Uma Arquitetura para Interfaces Inteligentes e Amigáveis

Otaviano C. Wanderley Neto*- ocwn@di.ufpe.br
Décio Fonseca†- df@di.ufpe.br
Departamento de Informática - Universidade Federal de Pernambuco
Caixa Postal 7851 - CEP 50732-970 - Recife

Abstract

This paper shows an architecture for Friendly and Intelligent Interfaces. Friendly because it can adapt to user's characteristics and adjust to the his necessities. It is Intelligent because it can learn with the user, while he interacts with the interface. The most important objective addressing by architecture is to become the interaction easier and efficient. In this paper we present the interface's structure and algorithms.

Resumo

Este artigo apresenta uma Arquitetura para Interfaces Inteligentes e Amigáveis. Amigável pois tem como uma de suas características o poder de adaptação ao perfil do usuário, ajustando-se a sua realidade, e Inteligente por aprender com o usuário através do uso, interagindo com o mesmo de modo a tornar a interação mais fácil e eficiente. É apresentada aqui a estrutura da interface, assim como algoritmos utilizados para tornar a interação mais produtiva e fácil.

*Mestrando em Informática - DI/UFPE
†Professor Orientador - Doutor pela Universidade de Paris

1 Introdução

Recentemente os computadores evoluíram bastante, levando o usuário a modificar os parâmetros antes utilizados na escolha de softwares. O que antes era utilizado como critério de seleção passou a ser menos observado, como a eficiência por exemplo, pois o poder de processamento e armazenamento de informações aumentaram consideravelmente. Por outro lado foi surgindo uma nova preocupação antes não muito questionada que é a facilidade de uso. Isto fez com que as atenções se voltassem para o parâmetro interface, a interação com o usuário, visto que a eficiência de sistemas similares era facilmente suprida pelas máquinas atuais.

Segundo [Som92] o estudo de fatores humanos na construção de softwares é importante porque computadores são utilizados por pessoas, e se as habilidades e limitações destas pessoas não forem levadas em consideração quando a interface é projetada, estas pessoas não a utilizarão adequadamente, podendo representar uma baixa produtividade do usuário.

Em [And88], o custo total do projeto de um sistema é distribuído em três categorias: *Hardware, Software e Orgware*. Esta última categoria diz respeito a treinamento de pessoal, reestruturação de procedimentos de trabalho, implementação de novos procedimentos e outros custos relacionados com o pessoal envolvido na utilização do sistema.

Na figura 1, de [And88], observamos que o custo com hardware está diminuindo enquanto o custo com software e orgware está aumentando.

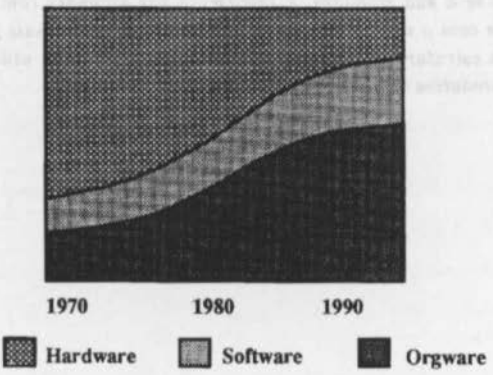

Figura 1: Divisão de custos com Hardware, Software e Orgware

É importante lembrar também que os projetistas de produtos comerciais estão atentos em que a melhor forma de vencer a concorrência entre aplicações funcionalmente similares é através do parâmetro "fácil de aprender" e "fácil de usar" de forma segura. Para a maioria dos usuários, o fácil de usar torna-se o principal indicador de qualidade da aplicação e assim um forte parâmetro para a sua escolha [Cam91].

Para melhorar a qualidade da interação assim como aumentar a produtividade do usuário é que se tem buscado meios mais eficientes de comunicação com o usuário, através

de interfaces amigáveis porém poderosas, atendendo a uma gama maior de usuários.

Já se sabe que é extremamente necessário se ter um projeto de interface independente do projeto da aplicação em si, o que convencionou-se chamar de *Independência de Diálogo* [HH89]. Esta separação traz inúmeras vantangens, entre elas :

- Manutenção do sistema torna-se mais fácil;
- Alterações na interface não necessariamente implicam em mudanças no sistema, e vice-versa;
- A interface é melhor produzida, uma vez que existe um projeto específico para ela.

Um fato importante é observado na construção de interfaces. É que no projeto de muitas delas já se faz uso de um projetista gráfico para melhorar o design de telas e cores [Ret92].

Um dos principais problemas na Interação Homem-Computador, IHC, é justamente a quantidade de áreas distintas envolvidas no seu estudo , tornando-a uma área multidisciplinar [Gui94],[Dow91]. A figura 2 ilustra algumas delas.

Figura 2: Áreas que contribuem para a evolução de IHC

Diante das dificuldades encontradas na interação com vários softwares e analisando as questões expostas anteriormente, assim como outras não mencionadas neste paper devido a espaço, é que surgiu a necessidade de se montar uma arquitetura que seja a base para interfaces realmente amigáveis sem que com isto perca-se em eficiência.

Será mostrada a estrutura da interface na próxima seção, definindo alguns objetos de diálogo, construtores e manipuladores, descrevendo depois alguns algoritmos utilizados de forma a incorporar na interface os parâmetros fácil de usar e fácil de aprender.

Alguns trabalhos têm sido feitos no sentido de se construir interfaces amigáveis e poderosas, como em [MMK93] e em [WF91], que são construtores de interface que funcionam por demonstração, tendo os dois as mesmas características. No entanto eles não permitem a criação de interfaces adaptativas, que se ajustem ao perfil particular do usuário, como em [JH93] por exemplo, onde artigos que chegam por e-mail são classificados de acordo com o interesse do usuário.

Existem ainda sistemas cooperativos, como [Cam91] e [NdG93], que implementam funções de ajuda diferenciada para cada usuário, de acordo com seu perfil, mas que não são adaptáveis nos estilos de diálogo nem são demonstracionais.

O que se percebe na realidade é que os sistemas de interfaces não são completos, os que implementam adaptação não são demonstracionais e os que são demonstracionais não são adaptativos, existindo também aqueles que são cooperativos mais não são demontracionais nem adaptativos.

Para construir uma interface mais completa, ou seja, adaptativa, demonstracional e cooperativa, é que surgiu a idéia de montar uma arquitetura para interfaces que permitisse tais funcionalidades.

Para se chegar a esta arquitetura foi realizado um estudo aprofundado sobre interfaces, onde parte dele pode ser encontrado em [FN95], que é um tutorial sobre interfaces.

2 Arquitetura

A estrutura proposta consegue absover algumas características inteligentes pois incorpora ao sistema simultaneamente três funcionalidades, ou seja, este modelo permite a criação de interfaces *Demonstracionais, Cooperativas e Adaptativas.*

Algumas vantagens obtidas com a incorporação destas características são:

- Uma interface demonstracional funciona por demonstração, ou seja, muitas das atividades realizadas pelo usuário consideradas repetitivas são facilmente identificadas pela interface demonstracional, que passa então a executá-las para o usuário, desde que permitida pelo mesmo, que dará ou não autorização para tal. Com isto uma interface demonstracional diminui sensivelmente a quantidade de digitação e tomada de decisões realizadas pelo usuário [Mye92].
- As interfaces adaptativas têm uma importância muito grande com relação a satisfação do usuário, uma vez que sua principal característica é a sua adequação ao perfil particular de cada um. Dessa forma quanto mais a interface se aproxima da realidade do usuário, satisfazendo suas exigências e habilidades, maior será a sua satisfação e sucesso na utilização do sistema [FG90].
- Quanto a interface ser cooperativa e interativa, esta é uma outra característica desejável em qualquer sistema de interfaces, uma vez que existe um maior diálogo entre o usuário e o sistema, pois tem-se uma constante ajuda da interface no sentido de o usuário conseguir o seu objetivo o mais rápido possível de uma forma bastante convergente [Cam91]. A cooperação neste tipo de interface pode se apresentar de diversas formas, como por exemplo:
  - tentar aproveitar ao máximo as ações do usuário, evitando digitação e comandos desnecessários;
  - fazer sugestões de execução;
  - tratamento de erro mais amigável, mais próximo do perfil do usuário.

Para permitir a criação de interfaces que sejam ao mesmo tempo demonstracionais, adaptativas, cooperativas e independentes de diálogo, que representam interfaces mais amigáveis e inteligentes, é que propomos neste trabalho uma forma de estruturação de interfaces baseadas em tabelas de controle, que definem as estrutura da interface, como ícones, menus, e outros objetos de interação, assim como define também as várias sequências de execução existentes na interface, atendendo ao máximo os objetivos do usuário.

Os objetos da interação homem-computador são divididos em dois grupos,o primeiro compõe-se exclusivamente de **Objetos da Interface**, ou seja, objetos que são utilizados na construção da interface propriamente dita. Estes objetos da interface são divididos em três sub-grupos de classes de objetos, são eles os *Objetos de Diálogo*, *os Objetos Construtores* e os *Objetos Manipuladores* ou de *Controle*.

O outro grupo, o de **Objetos do Usuário**, corresponde aos objetos que mantém informações a respeito de cada um, através da *modelagem do usuário*, adquirindo, mantendo e renovando informações referentes ao seu nível de conhecimento, seja no domínio da aplicação, na utilização do sistema ou a na sua formação específica. Estes objetos servem para identificar o usuário, monitorar suas atividades e ajudá-los sempre que for necessário de forma cooperativa.

3 Objetos da Interface

3.1 Objetos de Diálogo

Entre os objetos de diálogo observados, consideramos quatro deles essenciais, são eles: *Menu, Ícones/botões, Área de texto* e *Imagem*. Os objetos de diálogo são definidos cada um em suas respectivas tabelas. Existem portanto quatro tabelas de objetos de diálogo. Cada tabela contém os seguintes campos:

- **Código do objeto**
- **Descrição**
- **Título**
- Mais as informações inerentes a cada objeto, conforme a seguinte descrição:
  - **Menus**: tipo de menu, lista de opções, número de opções visíveis, quantidade máxima a ser selecionada e teclas de controle.
  - **Ícones/Botões**: texto para o botão ou nome do arquivo que contém imagem para o ícone, tipo de ícone/botão;
  - **Área de Texto**: código do campo, string inicial, tipo do campo, tamanho, domínio, posição relativa do título e variável do sistema associada ao campo.
  - **Imagem**: nome do arquivo que contém imagem e tipo.

Ao se projetar uma interface nesta arquitetura deve-se definir previamente todos os objetos de diálogo nestas tabelas, conforme os dados necessários para cada estilo.

3.2 Objetos Construtores

Com relação aos objetos construtores, estes são dois, um servindo de apoio, que é o objeto *janela*, e o outro é o objeto *cenário*, formado por um grupo de objetos de diálogo inseridos no seu contexto, que por sua vez é limitado por uma janela.

Seguindo a idéia dos objetos de diálogo, os objetos construtores janela e cenário são definidos em duas tabelas, contendo cada uma as seguintes informações:

- **Código do Objeto**
- **Descrição**
- **Título**
- Mais as seguintes informações específicas de cada objeto:
  - **Janela**: Tipo, PosX, PosY, TamX e TamY.
  - **Cenário**: Código da Janela, lista de menus, lista de ícones/botões, lista de imagens, lista de áreas de texto e cenários filho.

Com relação ao objeto cenário, este ainda tem outra tabela que mantêm a relação dos cenários sinônimos, ou seja, são os cenários que têm a mesma função na interface, variando de um para o outro apenas os objetos de diálogo utilizados. Tais cenários servem para adaptar a interface de acordo com as preferências do usuário. As informações desta tabela são:

- **Código do Cenário**
- **Lista de Cenários Sinônimos**
- **Lista de conceitos e definições associados ao cenário**

3.3 Objetos Manipuladores ou de Controle

A interface funciona basicamente apoiada em dois objetos, um deles já visto é o objeto cenário e o outro é o objeto *evento*. Através deste é que a interface é ativada, seguindo os *caminhos de execução* do sistema até se atingir o *objetivo do usuário*, o que representa a obtenção do *plano de execução* do mesmo.

3.3.1 Evento

Um evento é toda ação relevante gerada pelo usuário que configure uma alteração no estado do sistema. Muitas vezes um evento é identificado por uma mudança visual na interface.

Os eventos são utilizados pelo usuário para obter seu objetivo no sistema, realizando alterações no estado da interface quando achar necessárias, ativando ou desativando cenários.

A estrutura proposta neste trabalho tem como objeto principal o *cenário*, em cima do qual é feita toda especificação da interface. A existência de eventos, no entanto, é de

vital importância para o funcionamento do sistema, uma vez que através deles é que o usuário mostra o que quer, agindo sobre os objetos de diálogo.

Para identificar qual evento o usuário está executando é que existem as tabelas de controle dos objetos de diálogo. Cada objeto deste tem associado uma tabela de controle de eventos, que contém cada uma as seguintes informações:

- **Código do Evento**
- Mais os seguintes campos de acordo com o objeto de diálogo:
  - **Menus**: Código do menu e opção selecionada.
  - **Ícones/Botões**: Código do ícone ou botão.
  - **Área de Texto**: Código da área de texto e código do campo.
  - **Imagem**: Código da imagem e área sensitiva.

Além das tabelas de controle dos objetos de diálogo, existe mais uma que define a estrutura de cada evento, ou seja, uma tabela de eventos que mantém as informações referentes a cada evento do sistema. Estas informações são:

- **Código do Evento**
- **Descrição**
- **Variáveis a serem checadas**
- **Variáveis a serem alteradas**
- **Atualização do cenário**- Fechar, manter inalterado ou iconizar.
- **Tipos de usuários habilitados**

Ao se utilizar um objeto de diálogo, o sistema identifica-o e com o auxílio das tabelas de controle consegue determinar que evento está sendo acionado, fazendo logo em seguida uma verificação, com a ajuda da tabela de eventos, para saber se o mesmo pode ou não ser executado.

3.3.2 Caminhos de Execução

A transição de cenários é a passagem de controle de um cenário para o seu sucessor, desde que satisfeitas as condições para tal transição. A medida em que o usuário vai utilizando uma aplicação o sistema vai passo a passo apresentando uma evolução que se caracteriza pela passagem por vários cenários até se chegar ao objetivo do usuário, quando o mesmo então termina sua execução ou inicia um novo objetivo no sistema.

Conforme visto na figura 3, as transições de cenários são agrupadas na ordem em que vão ocorrendo, formando-se uma cadeia de cenários, contendo um cenário inicial e um outro final , que corresponde ao término da cadeia, que pode representar ao usuário a obtenção de seu objetivo no sistema.

O sistema de interfaces utiliza os cenários, que correspondem aos estados, que reagem as ações do usuário, que correspondem aos eventos. Deste modo fazemos uma adequação da transição de cenários com os Diagramas de Transição de Estados e com as Redes de

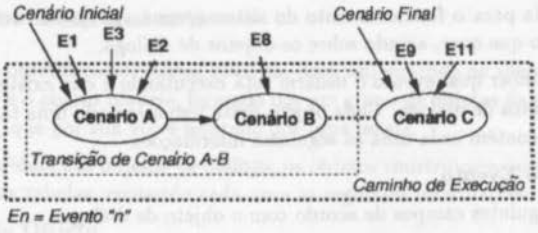

Figura 3: Transição de Cenários e Caminho de Execução

Petri, pois suas representações gráficas são de fácil utilização e entendimento [Ben94], [Sto94] e [Pfl91].

3.3.3 Planos de Execução

Todas as vezes que o usuário acessa e executa alguma aplicação ele está interessado em obter algum resultado. Baseado nesta necessidade é que achamos necessário no sistema de interfaces a estrutura de Planos de Execução.

Um Objetivo do Usuário é uma meta que o mesmo quer alcançar com a utilização do sistema, ou seja, é o trabalho que ele quer executar, seja definir uma aplicação, fazer uma consulta, executar um comando, digitar valores num campo, enfim, o objetivo do usuário é a tarefa que o mesmo deseja executar no sistema e o plano de execução é a forma através da qual ele vai obter suas tarefas, ou seja, é o conjunto de caminhos de execução que o mesmo vai ter que seguir para concluir com êxito suas atividades.

A idéia de planos de execução para o usuário é utilizada em vários sistemas. Em [PR94], por exemplo, um plano consiste de vários passos do usuário. Estes passos podem ser refinados, onde cada refinamento de uma passo é representado por um outro plano, resultando numa estrutura hierárquica que oferece ao usuário diversos níveis de abstração.

O objetivo de se definir planos de execução é que caso identifique-se qual o objetivo do usuário, seja inferindo através do uso ou através de perguntas diretas ao mesmo, o sistema poderá guiá-lo na sua execução, encaminhando para os cenários corretos.

A sua estrutura contém as seguintes informações:

- **Código do Plano**
- **Descrição**
- **Lista de Caminhos de Execução**
- **Nível**

4 Objetos do Usuário

Para a Modelagem do Usuário, nossa proposta é baseada em três classes de objetos, que tem por objetivo modelar três aspectos distintos do perfil do usuário, são eles :

- **Formação do Usuário** - Diz respeito a formação acadêmica do usuário. A estrutura deste objeto contém as seguintes informações:
  - **Código do Usuário**
  - **Código de Instrução**
  - **Descrição**
  - **Pontuação**
- **Utilização do Sistema** - Descreve que recursos do sistema foram utilizados e o status da utilização. Informações do objeto:
  - **Código do Usuário**
  - **Lista de Cenários Utilizados**- cada cenário da lista vem com um dos seguintes status:
    - **A** - Usou corretamente;
    - **B** - Usou mas cometeu erros;
    - **C** - Usou com ajuda de help;
    - **D** - Usou e cancelou, ou seja, não concluiu o cenário.
  - **Lista de Caminhos Utilizados** - Status semelhante ao de cenário.
  - **Lista de Eventos Executados** - Status semelhante ao de cenário.
  - **Pontuação**
- **Nível de Conhecimento** - Este objeto está relacionado aos conceitos pertencentes ao domínio da aplicação. De acordo com a área da aplicação, os conceitos questionados ao usuário serão distintos, visando modelar os conhecimentos do usuário na área. Informações mantidas neste objeto:
  - **Código do Objeto**
  - **Lista de Conceitos e Definições** - Esta lista contém as definições e conceitos que são conhecidos do usuário, com seus respectivos status:
    - **A** - Conhece e utiliza objeto
    - **B** - Conhece e já utilizou objeto
    - **C** - Já ouviu sobre o objeto
    - **D** - Desconhece objeto
  - **Pontuação**

Baseada nas informações dos objetos do usuário a interface se adapta ao seu perfil, cooperando na execução das atividades e fornecendo ajuda diferenciada para cada um.

5 Funcionamento da Interface

O funcionamento da interface é baseado na ocorrência de eventos, que podem ser oriundos do sistema ou do usuário. Vamos analisar como é o funcionamento da interface através da entrada de ações do usuário. O esquema geral de funcionamento é mostrado na figura 4.

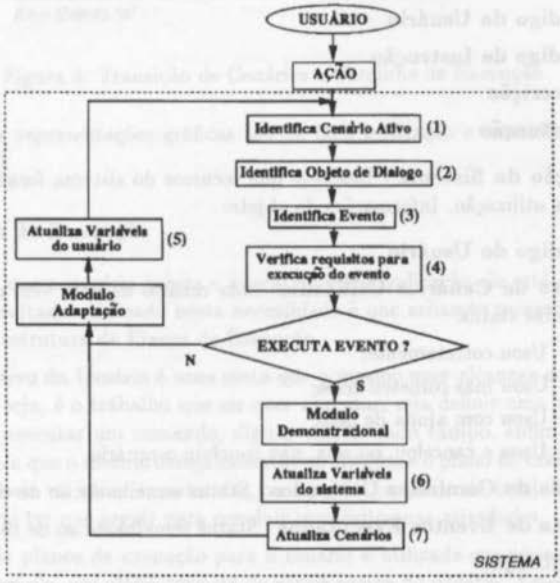

Figura 4: Esquema do funcionamento da interface

Ao utilizar um cenário o usuário identifica para o sistema qual o cenário ativo, sendo este composto por objetos de diálogo que são a base da comunicação na interface. O evento é identificado com base no objeto que o usuário estiver executando, ou seja, um menu, um ícone, uma área de texto ou uma imagem.

Inicializado o evento, o sistema verifica os requisitos para que o mesmo ocorra, checando valor das variáveis de controle que são mantidas na tabela de controle de eventos. Caso o evento esteja habilitado para execução, será ativado o módulo demonstracional, que será visto na próxima seção, atualizadas as variáveis e cenários do sistema assim como as variáveis do usuário. O módulo de adaptação será ativado de acordo com a execução do usuário. Este módulo também será descrito posteriormente.

5.1 Demonstrabilidade

Uma das características mais importantes neste sistema de interfaces é a demonstrabilidade. Esta funcionalidade é obtida devido a forma como é modelada e o modo como funciona esta arquitetura.

A especificação de uma interface nesta arquitetura é completamente realizada em tabelas, sejam elas de controle, construção ou de objetos de diálogo. Esta característica representa uma vantagem muito grande uma vez que possibilita uma fácil recuperação de qualquer informação a respeito de qualquer ponto da interface na qual o usuário esteja executando suas atividades. Isto é feito identificando o objeto de diálogo que está sendo utilizado, os eventos acionados, o caminho de execução que está sendo percorrido e outras informações adicionais necessárias.

Para que uma interface funcione por demonstrabilidade é necessário que a mesma identifique qual ação o usuário está executando e onde. Desse modo o sistema assume, quando possível, a execução daquela tarefa e deixa sobre a responsabilidade do usuário apenas a confirmação de alguns eventos e transição de cenários, assim como a entrada de dados.

A partir do momento em que o usuário vai percorrendo cenários, o número de caminhos de execução possíveis de serem seguidos a partir do cenário ativo vai diminuindo, até o ponto em que resta apenas um caminho a ser seguido pelo usuário, é então quando o sistema questiona ao usuário se é sua vontade prosseguir conforme o caminho proposto. Caso o usuário confirme o caminho de execução, o sistema continuará automaticamente abrindo e fechando os próximos cenários, cabendo ao usuário apenas a entrada de dados ou algumas confirmações.

Imaginemos por exemplo um sistema com os seguintes caminhos de execução:

- Cam001 = {C1,C2,C6,C18}
- Cam002 = {C1,C2,C7,C19}
- Cam003 = {C1,C3,C8,C20}
- Cam004 = {C1,C3,C9,C21,C22}
- Cam005 = {C1,C4,C23}
- Cam006 = {C1,C4,C24,C25}

Se o usuário está no cenário C1, verificamos que o mesmo pode seguir qualquer um dos seis caminhos, no entanto se ele prossegue para para o cenário C3, o mesmo agora só poderá seguir por dois caminhos, são eles o Cam003 e o Cam 004. Continuando a execução o usuário segue para o cenário C9, o que deixa o sistema com o leque de opções reduzido a um único caminho de execução a ser seguido, que é o Cam004. Neste caso o sistema executará automaticamente os cenários C21 e C22, realizando as suas transições, deixando para o usuário apenas a entrada de dados.

Desta forma o sistema torna-se uma agente antecipador de execução, predizendo os cenários a serem executados pelo usuário, diminuindo desta forma o trabalho a ser realizado pelo mesmo.

Existe uma outra ocasião em que o sistema também prediz cenários a serem per-

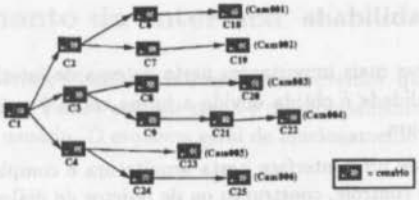

Figura 5: Grafo esquematizando os caminhos de execução

corridos. Neste outro caso, quando um usuário utiliza um cenário, é verificado quais os próximos cenários que podem seguí-lo, dessa forma, para cada transição possível iniciando no cenário ativo atual, é verificada a frequência de utilização da mesma pelo usuário.

Esta frequência de utilização corresponde ao número de vezes que o usuário já executou tal transição em sessões anteriores no sistema. Se a frequência de uma das transições for acima de uma frequência mínima estabelecida previamente, esta transição povavelmente irá ocorrer de novo. Caso haja mais que uma transição possível com frequência maior que a frequência mínima, será escolhida a de maior frequência, sendo então questionada ao usuário se ele deseja seguir ou não tal transição.

O fluxograma da figura 6 ilustra como a desmonstrabilidade é implementada, onde são utilizadas as seguintes operações :

prox(Ci) - Retorna uma lista contendo todos os cenários que seguem-se imediatamente após Ci;

ant(Ci) - Retorna uma lista contendo todos os cenários imediatamente anteriores ao cenário Ci;

#(lst) - Retorna o número de itens da lista lst;

FT(Ci,Cj) - Retorna a frequência de utilização da transição de Ci para Cj pelo usuário;

CenUso - Retorna uma lista contendo todos os cenários abertos no sistema.

CA - Cenário Ativo (o que o usuário está utilizando);

Cam(C) - Retorna uma lista de Caminhos de Execução nos quais o cenário C está presente;

FTM - Frequência de Transição Mínima.

A demonstrabilidade do sistema é baseada então em duas circunstâncias, são elas:

- Quando a partir do cenário ativo houver apenas um cenário a ser seguido ou quando existir apenas um caminho de execução a percorrer. No primeiro caso o sistema pode antecipar apenas o cenário seguinte, no segundo caso o sistema antecipa todos os cenários restantes contidos no caminho de execução selecionado;
- Quando existem vários cenários Cj's a serem seguidos pelo usuário a partir do cenário CA, no entanto existe um ou mais cenários nos quais a FT(CA,Cj)>FTM. Neste caso o sistema escolherá para sequenciar o cenário atual aquele que corresponder a maior frequência de transição, ou seja, a transição mais utilizada pelo usuário.

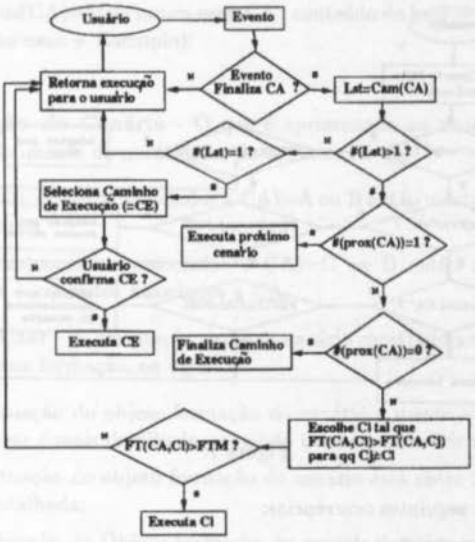

Figura 6: Esquema de funcionamento do algortimo para demonstrabilidade

5.2 Adaptabilidade

Adaptabilidade é a qualidade de ser adaptável, é a capacidade de se adaptar. Esta á uma característica importante num sistema, sua presença se faz notar através de uma interface mais flexível, que se ajusta as necessidades do usuário, levando em consideração as características apresentadas no seu perfil e que são atualizadas a cada interação com o sistema [Ben93].

Existem várias formas de adaptação, no entanto a nossa preocupação neste trabalho é a adaptação a nível de estilos de diálogo, através da oferta de vários estilos de diálogo para realizar uma mesma tarefa. Esta funcionalidade é possível devido a utilização de cenários sinônimos, que por definição são cenários que têm a mesma função na interface, apresentando-se no entanto com objetos de diálogo distintos. Enquanto um cenário apresenta-se baseado em menus, outro cenário sinônimo seu pode se apresentar em forma de ícones, e um outro em linguagem de comandos, por exemplo.

A figura 7 apresenta uma visão simplificada de como funciona o algoritmo de adaptação da interface, tendo as seguintes definições:

- **FMH** - Frequência máxima de chamadas ao help;
- **FME** - Frequência máxima de erros num cenário;
- **FMC** - Frequência máxima de cancelamento de cenário;
- **FEC(Ev,Ci)** - Frequência do evento Ev no cenário Ci.

Segundo a fluxograma, uma adaptação sempre é sugerida ao usuário como conse-

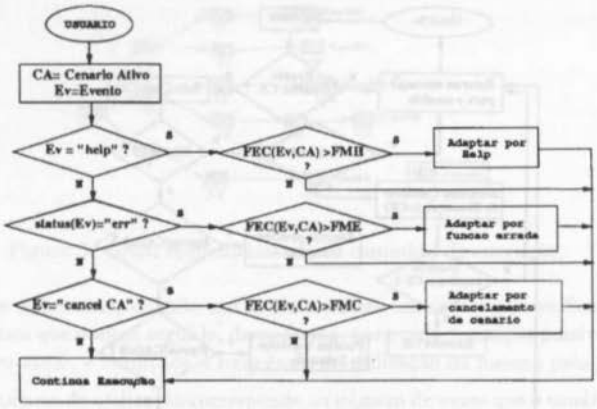

Figura 7:

quência de uma das seguintes ocorrências:

- Utilização excessiva de *help*;
- Execução de eventos que representem ações inválidas várias vezes num mesmo cenário;
- Cancelamento frequente de um mesmo cenário;

Estas são, a princípio, as situações mais comuns nas quais o usuário poderia estar insatisfeito com o cenário, o que nos leva a questionar se é vontade dele mudar o cenário por outro equivalente, o que caracteriza um cenário sinônimo.

5.3 Cooperação e Interação

Esta característica é mais uma consequência natural das duas primeiras, pois existe todo um diálogo com o usuário sempre que identifica-se uma situação crítica de execução, seja identificando caminhos de execução a serem seguidos ou adaptando cenários.

Um dos pontos importantes da cooperação é a oferta de ajuda diferenciada a cada usuário, baseando-se no perfil particular de cada um para atender a real necessidade dos mesmos.

Quando uma ajuda é solicitada em determinado cenário, esta é montada com base nos objetos do usuário, de forma que:

- Verifica-se o status do cenário com relação ao objeto utilização da modelagem do usuário:
  - Se status(CA)=A : conteúdo do help = (Descrição do cenário);
  - Se status(CA)=B ou C : conteúdo do help = (Descrição do cenário + Como usar);

- Se status(CA)=D ou nunca usou CA : conteúdo do help = (Descrição do cenário + Como usar + Exemplo);

Onde:

Descrição do Cenário - O que é apresentada ao usuário vai depender do objeto conhecimento da modelagem do usuário, tal que:

- Se status(conceitos associados a CA)=A ou B então mostrar apenas a descrição do cenário;
- Se status(conceitos associados a CA)=C ou D então mostrar descrição do cenário + conceitos associados a CA.

Como Usar - A orientação dada ao usuário mostrando como usar o cenário é baseada na sua formação, ou seja:

- Se pontuação do objeto formação do usuário é menor que 20, a descrição de como usar é mais detalhada, seguindo também uma orientação mais alongada;
- Se pontuação do objeto formação do usuário está entre 20 e 50, a descrição é mais detalhada;
- se pontuação do Objeto formação do usuário é maior que 50, a descrição de como usar é bem resumida.

6 Considerações Finais

Embora a descrição das estruturas da interface, assim como os algoritmos, não tenham sido completamente descritos, é visível a facilidade de uso que se obtém ao defini-la segundo esta arquitetura, pois consegue-se reunir várias funcionalidades numa só interface, habilitando-a adaptar-se ao perfil particular de cada usuário, fornecer-lhe ajuda conforme seus conhecimentos, formação e uso do sistema, assim como prever execuções. Desta forma obtém-se várias vantagens, entre elas:

- Diminuição de tomada de decisões;
- Adaptação da interface ao usuário, aumentando a satisfação do mesmo, consequentemente aumentando a sua produtividade;
- Cooperação no sentido de encaminhar o usuário para o seu objetivo no sistema;
- Ajuda personalizada para cada usuário, de acordo com o seu perfil.

Convém lembrar que a facilidade de uso se estende também para quem vai projetar a interface, tendo em vista a possibilidade de se construir uma ferramenta para prototipagem rápida.

Atualmente está sendo implementado um protótipo em C++ desta arquitetura, direcionado para ambiente de banco de dados, utilizando o E/D [Ban89] como modelo de dados e incorporando todas as características aqui apresentadas.

Referências

[And88] N. Bjorn Andersen. Are "human factors" human ? *Computer Journal*, 31(5), 1988.

[Ban89] Monica S. Bandeira. Modelagem estatica e dinamica de sistema de informacao. Master's thesis, UFPE - DI, 1989.

[Ben93] David Benyon. Adaptative systems: A solution to usability problems. *User Modeling and Users-Adapted Interaction*, (3):65–87, 1993.

[Ben94] P. A. Bennett. Safety. In Butterworth Heinemann, editor, *Software Engineer's - Reference Book*, chapter 60. Jonh MacDermid, 1994.

[Cam91] Luis Eduardo Saraiva Camara. Sicop: Um sistema de interfaces cooperativo para modelagem em banco de dados. Master's thesis, Departamento de Informática - Universidade Federal de Pernambuco, agosto 1991.

[Dow91] Andy Downton. *Engineering the Human-Computer Interface*. 1991.

[FG90] Gerhard Fisher and Andreas Girgensohn. End-user modifiability in design environments. *CHI'90*, pages 183–192, 1990.

[FN95] Decio Fonseca and Otaviano C Wanderley Neto. Interfaces. *Tutorial submetido ao IX Simposio Brasileiro de engenharia de Software*, 1995.

[Gui94] Nuno Guimaraes. Interactive systems, tools and applications. IX Escola de Computacao, Julho 1994. Recife - PE.

[HH89] H. Rex Hartson and Deborah Hix. Human-computer interface development:concepts and systems for its management. *ACM Computing Surveys*, 21(1):5–92, marco 1989.

[JH93] Andrew Jennings and Hideyuki Higuchi. A user model neural network for a personal news service. *User Modeling and User-Adapted Interaction*, 3:1–23, 1993.

[MMK93] Brad A. Myers, Richard G. McDaniel, and David S. Kosbie. Marquise: Creating complete user interfaces by demonstration. *ACM, INTERCHI'93*, pages 293–300, 1993.

[Mye92] Brad A. Myers. Demonstracional interfaces: A step beyond direct manipulation. *IEEE*, August 1992.

[NdG93] Mark Neerincx and Paul de Greef. How to aid non-experts. *ACM, INTERCHI'93*, pages 165–171, 1993.

[Pfl91] Shari Lawrence Pfleeger. *Software Engineering - The Production of Quality Software*. 2 edition, 1991.

[PR94] Gerhard Peter and Dietmar Rosner. User-model-driven generations of instructions. *User Model and User Adapted Interaction*, 3(4):289–319, 1994.

[Ret92] Marc Retig. Interface design when you don't know how. *Communications of the ACM*, 35(1):29–34, 1992.

[Som92] Ian Sommerville. *Software Engineering*. 4 edition, 1992.

[Sto94] David Alan Stokes. Requeriments analysis. In Butterworth Heinemann, editor, *Software Engineer's - Reference Book*, chapter 16. Jonh McDermid, 1994.

[WF91] David Wobler and Gener Fisher. A demonstrational technique for developing interfaces with dynamically created objects. In ACM press, editor, *UIST - Fourth Annual Symposium on User Interface Software and Technology*, pages 221– 230, 1991.