, free, Windows, UNIX (Motif & GTK), virtual toolkit.

X-In-Ada, Top Graph'X, 10 Allee de la Mare Jacob, 91290, La Norville, FRANCE, (33) 1 69 26 97 88, Fax: (33) 1 69 26 97 89, 100071,45@compuserve.com, X, PEX and Motif in Ada.

Xbuild, Siemens Nixdorf, 4

xc/doc/hardcopy/Fresco ,
free, C++/X/Unix, Toolkit/
Structured Graphics.

Galaxy, Ambiencia Information Systems, Inc. Campinas, Brazil, phone/fax: (019) 55-19-243-7328 support@ambiencia.com. *[formerly from Visix Software Inc., which has been dissolved; see Note R14. 11440 Commerce Park Drive, Reston, VA, 22091]*
Mac, Windows, Motif, OpenLook; very complete, Virtual Toolkit .

Garnet, Brad Myers, CMU, School Computer Science, Pittsburgh, PA, 15213-3891, (412) 268-5150, garnet@cs.cmu.edu, free, Common Lisp, X or Mac. UIMS – Research system.

GIB, TAO Research Corp., 39812 Mission Blvd, #205, Freemont, CA, 94539, (510) 770-1344, $475,
MS Windows, IB – Windows.

GINA, GMD (German National Research Center for Computer Science), Schloss Birlinghoven, St. Augustin, Germany, D-53754, , spenke@gmd.de,
free, LISP / Motif, Application Framework – UIMS.

Glade, glade@glade.pn.org,
free, Unix, GTK+, Gnome, Interface Builder.

GMW and Guide, ASTEC Corporation, Nagashimo-Daiichi Bldg. 9F, Shibuya-Ku, Tokyo, Japan 00150, +81-3-476-0183, UIMS.

GRAMMI, SETT, Inc., 5303 Spectrum Drive, Suite G, Frederick, MD, 21701, (301) 695-6960 or Jennifer Lott at (800) 877-1815; info@evb.com, Ada & X, IB.

Groupkit, Saul Greenberg, University of Calgary, Dept of Computer Science, Calgary, Alberta, Canada, T2N 1N4, (403) 220-6087, saul@cpsc.ucalgary.ca,
free, Unix, Tcl/Tk and Tcl-DP, Groupware Toolkit - Research System.

Guide 2, InfoAccess Inc. (formerly OWL International Inc.), info@prograph.com
or sales@prograph.com.

Newsgroup: comp.lang.prograph
Sales Office: Suite 300, 447 Battery St., San Francisco, CA U.S.A. 94111, (415) 773-8234;
Fax: (415) 391-3942, $695,
Macintosh, visual language.

Progress Version 7, Progress Software Corporation, 14 Oak Park, Bedford, MA, 01730,
(617) 280-4000, $300+,
Windows & Motif & DOS, IB/4GL.

Proteus 5.0, Genus, 11315 Meadow Lake, Houston, Texas, 77077,
(713) 870-0737, $249,
PC/Programmer's Toolkit, Cards – PC.

Protofinish, Genesis Data Systems, 8415 Washington Place, Albuquerque, NM, 87113, $300,
PC/Char Graphics, Proto – PC.

Protoscreens, Balley & Bailey, 859 East 2850 North, Ogden, UT, 84414, (801) 782-2345, $225,
PC/Char Graphics, Proto – PC.

Qt, Troll Tech AS, Postboks 6133 Etterstad, N-0602 Oslo, Norway, http://www.troll.no,
fax +47 22646949,
sales@troll.no, free or expensive depending on application,
Windows 95, NT, Linux, Solaris, HP-UX, AIX, Digital UNIX, IRIX, FreeBSD, BSDI, more Unixes - C++ GUI toolkit.

Rapid Design, Emultek Inc, 42 Farview Road, Hopewell Jct., NY 12533, 914-226-4090,
Fax: 914-227-6369,
103113.3204@compuserve.com,
$6,000, MS Windows, Windows 95, and NT, visual language.

RIPL, Computer Technology Assoc, 5670 Greenwood Plaza Blvd., Suite 200, Englewood, Colorado, 80111, (303) 889-1200, not for sale,
VAXStation, VMS, UIMS - C&C.

Sammi, Kinesix, 9800 Richmond Ave, Suite 750, Houston TX 77042; 713 953 8300; Fax: 713 953 8306; sammi@kinesix.com;
Cambridge Center,
Cambridge, MA, 02142,
(617) 864-0066, $1,895,
Unix/X/Motif, IB – Motif.

X-Designer, Imperial Software Technology, Berkshire House, 252 Kings Road Reading,
Berkshire RG1 4HP, United Kingdom, +44 118 958 7055,
fax: +44 118 958 9005;
USA Address: Imperial Software Technology, 120 Hawthorne Ave, Suite 101, Palo Alto, CA 94301, (650) 688 0200,Fax: (650) 688 1054,$3,500,
X Windows, Java, IB

XfaceMaker, Nova Software Labs *(formerly Non Standard Logics)*, 57-59 rue Lhomond, 75005 Paris, FRANCE, (33-1) 44 08 70 80, info@nsl.fr,
UNIX X/Motif, includes 2D and 3D graphics widgets, IB.

XMove, Siemens AG, Gudrunstrasse 11, Vienna, A-1101, +43 1 1707-45432, xmove@ws2302.gud.siemens.co.at, $10,000,
Unix, dynamic application defined graphics.

XPCE, SWI, University of Amsterdam, Roetersstraat 15, 1018 WB Amsterdam, The Netherlands, +31 20 5256121, xpce-request@swi.psy.uva.nl, 500 ECU (academic), 2000 ECU (commercial) (1 ECU ~ $1.3 US), Unix/X11 and Win32, Portable GUI for dynamically typed and strongly typed languages (Prolog, Lisp and C++).

XRT, KL Group Inc. 260 King Street East, Third Floor, Toronto, Ontario, Canada. (416)594-1026, info@klg.com, $995, Motif, Widget Libraries. Also components for Windows and Java.

Xview, Sun, (800) 872-4786, free, Unix/X OpenLook, Toolkit.

XVT, XVT Software Inc., 4900 Pearl East Circle, Boulder, CO, 80301, USA, 1-800-678-7988 or (303) 443-4223, FAX: (303) 443-0969, info@xvt.com, $1950-$6300, MS Windows, Windows NT, OS/2, Macintosh, OSF/Motif, OPEN

2800 156th Ave. SE, Bellevue, WA, 98007. (206) 747-3203, $275-$300, PC/HyperText, Cards – PC.

GX Series Developer's Pak, Genus, 11315 Meadow Lake, Houston, Texas, 77077, (713) 870-0737, $589, PC/Programmer's Toolkit, Toolkit – PC.

HP Interface Architech, Hewlett Packard, 4 Choke Cherry Rd., Rockville, MD, 20850, (301) 670-4300, Unix/X - based on UIMX, IB – X.

HyperCard, Apple Computer, 20525 Mariana Avenue, Cupertino, CA, 95014, (408) 996-1010, $50, Mac, Cards – Mac.

Hyperwire, Kinetix *(formerly AutoDesk Multimedia)*, $199, Windows 95/NT (runtime Java libraries can be used on any platform, but development tool is strictly Windows), Java Development Tool.

ICON Author, AimTech Corp, 20 Trafalgar Square, Nashua, NH, 03063-1973, (800) 289-2884, $995, Windows 3.0, proto & courses, training, Proto – PC.

UNIX(HPUX, IBM AIX, SCO, SunOS, Solaris, OSF/1, Ultrix, IRIX, REAL/IX, UnixWare, QNX, Lynx, Venix, VxWorks) and VMS; for real-time systems; IB and Data Visualization Tool.

Serpent, Carnegie Mellon Univ./SEI, 5000 Forbes Avenue, Pittsburgh, PA, 15213-3890, (412) 268-6763, ftp://ftp.sei.cmu.edu/pub/serpent/, **free,**X, research system by the Software Engineering Institute, UIMS – X.

SET, Caset Corp., 33751 Connemara Dr., San Juan, Capistr, 92693, (714) 496-8670, $5,000, UIMS - state transition models, X, UIMS – X.

SL-GMS, SL Corp., Suite 110 Hunt Plaza, 240 Tamal Vista Blvd., Corte Madera, CA, 94925, (415) 927-1724, $12,500, X, VMS, UIMS for real-time control, UIMS - X,VMS.

StarView, Star Division Corp., 2180 Sand Hill Rd #320, Menlo Park, CA 94025, (800) 888-8527, svinfo@stardiv.de, $495, MS-Windows 3.1, OS/2 2.1, Mac, Motif, Virtual Toolkit

LOOK, Character systems for DOS, UNIX, and VMS, Virtual Toolkit and IB.

YACL, M. A. Sridhar, University of South Carolina, Columbia, SC 29208, 803-777-2840 Fax: 803-777-3767, sridhar@cs.sc.edu, free, Windows, OS/2, X/Motif, Virtual Toolkit, GUI C++ Class Library.

ZApp, *(formerly from Inmark Development Corporation)*, Rogue Wave Software, Inc., 850 SW 35th St., Corvallis, OR 97333, (541) 754-3010 or (800) 487-3217, Fax: (541) 757-6650, zappsupport@roguewave.com, DOS Text, DOS Graphics, Windows(16-bit), Windows NT, Windows 95, OS/2, Warp, HP-UX, IBM AIX, SCO UNIX, SunOS, Solaris, UnixWare, SGI IRIX, Virtual Toolkit, GUI C++ Class Library and IB.

Zinc, Zinc Software Inc., 405 South 100 East, 2nd Floor, Pleasant Grove, UT, 84062, (801) 785-8900, info@zinc.com or europe@zinc.com $500+$300, DOS, MS Win, OS/2, Mac, Motif, C++ class lib, Virtual Toolkit, IB, also ZAF--Zinc Application Framework.