UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FAI - FERRAMENTA DE AVALIAÇÃO DE INTERFACE COM O USUÁRIO NO AMBIENTE AGILE

Lauro Yasumasa Nakayama

Campina Grande - PB
Maio - 1995

FAI - FERRAMENTA DE AVALIAÇÃO DE INTERFACE COM O USUÁRIO NO AMBIENTE AGILE

Lauro Yasumasa Nakayama

FAI - FERRAMENTA DE AVALIAÇÃO DE INTERFACE COM O USUÁRIO NO AMBIENTE AGILE

Dissertação apresentada ao Curso de Mestrado em Engenharia Elétrica da Universidade Federal da Paraíba, em cumprimento as exigências para a obtenção do grau de Mestre.

Área de Concentração: Processamento da Informação

Maria de Fátima Queiroz Vieira Turnell, PhD
Orientadora

Campina Grande - PB
Maio - 1995

N163f Nakayama, Lauro Yasumasa.
FAI : ferramenta de avaliação de interface com o usuário no ambiente agile / Lauro Yasumasa Nakayama. - Campina Grande, 1995.
102 f.

Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1995.
"Orientação : Profa. Dra. Maria de Fátima Queiroz Vieira Turnell".
Referências.

1. Interface - Avaliação. 2. Interface com o Usuário. 3. Ambiente Agile. 4. Dissertação - Engenharia Elétrica. I. Turnell, Maria de Fátima Queiroz Vieira. II. Universidade Federal da Paraíba - Campina Grande (PB). III. Título

CDU 004.5(043)

FEI - FERRAMENTA DE AVALIAÇÃO DE INTERFACE COM O USUÁRIO NO AMBIENTE ARGILE

LAURO YASUMASA NAKAYAMA

Dissertação aprovada em 29.05.1995

MARIA DE FÁTIMA QUEIROZ VIEIRA TURNELL, Ph.D, UFPB
Orientadora

EDILSON FERNEDA, Dr., UFPB
Componente da Banca

MISAEL ELIAS DE MORIAS, Dr.Ing., UFPB
Componente da Banca

CAMPINA GRANDE - PB
Maio - 1995

Agradecimentos

À professora Maria de Fátima Queiroz Vieira Turnell, pela orientação criteriosa, pelas recomendações sempre pertinentes e por não ter medido esforços no sentido de ver concluído este trabalho.

À "República dos Inocentes", através de Vicente, Isaac, Luiz Maurício, Fernando e Cristiana, pelo ambiente de estudo.

Aos amigos Dr. José Vicente, Barbara, Skaiko, Carol, Rita de Cassia e dona Antônia, pelo apoio prestado durante todo o mestrado.

A Maria José, pela força de seu trabalho na nossa residência; sem a sua ajuda, seria muito difícil a conclusão do curso.

À amiga Cláudia Procópio, pela ajuda significativa durante todo o processo de implementação do Projeto.

A Sheyla, pelo carinho e compreensão e por ter acreditado na conclusão deste trabalho.

Ao Professor Wilson Guerreiro Pinheiro, pela revisão do manuscrito e, sobretudo, pela amizade inestimável.

À minha irmã Celeste, pelo apoio quotidiano e estímulo que me impulsionam sempre a prosseguir.

A todos que deste trabalho fizeram ou venham a fazer uso, para aperfeiçoar um conhecimento já adquirido ou buscar novos conhecimentos, porque é o retorno do tempo e esforço dedicados.

A vida nos retrata a forma de atingirmos nossos objetivos;

A minha mãe, em especial, pela lição de otimismo, que é uma constante na sua vida.

Sumário

5 Estudo de caso

6 Conclusões

LISTAS DE FIGURAS

RESUMO

Nos ultimos anos, a Engenharia de Software tem-se voltado para o desenvolvimento de técnicas e ferramentas que facilitem o projeto de interfaces e aumentem as chances de aceitação do produto por parte do usuário.

Nesta dissertação, apresenta-se uma ferramenta de software capaz de avaliar quantitativamente a qualidade de uma interface com o usuário, modelada ou prototipada no ambiente AGILE [BRANCO 89 , PROCÓPIO 91 e SANTOS 91]. A avaliação tem o propósito de detectar os pontos críticos, tanto na fase de projeto quanto na reengenharia de produtos já desenvolvidos, a partir da determinação do nível de qualidade do software e da identificação de pontos de estrangulamento.

Na versão apresentada neste trabalho, esta ferramenta de avaliação não oferece mecanismos de inferência para analisar os dados coletados, limitando-se a oferecer informações de desempenho para que o analista tome as suas próprias decisões.

ABSTRACT

In the past few years, the Software Engineering Science has turned its attention to the development of tools and techniques to simplify and improve the development of user interfaces therefore raising the chances of product acceptance by the user.

This dissertation presents a software tool for evaluating in quantitative terms the quality of a user interface which has been modeled or prototyped in the AGILE environment [BRANCO 89 , PROCÓPIO 91 e SANTOS 91]. The evaluation has as the major purpose to detect the week points introduced in the product development as well as re-engineering already developed products based upon the establishment of the software quality level and the identification of botlenecks.

In the version presented here, this evaluation tool does not offer infference mechanisms to analyse the captured data, but it limits itself to produce performance information to support the analyst decisions.

CAPÍTULO 1

INTRODUÇÃO

Capítulo 1
INTRODUÇÃO

Nos ultimos anos, a Engenharia de Software tem-se voltado para o desenvolvimento de técnicas e ferramentas que facilitem o projeto de interfaces e aumentem as chances de aceitação do produto por parte do usuário.

A avaliação, qualitativa e quantitativa, de interfaces com o usuário é um tema novo e tende a ser explorado com grande ênfase doravante. Isto se deve ao fato de a tecnologia dos computadores (hardware) vir evoluindo mais rapidamente do que a dos sistemas neles implementados, e também à crescente utilização destes sistemas por pessoas sem formação especializada (interfaces para o grande público).

1.1 Motivação

Indiscutivelmente, o usuário avalia o software através de sua interface. Por esta razão, os engenheiros de software estão investindo, cada vez mais, em projetos de construção de interfaces de alta qualidade.

Na melhor das hipóteses, uma interface mal projetada resultará em custos adicionais de treinamento do usuário e numa grande proporção de erros por ele cometidos. Portanto, a construção de uma boa interface requer muitas revisões, avaliações e aperfeiçoamento do projeto. Para tanto, é necessário testar o produto como um todo, ou em parte, com diferentes tipos de usuário, como parte da metodologia de validação [COX, WALKER 93].

Podem-se citar dois grandes fatores que motivaram este trabalho. O primeiro deles foi a ausência de uma ferramenta automatizada que permitisse a avaliação de projetos de interface homem-máquina. O segundo foi a necessidade

de testar o projeto como um todo, ou em parte, com diferentes tipos de usuário, para posteriormente o analista/projetista validar ou não o seu projeto conforme planejado.

1.2 Justificativa para o Trabalho

Os modelos e métodos de avaliação analisados na literatura levam em consideração resultados das linhas de pesquisa baseadas em experimentos com técnicas de interação, particularmente com os resultados que exploram os aspectos de fatores humanos, por ser a área que tem trazido mais subsídios. A maioria destes resultados origina-se da observação, análise e medida de desempenho do elemento humano no sistema, e discorre sobre suas capacidades e limitações [ESTEVAM 90], [TULLIS 85], [SHNEIDERMAN 83] e [GALITZ 88] investigadas através dos seguintes tópicos: qualidade da apresentação da informação (visual ou auditiva), grau de controle do usuário sobre o sistema, ntre outros. O objetivo maior do estudo de fatores humanos é atingir, através de projetos apropriados, uma maior eficiência funcional do conjunto equipamento-usuário.

Como exemplos de sistemas de avaliação automatizada que mais se aproximam do presente contexto, podem-se citar as ferramentas Benchmark, que são o referencial existente no atual mercado de software.

1.3 Objetivos do Trabalho

A ferramenta apresentada nesta dissertação objetiva oferecer ao usuário projetista de interface no ambiente AGILE (prototipador de interface), [PROCÓPIO 92] e [SANTOS 92], informações gerenciais para uma tomada de decisão baseada na avaliação quantitativa de um protótipo de interface homem-máquina.

1.4 Organização do Texto

O texto desta dissertação está organizado conforme descrito a seguir.

No Capítulo 2, é apresentada uma visão global de ferramentas benchmark e os tipos existentes que serviram como referência para a concepção da ferramenta de avaliação, uma vez que são os sistemas de avaliação automatizada que mais se aproximam do contexto da ferramenta.

No Capítulo 3, são descritas as técnicas de avaliação de interfaces: qualitativas e quantitativas, assim como os objetivos da qualidade e os fatores de qualidade utilizados na análise dos dados coletados pela ferramenta.

No Capítulo 4, é apresentada uma breve introdução ao ambiente AGILE, seguida da descrição funcional e arquitetural da FAI - Ferramenta de Avaliação de Interface com o usuário no ambiente AGILE. Nele, ainda são descritos detalhes da implementação da ferramenta e de sua interface com o usuário.

No Capítulo 5, é apresentado um estudo de caso com base em um sistema real, onde é destacado o potencial da ferramenta na reengenharia deste produto, modelado no ambiente AGILE. A partir da determinação do nível de qualidade do software e da identificação de pontos de estrangulamento, com base nos dados coletados, foi possível fazer modificações no produto que levaram a uma comprovada melhora no desempenho do sistema usuário-computador na realização das tarefas sob análise.

Finalmente, no Capítulo 6, são retomados os objetivos do trabalho, discutindo o potencial e as limitações da ferramenta a partir do estudo de casos, além de propor direções para trabalhos futuros.

No Apêndice **A,** traz a lista-se o arquivo AGILE ".PTP", que contém todas as informações de um protótipo de interface modelado no ambiente AGILE.

No Apêndice **B,** apresenta-se o trecho do código FAI que cria o vetor de objetos com os seus atributos e variáveis de instância.

No Apêndice **C,** apresentam-se as estruturas de dados do AGILE que a FAI utiliza para coletar os dados durante uma avaliação.

CAPÍTULO 2

SISTEMA BENCHMARK

SISTEMAS BENCHMARK

Para conceituar sistemas de benchmark, será utilizada a descrição de uma situação. Vários problemas podem surgir em um centro de processamento de dados, o mais comum dos quais é a falta de terminais para novos sistemas [HINNANT 88]. Nesta situação, o gerente teria de dimensionar a instalação de novos terminais. Como ele saberia determinar quantos terminais poderiam ser instalados no equipamento sem prejuízo do tempo de resposta dos sistemas existentes? Uma abordagem do problema seria a simulação da carga dos novos terminais e de seus usuários que, rodando no ambiente, gerasse relatórios de desempenho (tempo de resposta, carga da CPU etc.) que o auxiliassem no processo de tomada de decisão. Algumas dessas características foram incorporadas na modelagem do sistema FAI. Esse tipo de abordagem é denominado teste de **Benchmark** e será descrito a seguir.

2.1 Visão Global de Sistemas Benchmark

O Sistema Benchmark é uma ferramenta utilizada para avaliar desempenho de um sistema computacional (hardware) ou de aplicativos de um software [ROYBAL 90]. Os benchmarks são bem distintos nas suas características, tais como seus componentes, suas aplicações e a metodologia utilizada para avaliação de desempenho dos sistemas sob análise.

Em cada avaliação de um sistema, é testada sua produtividade como um todo, através da medição da quantidade total do processamento realizado pelo hardware. Um outro ponto é a investigação do desempenho que se torne crítico a cada nova carga de trabalho submetido, isto é, quando o hardware é compartilhado para executar mais de uma tarefa.

Um teste benchmark também avalia partes específicas de um sistema, como, por exemplo, desempenho de CPU, "system call", discos etc., fornecendo parâmetros válidos para efetuar comparações.

Pode-se ter uma idéia do funcionamento de um sistema de benchmark através da figura 2.1 [WYRICK 78].

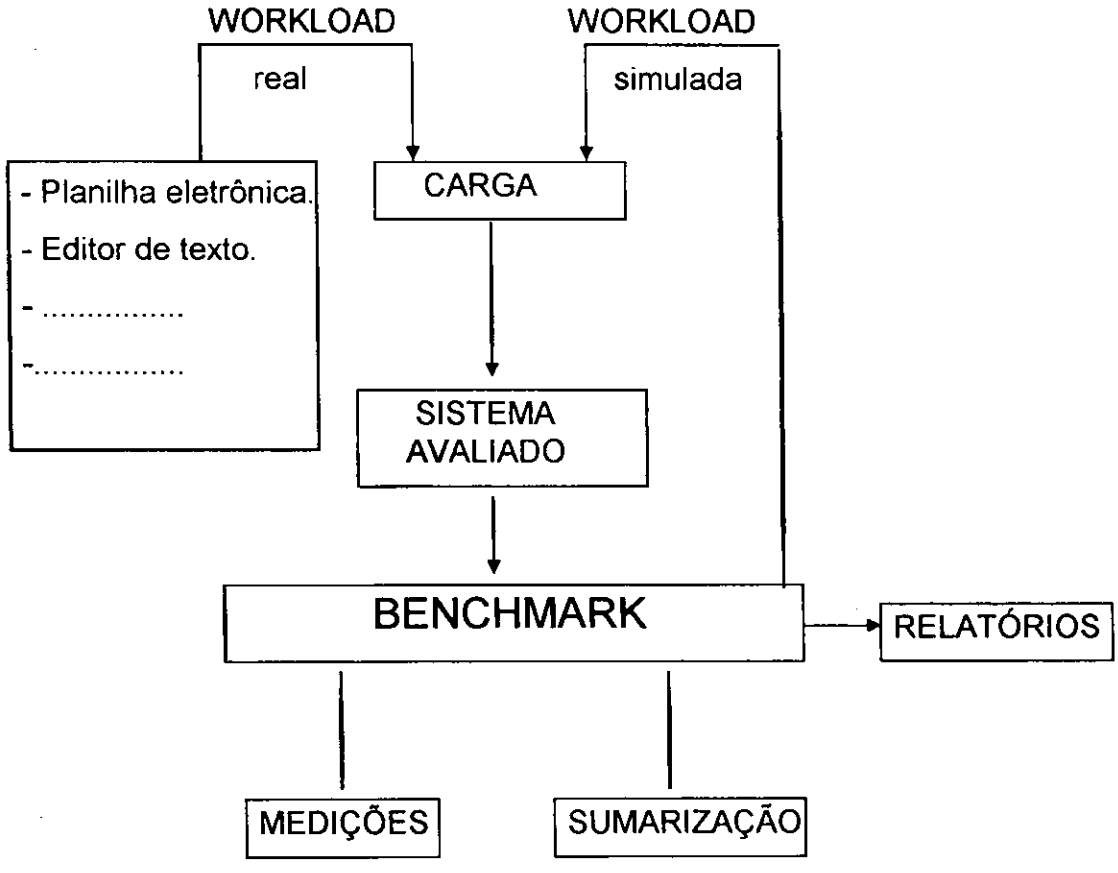


Figura 2.1 : Arquitetura de um sistema benchmark

A "workload" (carga) é definida como uma quantidade ou carga de trabalho imposta ao sistema computacional pelo usuário.

As medições são executadas através de software e, delas, se obtêm dados em função da carga aplicada ao sistema avaliado durante um determinado tempo. A sumarização dos dados utiliza uma metodologia baseada nas médias aritméticas, geométricas e harmônicas dos dados coletados ou medidos.

Na geração dos relatórios, existe a necessidade de um único número para caracterização de desempenho, o que é "um mal necessário" e muito polêmico. Segundo [HINNAMT 90 e SMITH 88], o desempenho é multidimensional e não pode ser precisamente representado com um simples número.

A ferramenta FAI, apesar de não seguir os modelos de benchmark tradicionais, tem as mesmas bases filosóficas, isto é, coletar dados para estabelecer níveis de desempenho, permitindo a comparação entre soluções alternativas.

Assim, estabelecem-se métricas para os fatores quantificáveis que permitem a avaliação de um projeto de interface em função do usuário.

Vale a pena salientar que a "workload"(carga) utilizada neste trabalho é real, em contraste com as cargas simuladas dos sistemas de benchmark.

2.2 Tipos de Sistemas Benchmark

Com o intuito de fornecer resultados representativos, foram desenvolvidos diferentes estilos de benchmark. Segundo [ROYBAL 90], estes estilos podem ser classificados nos três tipos descritos a seguir:

1 - *Benchmark de aplicação* - Simula uma aplicação real do usuário, usando segmentos do código da aplicação real em uma *workload* (carga). Para modelar o comportamento da aplicação, este benchmark reproduz, pelo menos, três fatores:

a) A variedade de instruções de uma aplicação;

b) o tamanho dos recursos consumidos pela aplicação;

c) padrões de acesso à memória e disco.

2 - *Emulação de Terminal Remoto (ETR)* - Uma ETR simula um usuário no seu terminal, conectado a um sistema computacional. É uma maneira direta de se medir o desempenho de um sistema, podendo ainda ser classificado em ETR externo ou interno.

ETR externo: Emula um terminal através de um dispositivo de hardware. Esse dispositivo é normalmente um outro computador, que é considerado uma caixa preta de algum tipo. Os ETRs enviam os comandos do usuário a partir do hardware externo para o computador que está sendo avaliado - System Under Test (SUT).

ETR interno: Não há um hardware externo, os comandos e o tempo de resposta são gerados e medidos internamente pelo SUT; portanto, medidas como tempo de eco de caracteres não fazem mais sentido, mas medidas de tempo de resposta e velocidade continuam válidas.

3 - *Modelagem Load/Mix* - Simula uma ou várias aplicações, a fim de testar o desempenho dos recursos de um sistema. Este tipo de benchmark impõe uma determinada carga ao sistema avaliado, simulando um "mix" subconjunto de instruções que as caracterize. O "mix" é definido pelo usuário a partir da seleção dos tipos de operações e funções, com os seus respectivos percentuais relativos.

Por exemplo, um gerente de sistemas em um ambiente de desenvolvimento de software poderia escolher o seguinte mix de aplicação [MARVIT 84]:

- 60% de compilação;
- 40% de processamento de texto.

Enquanto que um gerente financeiro escolheria o seguinte mix:

- 70% de cálculos;
- 20% de planilha eletrônica;
- 10% de processamento de texto.

Com esta variação do mix de aplicação, os gerentes poderiam avaliar o desempenho dos sistemas de acordo com as suas necessidades.

A literatura sobre benchmark consiste, predominantemente, de estudos de caso [MARVIT 84]; portanto, seus fundamentos teóricos são um tanto obscuros. A seguir, são apresentados alguns testes clássicos de benchmark :

Whetstone:- Este é um dos mais antigos testes, desenvolvido para analisar programas científicos típicos. É composto de vários módulos, cada qual com instruções de uma classe em particular, por exemplo: aritmética inteira, aritmética de ponto flutuante, comandos "if", comandos de chamadas a subrotinas etc. Pesos diferentes são associados a cada módulo, de modo que a distribuição das instruções do "whetstone" para o benchmark corresponda à distribuição observada no programa-exemplo [ROYBAL 90].

Os resultados são fornecidos em *KWIPS* (kiloinstruções whetstone por segundo) ou *MWIPS* (megainstruções whetstone por segundo). Este teste foi inicialmente considerado muito bom, mas análises recentes mostraram que ele é vulnerável às otimizações dos compiladores modernos.

Dhrystone:- Mede o desempenho do sistema, baseado na análise do uso de programas reais, avaliando o desempenho da CPU e a eficiência do compilador. Utiliza basicamente análise estatística de programas típicos, dando resultados em dhrystone por segundo, unidade de tempo utilizada para medir velocidade da CPU [ROYBAL 90].

Sieve:- Mede a eficiência ou capacidade do sistema, no que diz respeito a referências feitas à memória, comandos de estrutura de controle simples e operações inteiras, gerando números primos para testar o desempenho global do sistema [ROYBAL 90].

SI (Norton System Indicator):- Originalmente, este teste pretendia mostrar as diferenças de velocidades entre os vários elementos de hardware da família 8088. É muito dependente da arquitetura do equipamento e fornece resultados inválidos quando usado para comparar elementos de gerações diferentes, como 8088 x 80286 [ROYBAL 90].

Limpack:- Inicialmente, consistia apenas de um pacote de subrotinas de álgebra linear, usado em programas Fortran; depois, foi transformado em benchmark numérico, pois possui um alto percentual de operações de ponto flutuante, mas com uma variação pequena de funções chamadas. Os resultados são apresentados em milhões de operações de pontos flutuantes por segundo (MFLOPS) [ROYBAL 90].

Spec Mark:- Esta ferramenta de teste é uma prova do esforço cooperativo mais importante em benchmark, para avaliação de desempenho de sistemas. Surgiu da necessidade de um benchmark mais adequado à avaliação de sistemas, pois os que existiam se tornaram pequenos e obsoletos pelo avanço do hardware. Dessa forma, o Spec Mark ficou impossibilitado de gerar resultados representativos para programas reais que utilizem memória cache ou grandes áreas de memória [ROYBAL 90].

O objetivo da ferramenta é coletar, padronizar e distribuir programas de larga aplicação que possam ser usados como programas de benchmark. O usuário, por sua vez, pode selecionar a unidade de medida que melhor represente a sua aplicação. Vale ressaltar que os resultados de desempenho são fornecidos em relação ao equipamento VAX 11/780.

2.3 Uma Proposta de Benchmark de Interface (FAI)

O consenso entre os projetistas de interface é que a satisfação do usuário é o objetivo primordial do produto de seu trabalho. Este objetivo, para ser alcançado, depende da escolha adequada dos "fatores de qualidade" utilizados para avaliar o produto de acordo com a classe de usuário.

Neste trabalho, a escolha destes fatores é dirigida para o uso da ferramenta AGILE (prototipador rápido de interface) [PROCÓPIO 92] e [SANTOS 92] para que, durante a simulação de um protótipo de interface, seja possível coletar dados visando quantificar alguns fatores de qualidade. Com base nestas informações, o projetista decide se o protótipo atinge ou não seus objetivos de qualidade - benchmark de interface.

Aqui, é apresentada uma Ferramenta de Avaliação de Interface - FAI, capaz de realizar a avaliação quantitativa dos protótipos de interface gerados no ambiente AGILE .

Esta ferramenta se baseia na filosofia dos sistemas benchmarks apresentados anteriormente. Embora todas as ferramentas "benchmark" tenham características muito própria, todas elas medem o tempo de resposta de alguma tarefa submetida à ferramenta. Então, o tempo de resposta é o elo comum entre todos os "benchmarks" clássicos e o aqui proposto.

2.4 Classificação da FAI

Deve-se classificar a FAI como um tipo especial de benchmark, uma vez que não utiliza modelos matemáticos para quantificar tarefas, como fazem os tradicionais benchmarks. Em contrapartida, a FAI quantifica as operações executadas dentro de uma interação. Por exemplo, verifica quantas vezes o usuário solicitou ajuda para uma determinada tarefa, quanto tempo o usuário levou para responder a uma ação gerada na interface, etc. Estes aspectos são abordados no Capítulo 5.

Outra característica que a torna um tipo especial de benchmark é a consideração dos "fatores humanos". Os projetistas de interface encontram na pesquisa de fatores humanos (um ramo da Psicologia Experimental) uma das mais ricas fontes de idéias e conceitos para o seu trabalho. Os resultados dessas pesquisas são adotados como diretrizes, e o conjunto dessas diretrizes pode dar suporte a projetos de interfaces usuário-computador de sistemas de informação em geral [QUEIROZ 94].

2.5 Ferramenta de Avaliação de Interfaces

Os mais tradicionais métodos de avaliação de interface são os questionários utilizados após a operação dos sistemas [BEZERRA 92].

Estas ferramentas são muito interessantes, mas o sucesso, ou fracasso, dos resultados depende de muitas variáveis, como, por exemplo, o entrevistador, o perfil do usuário, o operador do planilhamento, os modelos utilizados para o evento, o delineamento estatístico (escolha do modelo estatístico) etc. Porém, o mais crítico ainda é o delineamento estatístico; se estiver mal dimensionado, os dados coletados são simplesmente inúteis e os resultados não serão confiáveis.

Na literatura sobre técnicas de avaliação de interface [DOWNTON 92], [RUBIN 88] e [COX, WALKER 93], o experimento é montado em testes de usabilidade na fase de concepção ou modelagem conceitual do sistema de interface, visando a sondagem do nível de informação do modo de comunicação e do sistema como um todo.

No próximo capítulo, são descritas generalidades do modelo de avaliação e abordadas as técnicas de avaliação.

CAPÍTULO 3

MODELO DE AVALIAÇÃO

MODELO DE AVALIAÇÃO

A presença de sistemas interativos em vários segmentos profissionais é crescente nesta última década, e a preocupação dos projetistas desses sistemas em avaliar e validar o projeto de interface é diretamente proporcional a esse crescimento. O processo de avaliação tem como objetivo melhorar o grau de eficiência desses serviços, de modo a assegurar elevados índices de satisfação do usuário [LAWSON 78].

Através de estudos sobre diversos métodos de avaliação e validação de interface homem-máquina [BEZERRA, TURNELL 91], constatou-se a existência de dois grandes grupos de avaliação, o qualitativo e o quantitativo, que serão apresentados com detalhes nas seções seguintes.

3.1 Técnicas de avaliação qualitativas

Estas técnicas têm como finalidade avaliar as interfaces, utilizando métodos de representação cognitiva do usuário [ESTEVAM 90]. Os métodos de representação cognitiva abordam os seguintes aspectos:

a) Processo cognitivo;

b) estrutura cognitiva;

c) estratégias cognitivas.

- **Processo cognitivo:** é a representação dos procedimentos cognitivos necessários aos usuários, para realização de uma certa tarefa. Para projetar um modelo que represente o conhecimento da seqüência de procedimentos, é necessário observar cuidadosamente a execução de uma tarefa pelo usuário. Utiliza-se um fluxograma para representar o que foi observado e analisado.

- **Estruturas cognitivas:** representam a estrutura do conhecimento para os procedimentos cognitivos, já observado e analisado durante o processo de execução da tarefa. Para representar esta estrutura, são utilizados diagramas de árvore, que facilitam a visualização do sequenciamento das ações do usuário.

- **Estratégias cognitivas:** Após todas as observações e análise das tarefas, procura-se compreender todos os controles executados pelo usuário, nas diversas partes do conhecimento, como, por exemplo, quais são as estratégias de Gerenciamento usadas para escolher uma determinada estrutura cognitiva. A representação da estratégia cognitiva é feita através de diagramas ou redes.

3.2 Técnicas de Avaliação Quantitativas

A avaliação quantitativa tem como objetivo realizar medições dos procedimentos dos usuários durante a execução de uma tarefa, visando apoiar o desenvolvimento de sistemas de informações interativos. Para realizar tais medições, salienta-se a representação matemática do desempenho do usuário, diagnosticando gargalos, com base nas características da tarefa ou do ambiente de trabalho [DOWNTON 91].

Tais técnicas utilizam a representação do desempenho, visando estabelecer estimativas do desempenho do usuário e suas capacidades e limitações em tarefas computadorizadas. A representação ergonômica é utilizada para obter dados antropométricos (processo de mensuração do corpo humano) e biomecânicos (fundamentos mecânicos das atividades biológicas) necessários ao projeto de interface. A Ergonomia é a ciência que estuda as relações entre homens e máquinas, sua interação física, visual e sensorial, determinando as leis do trabalho, operação e movimento dessa interação, utilizando-se de estudos antropométricos que proporcionam a melhor adaptação da forma de um produto

ao tamanho de um usuário médio. Portanto, a Ergonomia busca a melhor e mais adequada interação entre homens e máquinas [SILVA 94].

A representação de simulação por computador e representação estatística visam à especificação de um modelo lógico-matemático da interação usuário-computador, também utilizada para determinar alternativas de tarefas e testes de diálogos [BEZERRA 92].

Os procedimentos estatísticos são utilizados para construir modelos baseados diretamente no comportamento do usuário. Em [TULLI 85], utilizam-se processos de agrupamento para coletar dados sobre a percepção dos usuários em relação às funções de um sistema de menus de comandos para um sistema operacional.

Para que qualquer método acima mencionado obtenha um bom rendimento, torna-se necessário definir:

a) Objetivos da qualidade;

b) fatores de qualidade de interface do usuário;

c) e os subfatores, que definem, quando necessário, atributos mais primitivos, facilitando a avaliação.

A seguir descrevem-se a descrever esses três itens, que forneceram subsídios na escolha dos fatores de qualidade que a FAI possui .

3.3 Objetivos da Qualidade

Com relação à qualidade de interface, existe uma relação de fatores (atributos ou características) que determinam a qualidade. Sendo assim, a qualidade é um conjunto de propriedades a serem satisfeitas em um determinado

grau, tal que o software de interface satisfaça às necessidades de seus usuários [ROCHA 87].

Com relação a objetivos da qualidade, podem-se citar três propriedades fundamentais que um projeto de interface deve possuir: **confiabilidade conceitual, confiabilidade de representação e usabilidade** [MARTIN 73].

Confiabilidade conceitual - Este objetivo é atingido quando se implementa satisfatoriamente o que foi projetado, isto é, quando se atende às necessidades e aos requisitos que motivaram o projeto e sua construção.

Confiabilidade de representação - Refere-se aos atributos de representação da interface do usuário, que dizem respeito à sua facilidade de manipulação pelos usuários. Este objetivo é atingido pela consideração dos fatores de compreensão, comunicação e manipulação.

Usabilidade - Não tem sentido projetar alguma coisa que não tenha nenhuma utilidade em si ou que não seja utilizável na prática. A usabilidade é determinada pelos seguintes fatores: manutenibilidade, portabilidade, operacionalidade, flexibilidade, amigabilidade, reutilizabilidade e avaliabilidade.

3.4 Fatores e Subfatores de Qualidade da Interface

Os fatores definem a qualidade da interface e os subfatores, quando necessário, definem os atributos mais primitivos que facilitam a avaliação que será vista a seguir [ROCHA 89] e [ESTEVAM 90].

Fidelidade - É necessário que a interface seja fiel, correspondendo, na íntegra, à sua especificação no que diz respeito ao modelo de diálogo usado. Este modelo de diálogo cria um modelo na mente do usuário, evidenciando

a importância da coerência do modelo usado ao longo de toda a interação homem-máquina.

Integridade - É o atributo da interface que permite confiar que o sistema tenha comportamento satisfatório em situações hostis ou quando da ocorrência de falhas do sistema ou ações incorretas ou inadequadas do usuário. Este fator é atingido através dos subfatores de robustez e segurança.

Robustez - É a característica que assegura que a interface não pode perder sua utilidade quando há falhas do sistema ou erros do "software" ou do usuário.

Segurança - Uma interface segura é aquela que, além das verificações dos procedimentos de recuperação de erros do sistema ou do usuário, deve prever, se possível, erros e permitir ao usuário desfazer ações erradas ou não desejadas. Mesmo oferecendo ajuda para prevenir ações inadequadas, devem também existir comandos, tais como:

- Stop: permite interromper um comando em execução, tornando-o sem efeito caso o usuário perceba um erro ou ação inconveniente;

- Escape: permite retornar ao módulo anterior;

- Undo : permite tornar sem efeito a ação de um comando já executado.

Ações dos tipos acima citados são utilizados pelos sistemas para permitir a guarda automática de arquivos, a gravação periódica de alterações em arquivos ou de textos (em editores de textos), a possibilidade de restaurar arquivos "deletados" durante algum tempo etc. [GALITZ 88].

A interface deve informar ao usuário, de forma clara, acerca das conseqüências de falhas do sistema ou de usos inadequados.

Compreensão - É a característica da interface que permite a seus usuários a compreensão, sem dificuldades e sem necessidade de grandes treinamentos, do que ela faz e de como devem interagir com ela. Os parâmetros para medida da compreensão são a simplicidade e a concisão.

Simplicidade - É a característica da interface que assegura a ausência de elementos complexos, pouco familiares ou muito trabalhosos. Um projeto simples é fácil de manter e usar, e tende a permanecer. A tendência natural do projetista é achar que sua interface é simples; freqüentemente, ignora que esta simplicidade deve ser construída na mente do usuário.

Concisão - É o atributo da interface que assegura que ela possua apenas aqueles recursos que tenham efetiva utilidade. Para atingir esse objetivo, deve-se cuidar para não aumentar a complexidade com o objetivo de obter a concisão.

Comunicação - É a capacidade que a interface oferece de se apresentar aos usuários sem que estes necessitem de um treinamento prévio para conseguir usá-la. Excelentes conceitos por trás de interfaces podem perder todo o seu valor se as características de comunicação forem obscuras e ineficientes. A boa comunicação é atingida através dos subfatores descritos a seguir: forma de apresentação, uniformidade da linguagem e adequação da simbologia.

Forma de apresentação - Está muito ligada à comunicação da interface: é a característica de estar em conformidade com o modelo mental do usuário e ajustada ao seu modo de comportar-se. Tem-se necessidade de conhecer o que influencia uma boa interface com o usuário.

Entre outras coisas, a interface deve conseguir manter o interesse do usuário, comunicar-se, sempre que possível, através de técnicas visuais, levar em conta o conhecimento do usuário, falar a mesma linguagem dos usuários etc.

Uniformidade da linguagem - É a característica da interface que faz com que seja usada uma mesma linguagem (sintaxe e semântica) nos diferentes contextos em que é realizada uma mesma operação. Não se pode projetar uma boa interface sem conhecer a linguagem do usuário. Isto se reflete de forma acentuada na escolha dos termos usados ao longo do diálogo, que devem sair do vocabulário do usuário e, não, do projetista.

Adequação da simbologia - É o atributo da interface onde os símbolos usados, assim como a linguagem, são adequados às metáforas ou analogias que pretendem representar. A maioria das interfaces atuais é eminentemente gráfica, baseando-se no princípio de que a efetiva comunicação é visual. Para tanto, utilizam figuras, fotografias, desenhos, fluxogramas e diagramas em geral.

A maioria dos símbolos, bem como, das palavras de nosso vernáculo tem sua origem em uma metáfora ou alguma forma de associação. As metáforas usadas pela interface, geralmente, são símbolos (desenhos) de objetos do mundo do usuário, tais como: folhas de papel, topo de uma mesa, pastas de arquivos, cestas de lixo etc. É necessário que os símbolos usados sejam familiares aos usuários da aplicação, sejam inconfundíveis e sempre utilizem o mesmo objeto para um determinado evento.

Manipulação - É o fator que retrata a facilidade de manipular a interface pelos seus diversos usuários. Ela é atingida pelos subfatores de disponibilidade, estrutura e adaptação, descritos a seguir:

Disponibilidade - É a característica da interface que assegura que ela esteja disponível aos usuários com todas as suas facilidades, a qualquer instante

que dela necessitem. A disponibilidade exige também que a interface possua "Helps" que evitem que o usuário venha a depender de terceiros ou de consultas a manuais quando necessitar de alguma operação nova ou quando tenha esquecido de algum procedimento. É importante também a estruturação dos "Helps" com vários níveis, permitindo que o usuário navegue até o nível desejado.

Estrutura - É o atributo que facilita a manipulação da interface por parte do usuário, considerando aspectos ergonométricos, fatores humanos e psicológicos. As ações físicas que o usuário terá de realizar para responder a estímulos ou ativar procedimentos da máquina não devem ser complicadas e tampouco demoradas. O projeto da interface deve levar em conta que o usuário necessita dedicar uma atenção mínima às ferramentas que tem de utilizar para obter a máxima efetividade. O mais importante não é a organização dos elementos da interface em si, e que aparecem na tela (como menus, janelas etc.), mas a organização que deve ser criada na mente dos usuários a partir desses elementos.

Adaptação - É a característica da interface que a torna útil para diferentes tipos de aplicação e para usuários com diferentes graus de experiência. A adaptabilidade refere-se à habilidade do sistema de agir de forma apropriada em um dado contexto, ajustando-se às necessidades e preferências do usuário. Existem, pelo menos, dois tipos de adaptação em interfaces, a saber:

a) Interface adaptativa, no sentido de adaptar-se ao estilo de trabalho preferido do usuário e à sua necessidade de informação. Neste caso, a ênfase é no desenvolvimento de um sistema ativo que se adapta ao usuário.

b) Interface adaptativa, no sentido de que a interface é capaz de responder diferentemente a diferentes níveis de conhecimento do usuário. Neste caso, a ênfase é no desenvolvimento de um sistema capaz de moldar-se segundo a realimentação recebida , ajustando-se e dialogando com os diversos tipos de usuários.

Manutenibilidade - É o atributo da interface que permite introduzir alterações após a sua colocação em uso, com alguma facilidade. Com certeza, uma das poucas coisas certas acerca de um produto de software é que sofrerá modificações durante o seu tempo de vida útil.

Operacionalidade - É a característica da interface que a torna adequada ao usuário, à aplicação e ao ambiente em que é usada. A facilidade de aprendizado e uso, a rapidez das respostas, a forma de apresentação do diálogo etc. são elementos importantes perante a operacionalidade, que é determinada pelos fatores de adequação do estilo de diálogo, o controle do diálogo e o sequenciamento do diálogo.

Adequação do estilo de diálogo - É a característica que leva em conta a notável diferença entre as aplicações, os ambientes em que a interface vai atuar e, sobretudo, as diferenças entre as habilidades, o background, as motivações, a personalidade e os estilos de trabalho dos usuários. A interface deve suportar diversos tipos de diálogos, como por exemplo:

. interação simples através de comandos;
. interação através de figuras;
. interação através de perguntas/respostas;
. manipulação gráfica somente com feedback léxico;
. manipulação gráfica com feedback semântico;
. manipulação direta de imagens gráficas.

Controle do diálogo - É o atributo que indica de quem é a iniciativa do diálogo, se da aplicação ou do usuário. Os projetistas são naturalmente levados a desenvolver programas que assumem o controle das ações do usuário.

Sequenciamento do diálogo - É o atributo da interface que permite uma livre navegação através do diálogo, ressaltando, dessa forma, a utilidade e a operacionalidade do sistema.

Flexibilidade - É o atributo que faz com que a interface atenda tanto aos usuários principiantes quanto aos usuários experientes. A maior parte das interfaces atuais exige algum conhecimento ou da aplicação ou da máquina ou do software operacional usado. Contudo, é importante considerar que:

- todo usuário experiente um dia foi um principiante;
- nenhum usuário gosta de executar tarefas de iniciante, à medida que se torna um usuário experiente;
- em interfaces bem projetadas, esta passagem de principiante para experiente pode ocorrer rapidamente.

A flexibilidade da interface requer que ela se ajuste ao nível de experiência do usuário. Na realidade, a grande dificuldade no projeto de interfaces adaptativas reside no fato de que os projetistas experientes geralmente perdem a empatia com os usuários principiantes para os quais têm de projetar o seu software.

Portabilidade - É a característica da interface que a torna utilizável para diferentes aplicações em diferentes máquinas e para os mais variados tipos de usuários. Dessa forma, surgiu a idéia de desenvolver as interfaces do usuário independentes das aplicações. Conseqüentemente, surgiu, por sua vez, outra idéia com o objetivo de facilitar esta tarefa: a dos Sistemas de Gerenciamento

de Interfaces do Usuário (SGIU), semelhantes aos Sistemas de Gerenciamento de Banco de Dados (SGBD), [BRANCO 90] .

Reutilizabilidade - É a característica da interface que torna partes dela independentes da aplicação e do equipamento, a fim de permitir que um mesmo código possa servir em vários locais de uma mesma interface ou interfaces diferentes.

Amigabilidade - É a característica da interface que oferece fácil entendimento e simples manipulação simples por parte dos usuários. O usuário não espera necessariamente, que a aplicação resolva o seu problema de forma eficiente, mas ele espera que seu programa comunique claramente o que faz. A interface deve ser baseada não no conhecimento do projetista e, sim, no conhecimento do usuário. O projetista é levado a basear seu projeto na seleção de funções, na escolha dos dados, na seqüência de operações que facilita a construção do programa que ele imagina adequado e se, por alguma razão, pensa no usuário, pensa em si mesmo como sendo esse usuário. Uma interface amigável deve facilitar a vida do usuário e, não, a do projetista.

Avaliabilidade - É o atributo da interface que permite que ela seja revista facilmente, com o objetivo de melhorar a sua adequação à aplicação, ao usuário ou ambiente no qual ela irá atuar. Como todo bom artista, o projetista de "software" também se envolve emocionalmente em seu trabalho; por este motivo, é difícil avaliar a si mesmo. A avaliação é atingida através de dois fatores, que são validação e verificação.

Validação - permite uma avaliação em termos da efetiva correspondência entre o que foi idealizado e projetado e o que se encontra atualmente implementado.

Verificação - é a característica da interface que permite que se façam medições da adequação dos elementos usados e da adequação de sua representação. E nela que se encontra, por sua vez, o subfator consistência, descrito a seguir

Consistência - O tipo de aplicação e o modelo de diálogo usado devem ser perfeitamente adaptados um ao outro, complementando-se e, não, conflitando entre si. Requer ainda que as ações realizadas pelo usuário se harmonizem com o modelo de diálogo que lhe é apresentado. A consistência entre os procedimentos realizados pelo usuário reduz o volume dos conhecimentos que devem ser assimilados por ele. Caso uma mesma tarefa seja realizada em diferentes contextos, é importante que os procedimentos que devem ser cumpridos pelo usuário sejam os mesmos.

Segundo [RHYNE 87], existem vários níveis de consistência de interface, a saber:

- Consistência de papéis - Fundamentada em interações que envolvem objetos, os quais cumprem papéis similares, deve ser semelhante à estrutura da interface, permitindo aos usuários fazer generalizações;

- Consistência de categorias - Os usuários finais deverão ser capazes de realizar manipulações similares sobre objetos do mesmo tipo;

- Consistência de apresentação - Os mesmos objetos não deverão aparecer em diferentes representações que não estão visivelmente relacionados;

- Consistência da interação - Os componentes da interação, tais como desenhos, seleção etc., devem ser realizadas sempre da mesma maneira;

- Consistência das Metáforas Usadas - É importante que a interface explicite claramente quaisquer discrepâncias entre as funções implementadas e as metáforas usadas para auxiliar a compreensão dos usuários.

Após estudos dos conceitos aqui descritos, selecionam-se cinco itens que servem como parâmetros quantificáveis de qualidade, a saber:

- Acertos;
- Ajuda;
- Erros;
- Duração da operação;
- Comandos (utilização de comandos).

No próximo Capítulo, descreve-se a Ferramenta FAI com a presença desses parâmetros, que, combinados entre si, podem ser relacionados com alguns desses conceitos. E no Capítulo 5, são comentados os resultados da coleta de dados de uma interface, como estudo de caso.

CAPÍTULO 4

A FERRAMENTA FERRAMENTA DE AVALIAÇÃO DE INTERFACE

A FERRAMENTA FERRAMENTA DE AVALIAÇÃO DE INTERFACE

A FAI (Ferramenta de Avaliação de Interface) compõe o ambiente AGILE. Portanto, para que se possa apresentar suas características, é necessário apresentar, inicialmente, o ambiente AGILE [PROCÓPIO 92].

4.1 Ambiente AGILE

O AGILE é um sistema de Prototipagem de Interfaces, que foi desenvolvido com o propósito de oferecer recursos aos projetistas de interface para prototipagem e simulação de interfaces.

Conforme [SANTOS 92] e [PROCÓPIO 92], o AGILE representa um tipo de arquitetura de SGIU, baseando-se no modelo de Seheim [LOWGREN 88] e [GREEN 85] (figura 4.1).

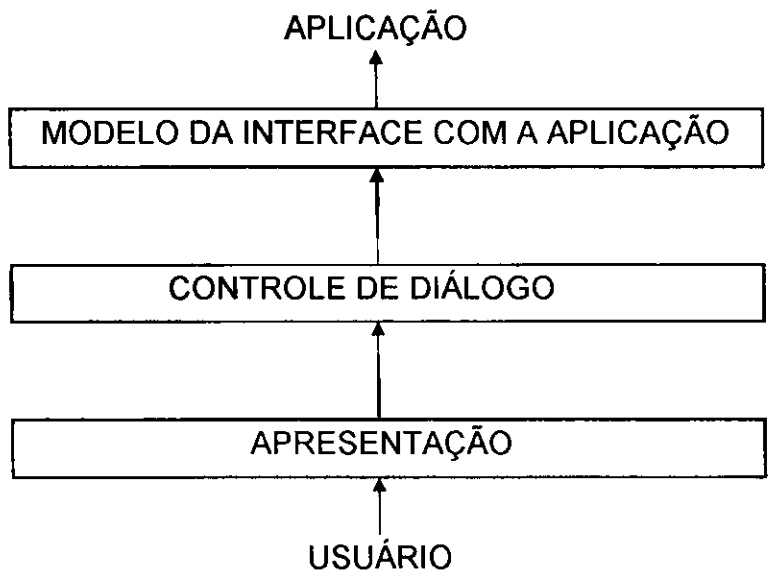


Figura 4.1 - Modelo de Seehein

O AGILE é composto de dois grandes módulos: Gerente da Apresentação e Gerente de Diálogos, correspondendo, respectivamente, ao módulo de Apresentação e de Controle de Diiálogo do modelo de Seehein, e são responsáveis pelas funções de simulação. A gerência da interface baseia-se no modelo da interface aplicação-do-usuário.

Com o desenvolvimento da FAI, algumas funções do AGILE sofreram modificações, especificamente as funções de gerenciamento dos diálogos. Na realidade, as alterações nessas funções foram feitas para permitir que a FAI, coletasse os dados no momento da simulação de um protótipo.

A figura 4.2 apresenta a arquitetura do AGILE com a FAI acoplada. As funções do Gerenciador de Diálogos, Gerenciador de Apresentação, Editor de Interfaces, Gerador de Documentação, foram mantidas na sua forma original, com pequenas modificações no gerenciador de diálogo para inclusões das funções de captura de dados da FAI.

O prototipador de interfaces do ambiente AGILE oferece quatro tipos de diálogos para o projetista: menu, pergunta-e-resposta, comando e formulário. Sua implementação foi em linguagem de programação C.

Após a criação de um protótipo de uma interface, o AGILE armazena todos os dados referentes a esse protótipo em um arquivo do tipo ".PTP" (ver Apêndice **A,** onde é apresentada e comentada a estrutura desse arquivo).

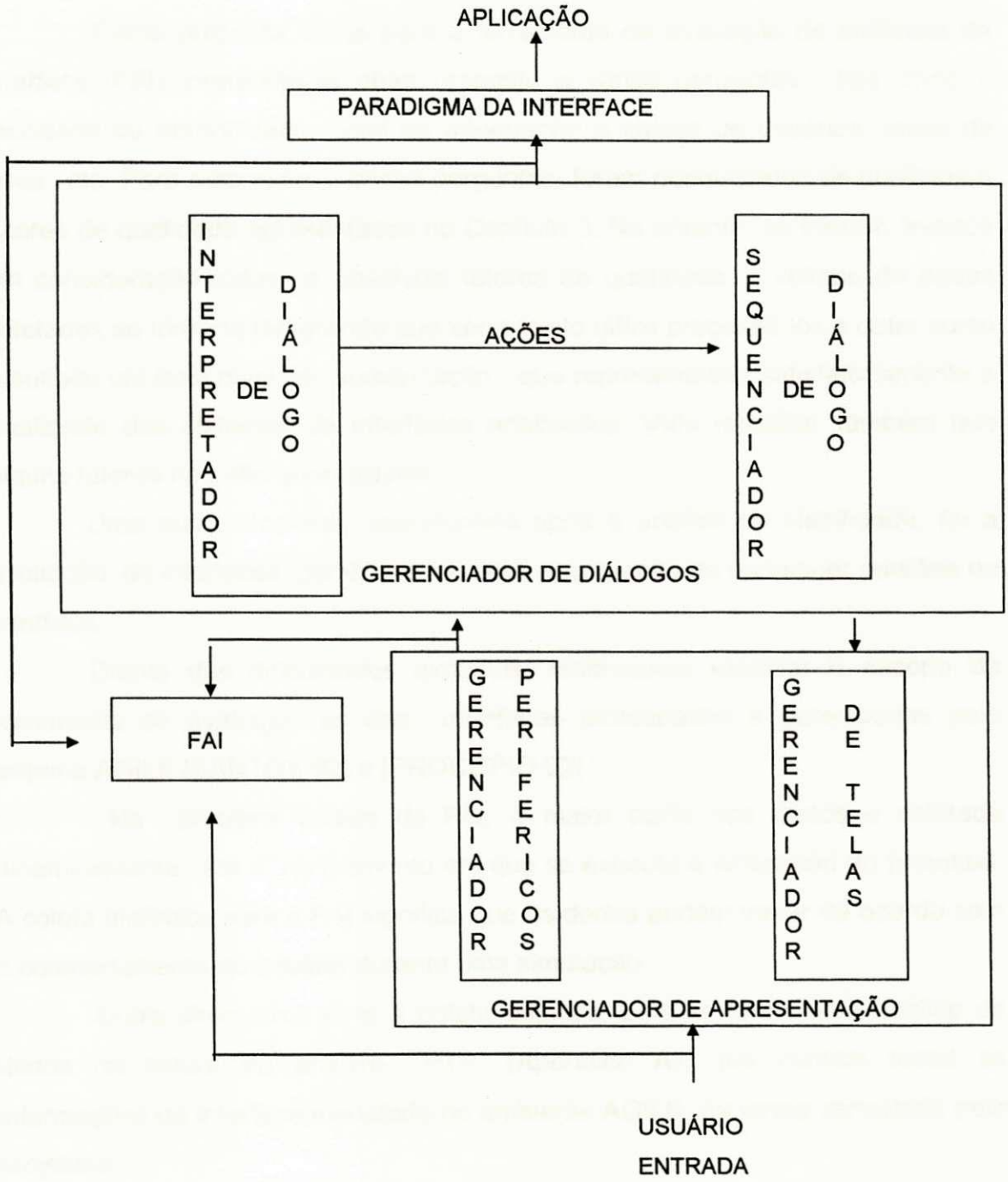


Figura 4.2 - Arquitetura AGILE/FAI

4.2 Escopo da Ferramenta FAI

Como proposta inicial para a ferramenta de avaliação de sistemas de interface (FAI), pretendia-se obter resposta a várias perguntas, tais como a facilidade de aprendizado, nível de adequação à classe de usuários, taxas de erros etc. Para responder a essas perguntas, foram pesquisados os objetivos e fatores de qualidade apresentados no Capítulo 3. No entanto, se fossem levados em consideração todos os possíveis fatores de qualidade, o volume de dados coletados se tornaria tão grande que seria muito difícil processá-los e obter como resultado um bom nível de sumarização, que representasse satisfatoriamente a qualidade dos sistemas de interfaces analisados. Vale ressaltar também que alguns fatores não são quantizáveis.

Uma outra proposta, abandonada após a análise de viabilidade, foi a avaliação de interfaces genéricas, ou seja, a avaliação de quaisquer padrões de interface.

Diante das dificuldades expostas, resolveu-se estreitar o escopo da ferramenta de avaliação ao das interfaces prototipadas e gerenciadas pelo sistema AGILE [SANTOS 92] e [PROCÓPIO 92].

Na primeira versão da FAI, a maior parte dos dados é coletada dinamicamente, isto é, no momento em que se executa a simulação do protótipo. A coleta dinâmica para a FAI significa que os dados podem variar de acordo com o comportamento do usuário durante uma simulação.

Outra alternativa seria a coleta estática, pois se teria uma análise de dados na leitura do arquivo ".PTP" (Apêndice **A**), que contém todas as informações da interface modelada no ambiente AGILE, na forma concebida pelo projetista.

Com este trabalho, tem-se a intenção de validar a viabilidade técnica da FAI como o desenvolvimento de um avaliador de interface para o ambiente AGILE.

A partir da coleta de dados, a FAI oferece as seguintes informações ao projetista:

- Tempo de execução de uma tarefa na sessão de interação;
- o percentual que cada tempo de resposta representa a cada simulação;
- quantidade de ajuda solicitada por diálogo;
- quantidade de erros por diálogo;
- combinação de cores de cada diálogo;
- tempo total de uma sessão;
- total dos erros cometidos durante uma sessão;
- total de acertos durante uma seção.

Através dessas informações, o usuário da ferramenta FAI, o projetista, terá condições de detectar os possíveis gargalos de comunicação no projeto de uma interface após a coleta de dados de algumas sessões.

Os principais aspectos de avaliação da interface que o projetista pode explorar são [ESTEVAM 90] (conceito descrito no capítulo anterior):

- Rapidez (tempo de resposta do usuário);
- indulgência e funcionalidade (do sistema de interface);
- facilidade de aprendizagem (por parte do usuário);
- clareza (por parte do diálogo);
- usabilidade (da interface);
- comunicação;
- retenção dos comandos (memorização).

Rapidez - Este item diz respeito à capacidade do usuário de responder imediatamente a uma ação da interface, dependendo do tipo de diálogo, adaptado à realidade dos fatos. Segundo [GALITZ 88], existem facilidades quando se mede o tempo de resposta do sistema de aplicação em relação ao

usuário, mas, quando a situação é inversa, é muito difícil dimensionar esse tempo, devido às diversidades dos perfis dos usuários. Porém, pode-se ter como base o tempo de resposta de um procedimento para sistema interativo, obedecendo aos seguintes limites:

- Cerca de 2 segundos, quando o usuário necessita de realimentação que oriente cada ação subsequente;
- entre 2 e 4 segundos, mesmo após o encerramento da subtarefa, se ainda for necessária a concentração para a continuidade da tarefa principal;
- entre 4 e 15 segundos, no caso de encerramento da tarefa principal, ou processamento longo, do tipo atualização de arquivos;
- maior que 15 segundos, liberação do usuário para realizar outras tarefas.

Os sistemas interativos, por si, exigem respostas rápidas. O tempo de resposta do usuário em relação ao sistema depende do seu nível de experiência. Quanto mais freqüente e experiente, mais o tempo de resposta gira em torno de 2 segundos [GALITZ 88].

Indulgência e funcionalidade - Através do tempo de resposta do usuário e de erros cometidos durante um diálogo, verifica-se a eficiência do tratamento de erros cometidos durante o processo de operação da interface. O tratamento de erros durante um diálogo de uma interface deve assegurar a minimização da quantidade de erros cometidos pelo usuário, bem como o fornecimento de maneiras rápidas e fáceis para que o usuário se recupere das situações causadas pelos erros [SHNEIDERMAN 83].

Facilidade de aprendizado - Quanto mais próximo do índice zero de erros, de solicitação de ajuda e de tempo de resposta, tanto maior será o índice de amigabilidade do sistema. Este aspecto é o primeiro detectado pelo usuário, pois se trata de assimilação dos conhecimentos necessários ao manuseio da interface [SHNEIDERMAN 87]. Estas informações que a FAI fornece podem ser enquadradas dentro dos aspectos de clareza, comunicação e usabilidade, pois no momento em que a interface se torna amigável, existe automaticamente a facilidade de comunicação, clareza e manuseio, que são fatores de decisão na aceitação e comercialização de um produto [GALITZ 88].

4.3 Descrição Funcional da FAI

A FAI é um sistema que executa a avaliação quantitativa de uma parte ou de toda interface de um sistema gerado pelo prototipador de interface AGILE [BRANCO 89]. O sistema FAI, implementado na linguagem de programação "C + +", para rodar em ambiente MSDOS, terá dois grandes blocos de funções: o primeiro, coletar dados; o segundo, sumarizar os dados coletados dentro dos parâmetros citados na seção anterior.

A sumarização dos dados é o gerenciamento das informações coletadas para fornecer relatório conforme "lay-out" apresentado na seção 4.5 deste Capítulo.

Este sistema não executa inferências nos dados coletados, apenas oferece informações gerenciais para que o analista tome as suas próprias decisões.

Para que ocorram a coleta de dados e a sumarização, o projetista modela a interface com o prototipador AGILE (ver Capítulo 5), seleciona dentre os quatro tipos de usuários que a FAI considera [MARTIN 73] e determina o número de sessões de execução da simulação com os usuários. O projetista não necessariamente tem de utilizar os quatro tipos de usuários; ele pode definir e executar testes com apenas um tipo de usuário. Seria o caso de interfaces de sistemas específicos que atendem apenas uma categoria de usuários.

4.4 Arquitetura da Ferramenta FAI

Na figura 4.3, apresenta-se a arquitetura da FAI, onde se verifica a presença dos dois grandes módulos, o coletor de dados e o sumarizador.

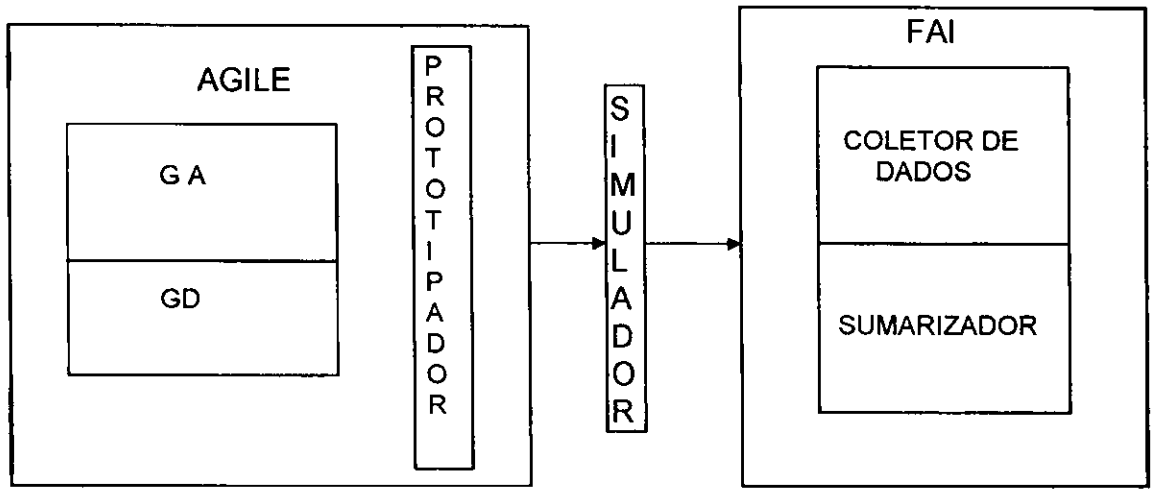


Figura 4.3 - Arquitetura da FAI

No ambiente do simulador, processa-se a gerência da interação do usuário com uma aplicação qualquer. Esse gerenciamento é feito em nível de simulação vazia, desde que a prototipagem seja vazia, isto é, gerencia a interação entre o usuário e a interface da aplicação, sem a presença das funções da aplicação, permitindo, dessa forma, que o usuário final possa interagir com o protótipo da interface, de maneira a validá-lo, ou não.

A coleta de dados é feita de duas formas conforme descritas a seguir. A primeira dá-se na leitura do arquivo que contém todas as informações da interface prototipada, ou seja, executa a leitura do arquivo com terminação ".PTP" (Apêndice **A**), previamente informado pelo projetista. Essa coleta de dados é

denominada **estática**, e as informações que a FAI coleta nesta versão são a quantidade de diálogos existentes, os tipos de diálogos, as teclas utilizadas nos diálogos e os comandos existentes. As teclas utilizadas nos diálogos, lidas do arquivo ".PTP", servem como base para o "software" de captura da entrada de dados do teclado [KLEBER, TURNELL 92].

As informações obtidas na **coleta estática**, nesta versão, não provocam impacto visível na avaliação, mas servem como referencial para a coleta **dinâmica**.

A segunda forma de coleta, denominada **coleta dinâmica**, acontece durante a simulação vazia do protótipo em tempo de execução. Armazenando-se os dados coletados em um arquivo com o mesmo nome do arquivo do protótipo, acrescido do caractere de controle da FAI, ou seja, quantidade de entradas no protótipo e a terminação do nome do arquivo ".FAI", (por exemplo, LAURO1.FAI), significa que a FAI coletou dados do protótipo modelado no AGILE a partir do arquivo LAURO.PTP, com uma entrada na interface.

O sumarizador tem como função classificar as informações coletadas para gerar um relatório com todos os dados coletados, permitindo a comparação com os fatores de avaliação apresentados na seção 4.2 (ver seção 4.6 deste Capítulo, onde é apresentado o "lay-out" da tela de saída de dados).

A saída de informações para arquivos é armazenada no modo texto com terminação no nome do arquivo ".TXT", ou ".PRN" (formato de impressão), dependendo da opção do usuário da ferramenta .

A decisão pela saída de informação em um desses formatos permite que sejam utilizadas planilhas eletrônicas, que facilitam a análise dos dados sumarizados. A partir de pesquisa nos manuais de planilhas, constata-se que a maioria delas aceita arquivos do tipo texto (TXT) ou do tipo formato de impressão (PRN).

4.5 Estrutura de Dados da FAI

Para a estrutura de dados da FAI, foi definido um vetor que, em cada endereço, contém um único objeto: a classe janela (diálogo), conforme figura 4.4. No Apêndice **B**, encontra-se o trecho do código FAI que cria o objeto janela com todos os seus atributos e o vetor de objetos.

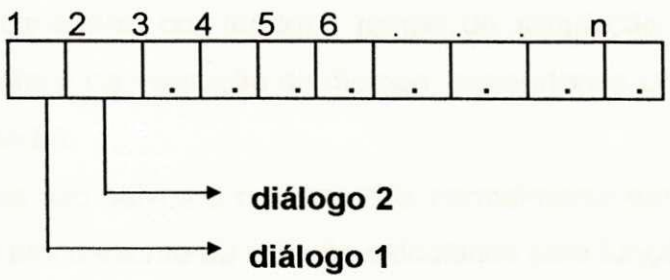


Figura 4.4 - Vetor de objetos

Dessa forma, pode-se lidar com a simulação de um protótipo com **n** diálogos, ficando este número limitado a capacidade de memória do computador. Esse controle do tamanho do vetor é dado através da variável numérica definida como **total,** na função " *main* ( ) " , através da qual se determina o tamanho do vetor ou a quantidade de diálogos existentes no protótipo. Esta variável, por sua vez, é repassada, junto com o vetor de objeto, para diversas funções, como " *coletor( )* " no AGILE , " *avaliar ( )* ", e " *salvar ( )* ". O vetor de objetos é modificado segundo o valor retornado .

Os atributos da classe diálogos, como, por exemplo, temporizador do serviço "*cronometro* ( ) ", mensagens do serviço " *mensagem* ( ) " , " *menu* ( ) " e outros atributos, contribuem para que cada objeto possua seu próprio controle independente, podendo salvar e recuperar os dados através da função " *salvar* ( ) " e " *ler* ( ) " (Apêndice **B**).

A posição do diálogo e suas cores são definidas no serviço "*posicionar* ()", registrados nos atributos " **x, y,** *cor_fundo, cor_borda, cor_caractere*". O tempo que o usuário passa em cada diálogo, executando uma tarefa dentro de uma sessão, é registrado no atributo "tempo em cada objeto". O tempo total da sessão é obtido, somando-se os valores da variável "tempo de cada objeto", e armazenado na variável *"tmp"* da função " *avaliar* ( ) " .

Na classe " janela " (diálogo), as variáveis de instância são definidas através da posição do diálogo (x, y), função do diálogo (função), cores do diálogo (cor_borda, cor_caractere, cor_fundo), tempo de simulação (tempo) e outras variáveis que auxiliam na execução do diálogo, dependendo unicamente de suas funções(Apêndice **B**).

Os objetos são salvos e recuperados normalmente em arquivo binário e os bytes de cada bloco escrito ou lido são calculados pela função "sizeof ( )".

O coletor de dados utiliza as estruturas de dados do tipo pilha de diálogos e estrutura de ações do AGILE [PROCÓPIO 92] (ver Apêndice **C**). A FAI lê o topo da pilha para saber qual o diálogo que está ativo naquele instante, e passa a operar com a lista encadeada das ações do diálogo (Apêndice **C**) para atualizar os atributos do objeto janela (FAI). Tanto na coleta dinâmica, quanto na coleta estática que ocorre no momento da leitura do arquivo ".PTP" no AGILE (Apêndice **A**) são atualizadas as variáveis "total", "ponteiro de sequenciamento", "tipos de diálogos", "teclas aplicáveis" e os "comandos" existentes.

4.6 Interface com o Usuário

A interface da FAI foi prototipada, utilizando o Editor de Interfaces do ambiente AGILE, dentro das limitações dos recursos disponíveis, aproveitando análise ergonômica da FAI com base em diretrizes de programação visual [QUEIROZ e TURNELL 94]. A FAI é uma interface do tipo texto com características do ambiente.

A entrada da ferramenta FAI encontra-se na opção do menu principal do AGILE. Portanto, para utilizar a FAI, torna-se necessário acionar o AGILE, simular um protótipo, para proceder à coleta dos dados. A seguir, é necessario selecionar a opção "AVALIAR (FAI)" do diálogo (figura 4.5), para se visualizar o resultado da coleta.

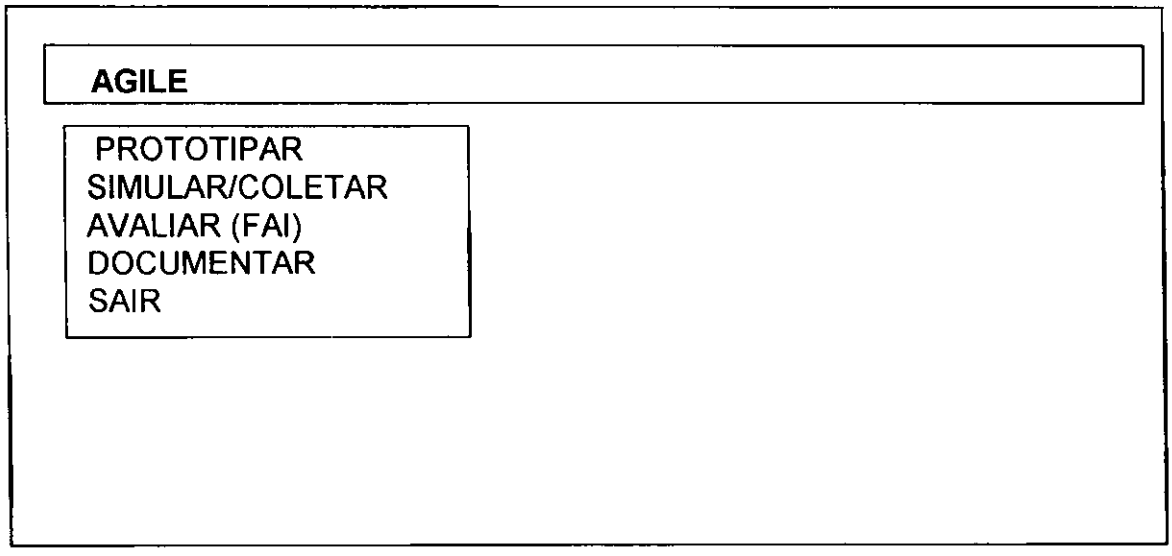


Figura 4.5 - Interface AGILE

Após selecionar esta opção (AVALIAR/FAI), estar-se-á navegando na interface do sistema FAI (Ambiente FAI), conforme figura 4.6.

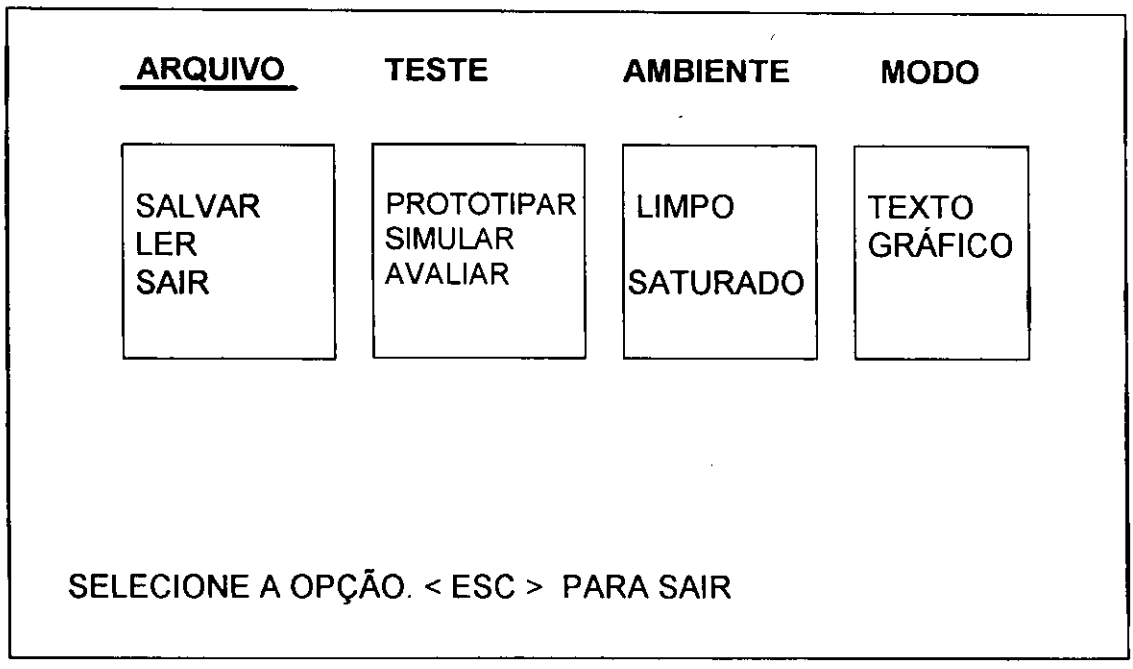


Figura 4.6 - Interface da FAI

Seleção do protótipo para análise:

Na figura 4.6, na opção ***arquivo,*** têm-se três funções: **salvar**, **ler** e **sair**. A função **salvar**, coleta os dados relativos a um protótipo de interface na memória e grava na unidade corrente (por exemplo, drive **c**, **a**, ou **b**), o nome do arquivo que irá conter essas informações sobre o protótipo em memória. Selecionada a opção, o sistema solicita ao usuário, através de uma mensagem no rodapé, o nome do arquivo. O sistema só sai desse estado no momento em que o nome do arquivo é informado.

A função **ler** carrega para a memória o arquivo do tipo ".FAI", selecionado pelo projetista, que contém o resultado da coleta de dados .

A função **sair** abandona o ambiente FAI e retorna ao AGILE. A tecla ESC, quando acionada nesta interface, também ocasiona o retorno ao ambiente AGILE.

Nesta opção **teste,** estão disponíveis três funções: prototipar, simular e avaliar.

Embora já estejam disponíveis recursos no AGILE para prototipação e simulação, foi iniciado um trabalho de desenvolvimento de um módulo alternativo para prototipar/simular. Este módulo, que atualmente só permite prototipar/simular o estilo de interação MENU, teve como objetivo testar uma alternativa de coleta de dados que, por razões de prazo, foi temporariamente abandonada. Na realidade, foi utilizado, neste trabalho, o prototipador/simulador do AGILE.

A função **Avaliar,** da opção **teste,** permite a entrada de dados do protótipo e da sessão. O sistema apresenta um menu com os quatro tipos de usuário que a ferramenta FAI comporta, com uma mensagem, no rodapé, solicitando a seleção do tipo de usuário (ver figura 4.7).

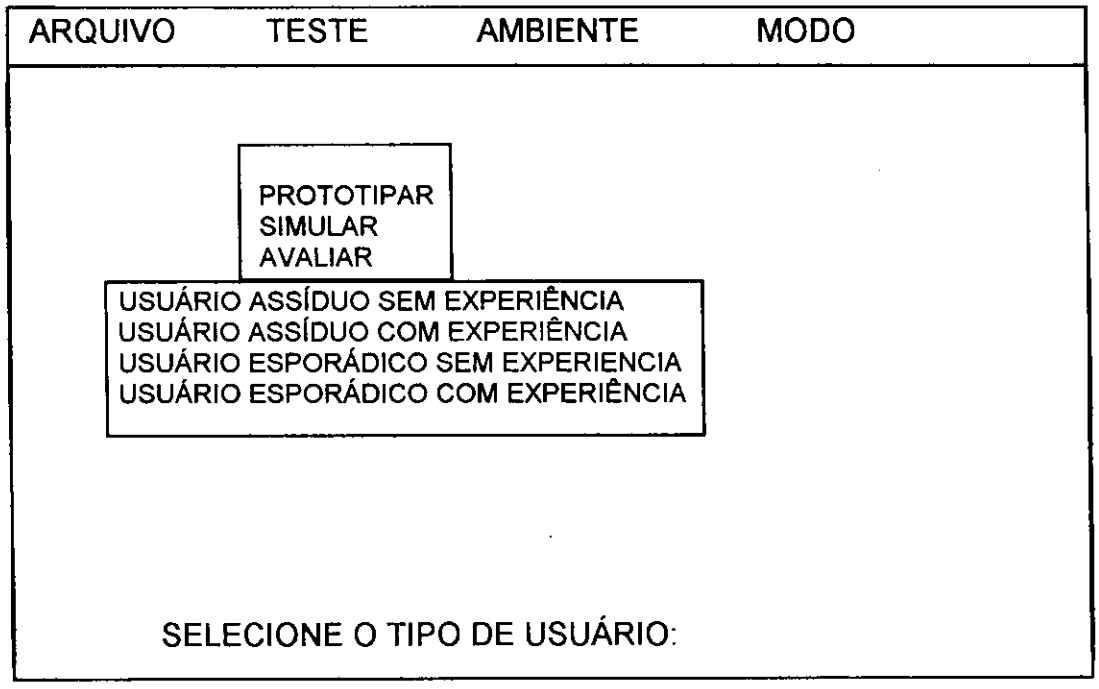


Figura 4.7 - Interface FAI (Função avaliar)

Após a seleção do tipo de usuário (figura 4.7), o sistema solicita, através de mensagens no rodapé, o número da versão, nome do projetista e o número da sessão.

O sumarizador fornece ao usuário-projetista informações conforme "lay-out", figura 4.7.a, figura 4.7.b e figura 4.7.c.

FERRAMENTA DE AVALIAÇÃO DE INTERFACE
Dados do Protótipo

Nome : SISPES Projetista : NAKAYAMA
Versão : 1

Dados da Sessão

Data	: 14/05/1995
Tipo de usuário	: Assíduo sem experiência
Id. da Sessão	: 1/2
Duração da Sessão	: 62 seg.
Total de diálogos	: 7
Erros	: 8
Acertos	: 31
Solicitação de Ajuda	: 8
Comandos Utilizados	: 1
Comandos Disponíveis	: 4

Figura 4.7.a - Sumarização dos dados (Página 1)

Nesta figura, apresenta-se a primeira tela de saída de informação da FAI. Na primeira parte deste "lay-out", aparecem o nome do sistema modelado no AGILE, o número da versão e o nome do projetista responsável. Na segunda parte, têm-se os dados que dizem respeito à sessão: a data em que ocorreu a sessão, tipo de usuário (descrito no Capítulo 5, sessão 5.2), identificação da sessão, duração da sessão, total de diálogos (quantidade de diálogos que o sistema possui), erros, acertos, solicitação de ajuda, comandos utilizados e comandos existentes.

Para a FAI, é considerado **acerto** quando uma tarefa é concluída com êxito em um diálogo, seguindo a sintaxe e lógica do sequenciamento .

Erros - São desistências da seleção efetuada na interface ou procedimentos executados pelo usuário não previstos na interação do diálogo ativo. Outro tipo de erro que está sendo registrado é a não conclusão da tarefa. Por exemplo, se o usuário notar que o procedimento no primeiro diálogo está errado e ele retornar do quinto para o primeiro, os quatro diálogos executados serão considerados errados, isto é, cada diálogo terá um erro computado.

Ajuda - É o somatório de solicitação de ajuda em cada diálogo.

Duração(s) - É o tempo (em segundos) que o usuário demora para executar uma tarefa durante uma sessão.

%Duração (percentagem de duração de sessão) - É o percentual que o tempo de resposta do usuário representa durante a execução de uma tarefa a cada diálogo.

$$\%\ \text{duração de sessão} = \frac{\text{duração de um diálogo(s)}}{\text{duração da sessão(s)}} \times 100$$

Na figura 4.7.b, pode-se visualizar a coleta de dados por diálogo.

FERRAMENTA DE AVALIAÇÃO DE INTERFACE

COLETA DE DADOS POR DIÁLOGO

Diálogo	:	1	2	3	4	5	6	7
Tipo (dial.)	:	MEN	P_R	MEN	MEN	MEN	MEN	P_R
Acertos	:	007	005	006	003	006	002	003
Erros	:	001	002	003	007	002	001	003
Ped. Ajuda	:	004	003	004	001	002	001	002
Duração (s)	:	012	013	021	012	013	002	034
%Duração	:	065	045	032	012	012	031	003

Figura 4.7.b - Sumarização dos dados (página 2)

Na figura 4.7.b, têm-se o posicionamento numérico do diálogo correspondente, o tipo de diálogo, quantidade de acertos, erros cometidos, pedido de ajuda, duração (medida em segundos) e o percentual de duração da sessão em cada diálogo .

A figura 4.7.c mostra as cores que são utilizadas em cada diálogo.

FERRAMENTA DE AVALIAÇÃO DE INTERFACE

USO DE COR

Diálogo	Cores
1	AZUL BRANCO PRETO
2	AZUL BRANCO PRETO
3	AZUL BRANCO PRETO
4	AZUL BRANCO PRETO

Saída do relatório em (I) impressora ou (A) arquivo

Figura 4.7.c - Sumarização dos dados (página 3)

A partir das informações sobre cores, o usuário projetista da FAI pode avaliar o impacto dos recursos cromáticos utilizados. Para os aspectos a serem avaliados destacam-se as seguintes diretrizes de projeto [SILVA 94]:

- **Uso moderado da cor**: Evitar diálogos excessivamente coloridos, que possam resultar em telas confusas e desconfortáveis.

- **Limitação do número de cores**: Limitar em 4 ou 5 o número de cores a ser apresentado simultaneamente em cada tela. Evitar o uso de mais de sete cores ao longo da sequência completa do diálogo.

- **Potencialidade da cor como técnica de codificação**: Experimentar relações entre cores que possibilitem usar o recurso da cor como um mecanismo de codificação eficiente e agradável de usar. Procurar reconhecer as situações em que tal uso compromete o desempenho das tarefas.

- **Discriminação de cores**: Não empregar mais de sete cores em situações onde a cor for usada, não como um atributo de separação física ou lógica, mas como um recurso para discriminação, ênfase ou associação de itens. Selecionar cores contrastantes, aos pares ou em grupos (e.g., vermelho e branco; azul e amarelo; vermelho, verde e azul), nos casos onde o objetivo seja o de discriminar ou enfatizar itens.

Usar cores similiares, associadas aos pares ou em grupos maiores (e.g., laranja e amarelo; azul e violeta; vermelho, laranja e amarelo), ao desejar sugerir similaridade de itens. Usar cores complementares (opostas no círculo de matizes) para produzir efeitos cromáticos contrastantes, empregando, no máximo, 10 matizes distintos.

- **Combinação de cores**: Estudar exaustivamente todas as combinações de pares de cores, tendo em mente que o contraste é um elemento-chave para a compreensão de mensagens codificadas por cores.

- **Codificação de mudanças de estado através de cores**: Empregar variações cromáticas para indicar mudanças de estado de variáveis contidas no domínio da interface.

- **Associação com denotações familiares**: Associar, sempre que possível, cores a eventos, tomando, como base, a experiência anterior do usuário. Usar denotações comuns onde for possível (e.g., usar a cor vermelha em circunstâncias que denotem perigo, sugiram atenção extrema ou indiquem o encerramento de uma tarefa sob condições de erro).

Os recursos cromáticos só podem ser avaliados se o sistema estiver utilizando monitor colorido. No estudo de caso apresentado no Capítulo 5, a avaliação de cores não foi possível porque o sistema escolhido utiliza monitor monocromático.

Arquivos de saída da FAI

Na última tela de sumarização da FAI (rodapé da figura 4.7.c), o sistema solicita o tipo de saída desejada, monitor ou impressora. Se a saída for selecionada em arquivo, o sistema solicita, também no rodapé, o nome do arquivo, acrescido de ".TXT", para arquivo tipo texto-padrão "DOS" e ".PRN" formato de impressão, de acordo com o tipo de planilha que o prejetista pretenda utilizar (ver figura 4.7.c).

Está-se usando o nome do arquivo para identificar as informações relacionados a cada usuário, sessão e versão, ou seja, os cinco primeiros dígitos são utilizados para identificar o projeto, o sexto digito é o tipo do usuário, o

sétimo dígito é a sessão e o oitavo digito, a versão do projeto de interface. A terminologia dos gráficos adotada para descrever as apresentações de gráficos (Capítulo 5) e identificação de arquivos é a seguinte:

- USR1: usuários assíduos sem experiência;
- USR2: usuários assíduos com experiência;
- USR3: usuários esporádicos sem experiência;
- USR4: usuários esporádicos com experiência;
- V1 : primeira versão do sistema;
- V2 : segunda versão do sistema;
- S1 : primeira sessão;
- S2 : segunda sessão.

Identificador dos arquivos de coleta:

Para melhor compreensão, dar-se-á aqui um exemplo de como identificar as informações contidas nos arquivos. A partir de um nome qualquer, do tipo "SISPE112.TXT", a FAI interpreta que esse arquivo contém informações do projeto SISPES do usuário 1 (assíduo sem experiência), primeira sessão, segunda versão, no formato texto.

Para análise dessas informações, decidiu-se utilizar planilha eletrônica existente no mercado, e o formato de arquivos que a maioria das planilhas aceita é do tipo texto, conforme pesquisas efetuadas nas cartelas de operações do "EXCEL", "LOTUS 123" e "QUATRO PRO". A planilha adotada foi o "QUATRO-PRO" para ambiente "Windows".

Através da planilha "QUATRO PRO", serão obtidos gráficos que facilitarão a análise dos dados coletados através da FAI, descritos no próximo capítulo. A terminologia que será adotada nos gráficos é a mesma da saída de informações da FAI, descrita acima.

Na opção ***ambiente,*** existem duas opções, relativas à apresentação do monitor de vídeo, que são:

- Limpo : O monitor de vídeo apresenta-se sem "hachuras", com o pano de fundo totalmente limpo, com a cor cinza predominando.

- Saturado : O monitor de vídeo apresenta-se com "hachuras".

Na opção ***modo***, a Ferramenta FAI oferece ao usuários duas maneiras de operação do monitor, no modo texto e no modo gráfico. Esta opção está presente porque o AGILE funciona todo em modo gráfico e a implementação da FAI foi em modo texto.

CAPÍTULO 5

ESTUDO DE CASO

ESTUDO DE CASO

Uma vez concluído o desenvolvimento da primeira versão da ferramenta, foi selecionada uma aplicação já desenvolvida e em utilização.

Uma aplicação, para ser avaliada na FAI, deve, inicialmente, ser modelada no ambiente AGILE, uma vez que os recursos de prototipação ali existentes permitem a criação apenas de interfaces do tipo texto, para os estilos de interação: Menu, Pergunta e Resposta, Formulário e Comando. A FAI limita-se a avaliar interfaces neste escopo.

Portanto, necessita-se modelar um sistema com essas características e que esteja disponível, não apenas para a modelagem, mas também para coleta de dados a partir de sessões de avaliação com seus usuários.

Foi, então, selecionado um sistema de gerência e informação de uma empresa de pequeno porte, distribuidora de produtos farmacêuticos.

O sistema SISPES - Sistema de Pessoal apontava vários problemas com os aplicativos Cálculo da Folha de Pagamento e dos Tributos.

5.1 Descrição do SISPES

O Sistema de Gerenciamento de pessoal, na realidade, é um sistema de controle de recursos humanos e financeiro da empresa. O trecho escolhido foi a folha de pagamento, com os seus devidos tributos, por ter sido apontado pelos usuários como uma fonte de gargalos.

A parte financeira do SISPES tem uma interação com o setor de faturamento, porque os vendedores dos produtos têm seus vencimentos calculados com base nas vendas efetuadas. Portanto, a parte financeira do SISPES acumula as atividades financeiras e de tributação do setor de faturamento. Outra ligação do SISPES é com o setor de contabilidade, porque o

sistema de tributação do SISPES alimenta diretamente contas a pagar do sistema de contabilidade (ver figura 5.1).

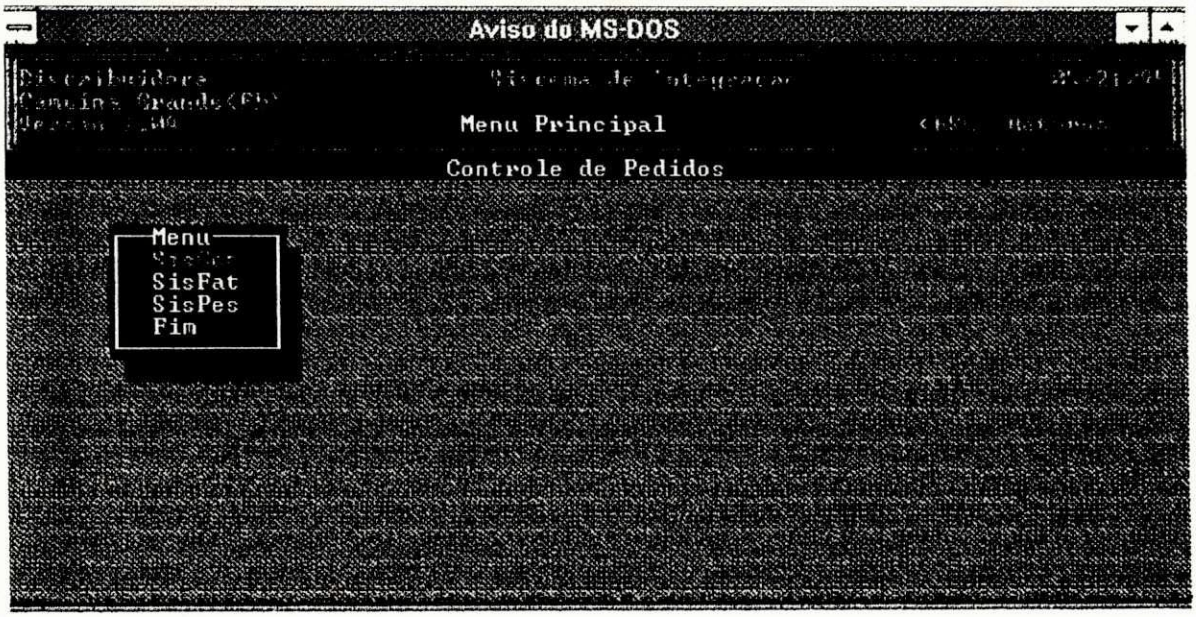


Figura 5.1 - Menu principal do SISPES

A figura 5.2 apresenta, para a interface original das tarefas escolhidas no SISPES, o esquema da seqüência de ações do usuário.

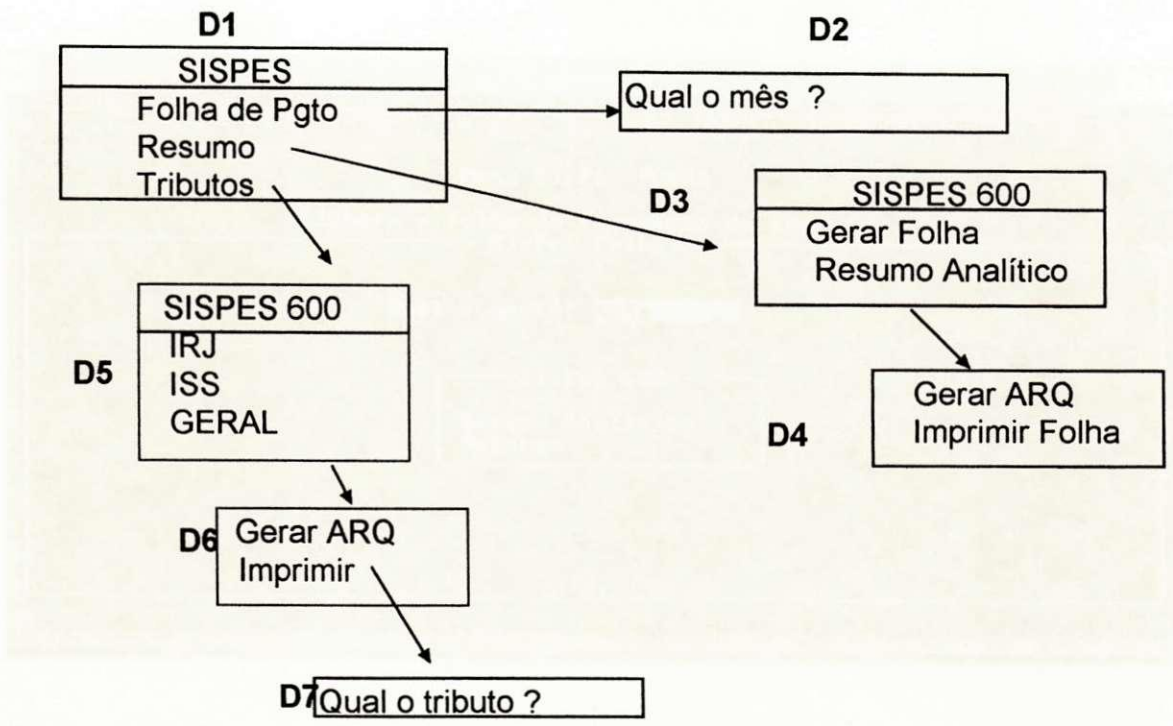


Figura 5.2 - Interface do SISPES Versão V1

Cálculo da folha

No menu principal SISPES (D1), o item folha de pagamento permite o cálculo de dois tipos de pagamento: quinzenal e mensal, para um mês selecionado em D2. O sistema efetua, então, os cálculos da folha a partir do mês informado. Se o mês informado não for o atual, o sistema trava, exigindo o reinício da operação, quando será possível entrar com o arquivo-mestre do mês desejado. Em seguida, o usuário deve selecionar, em D3, a geração da folha ou o resumo analítico. No caso da geração da folha, o sistema só aceitará a opção geração de arquivo, que é o primeiro item do próximo menu, D4. Se, por outro lado, for selecionado o item resumo analítico, o sistema aceita a geração de arquivo ou a impressão do relatório.

A figura 5.3 ilustra os diálogos D2 e D3 da figura 5.2.

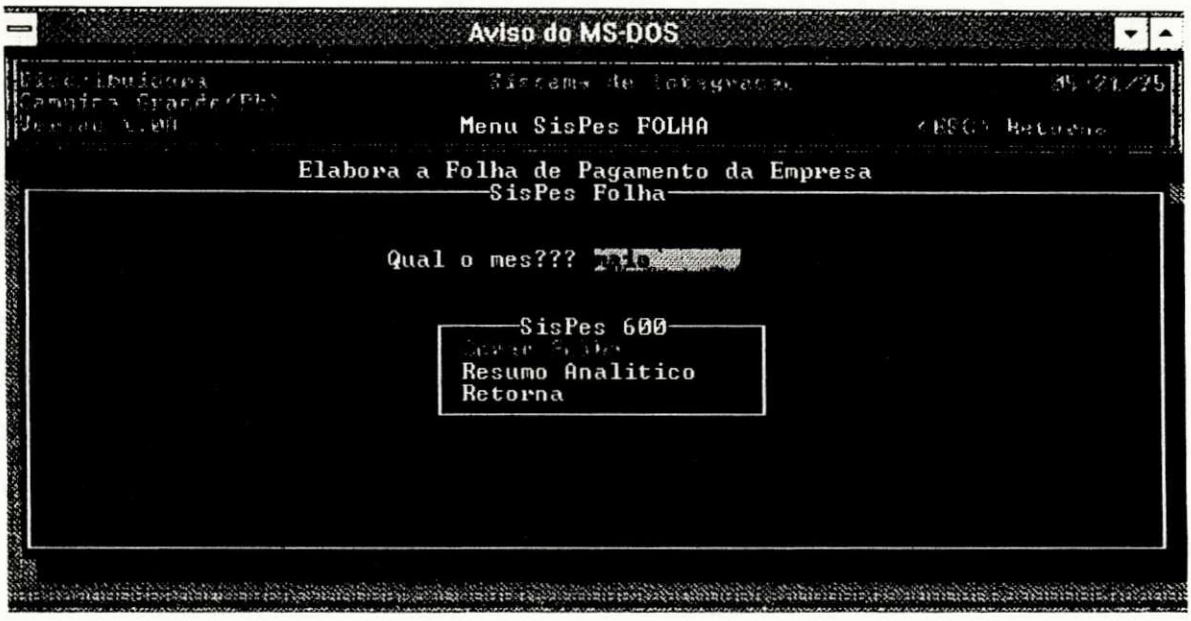


Figura 5.3 - Menu SISPES- FOLHA

O usuário deve ter conhecimento de que, dentro do mesmo mês, após a primeira quinzena, o sistema aceita o item resumo do menu principal, D1. Este item só permite imprimir o resumo analítico da quinzena ou do mês em vigor. Se for necessário o resumo do mês anterior, o operador tem de instalar os arquivos-mestre do mês desejado, uma operação muito complicada no atual sistema.

Cálculo de tributos.

A parte de tributos, que é o terceiro item do menu principal, oferece a opção do imposto de renda, tanto jurídico quanto físico, que são as primeiras opções do diálogo seguinte, D5; a outra opção, GERAL, oferece todos os tributos existentes, ou seja, ISS, FGTS, IRJ, CS etc.

Os três itens de tributos oferecem as opções de geração de arquivo ou impressão do relatório. Na impressão dos tributos, o sistema permite relatório separado de alguns tributos do tipo IRJ, ISS. Quando é solicitado, em D5, o item GERAL dos tributos e, em D6, a seleção de impressão, o próximo diálogo, D7, solicita o nome do tributo. Em D7, se o usuário digitar a tecla "enter", o tributo default é o GERAL. Mas, se desejar o tributo ISS ou IRJ, tem que informar em D7 qual o tributo.

Antes mesmo da avaliação quantitativa, pode-se observar, da descrição das tarefas, que há várias informações de restrição no uso da interface, que devem ser do conhecimento do usuário e para as quais o sistema não oferece ajuda, seja na forma de mensagens de erro, seja na forma de orientação no uso da interface.

5.1.1 Modelagem das tarefas no ambiente AGILE

Utilizando-se os recursos do prototipador do AGILE, foram modeladas as duas funções selecionadas para avaliação.

Para tornar mais objetiva a coleta de dados, foi solicitado aos usuários que, dentro das opções disponíveis nas funções cálculo da folha e cálculo dos impostos, eles realizassem duas tarefas específicas:

(a) Impressão da folha;

(b) cálculo e impressão do ISS.

5.2 Organização do Experimento

Com as duas tarefas delineadas, foram identificados quatro usuários do sistema:

a) Usuário assíduo, sem experiência (USR1) - É o usuário que utiliza o cálculo manual da folha de pagamento, mas não tem experiência nenhuma com o computador, isto é, nunca calculou utilizando o computador.

b) Usuário assíduo, com experiência (USR2) - É o usuário que conhece o cálculo da folha de pagamento e tem experiência com o sistema informatizado. É a pessoa mais qualificada - o expert.

c) Usuário esporádico, sem experiência (USR3) - É o usuário que não conhece o sistema da folha de pagamento e não tem conhecimento nenhum do sistema informatizado ou sobre computadores. É o leigo dos usuários.

d) Usuário esporádico, com experiência (USR4) - É o usuário que utiliza esporadicamente o cálculo manual da folha de pagamento e tem conhecimento do sistema informatizado.

O critério adotado para classificar um usuário como detentor de conhecimentos do sistema de computação é que ele tenha algum treinamento de iniciação à ciência da computação; portanto, com condições de utilizar os comandos básicos do sistema operacional, no caso, o MSDOS.

5.3 Coleta de Dados

O experimento foi planejado de modo que a coleta de dados fosse feita em duas etapas, cada uma delas correspondendo a duas sessões, com intervalos de um dia, para cada tipo de usuário.

O intervalo de um dia de uma sesão para outra é o tempo mínimo sugerido em [GALITZ 88] para verificar se o usuário consegue reter as informações sobre os comandos utilizados na interação com a interface.

Na primeira etapa, coletaram-se dados sobre a versão original do SISPES, modelado no AGILE. A segunda etapa correspondeu à coleta dos dados da versão 2. Esta segunda versão consistiu em uma proposta de solução para os gargalos detectados a partir dos dados, coletados e analisados, da primeira versão.

As sessões não foram limitadas no tempo. A primeira sessão foi precedida de uma explicação das funções da interface, exceto para o usuário assíduo/experiente - USR1.

Durante as sessões, esteve presente o responsável pela avaliação, disponível para responder às dúvidas dos usuários.

Após a análise dos dados coletados da primeira versão, foi elaborada uma versão alternativa, que foi avaliada, utilizando-se a mesma metodologia da primeira etapa. Esta versão e os resultados de sua avaliação são tratados na sessão 5.4.2.

5.4 Resultado da Coleta de Dados da Versão V1

Os dados coletados na FAI são apresentados em três telas de saída, discutidas no Capítulo 4, sessão 4.6.

A primeira tela (figura 4.7.a), apresenta os dados do protótipo sob avaliação, dados sobre a sessão de avaliação e o resultado geral da coleta.

O detalhamento dos dados que aparecem nas telas encontra-se na sessão 4.6, Capítulo 4. No presente Capítulo, deter-se-á na análise dos dados coletados.

Uma vez que os recursos disponíveis no ambiente AGILE se limitam ao modo texto, será utilizada planilha eletrônica para gerar gráficos, pois se acredita que esta representação seja mais adequada para análise, conforme mencionado no Capítulo 4, sessão 4.6.

Os gráficos utilizados para a análise dos dados foram do tipo "pizza", para representar a duração da sessão, e do tipo barra, para representar os outros parâmetros, como se verá a seguir.

5.4.1 Análise dos gráficos

A análise inicia-se com a observação dos índices de Erros, Acertos e Pedidos de Ajuda para cada um dos passos de execução da tarefa, aqui denominados Diálogos, D1 a D7 (ver figura 5.4). A razão desta nomenclatura é a herança de uma terminologia do ambiente AGILE [PROCÓPIO 92].

Na figura 5.4, observa-se a distribuição dos Erros, Acertos e Pedidos de Ajuda para o usuário USR1 - inexperiente, na primeira sessão S1, da análise da versão V1. Estes índices aparecem agrupados com o percentual de duração da sessão (ver Capítulo 4, seção 4.6).

A partir da observação da figura 5.4, verifica-se que o índice do percentual de duração da sessão um (S1), versão um (V1), do usuário sem experiência (USR1) está bastante acentuado nos diálogos D2 e D7, visto que, nesses passos, há maior solicitação de ajuda em relação aos outros diálogos.

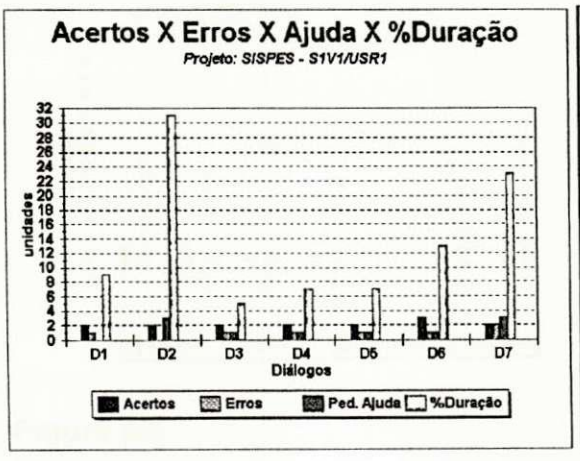


Figura 5.4

Figura 5.5

Para observar melhor a distribuição do tempo de execução dos passos da tarefa, foi construída a figura 5.5, "duração da sessão por diálogo". Nesta figura, observa-se que o maior tempo de resposta registrado foi no diálogo D2 e

diálogo D7. Portanto , detectou-se um gargalo nos diálogos D2 e D7 para este tipo de usuário.

Tendo analisado os dados relativos ao primeiro tipo de usuário, passa-se a apresentar os resultados (na forma gráfica) relativos aos demais tipos de usuário.

Usuário assíduo, com experiência:

O usuário assíduo, com experiência (USR2), que é o mais qualificado tecnicamente para executar estas tarefas.

Na figura 5.6, na primeira sessão, nota-se que não existe presença de erros nem solicitação de ajuda e que o índice de acertos corresponde às expectativas.

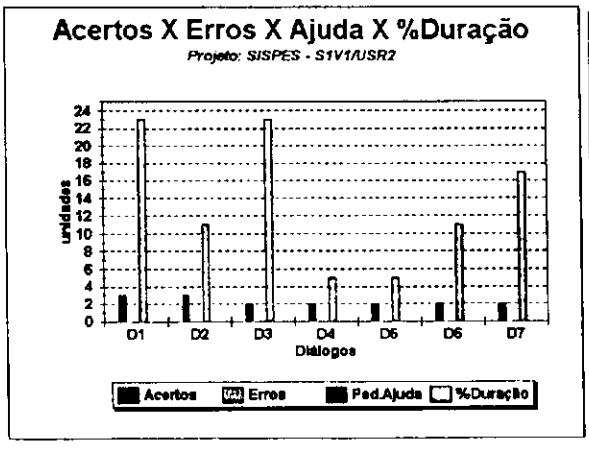


Figura 5.6

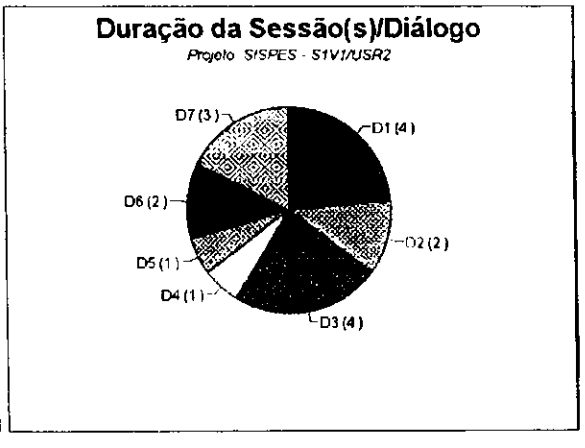


Figura 5.7

Na figura 5.7, retrata-se a duração da sessão por diálogo, podendo-se notar um excelente tempo de resposta do usuário. Vale a pena salientar que esta interface é operada por este usuário há mais de dois anos.

Os resultados deste usuário foram utilizados como referência de desempenho para análise dos outros tipos de usuários.

Usuário sem experiência :

O usuário sem experiência (USR3), nesta primeira sessão, também teve bastante dificuldade nos diálogos dois e sete, durante a simulação. Vê-se, na figura 5.8, que há presença de erros e solicitação de ajuda em todos os diálogos, mas, nos diálogos dois e sete, a combinação erros versus solicitação de ajuda elevou o índice do percentual da duração desta sessão.

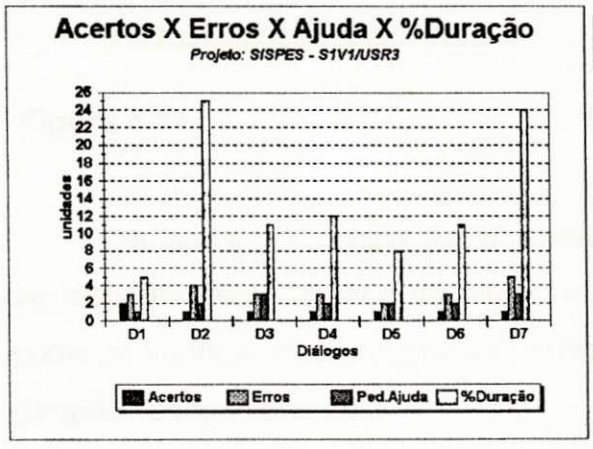


Figura 5.8

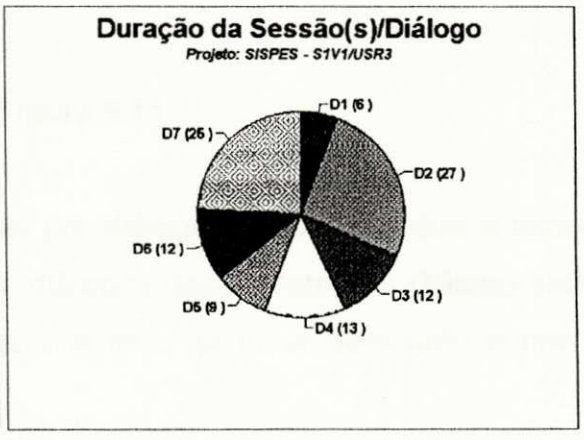


Figura 5.9

Na figura 5.9, duração da sessão por diálogo, serão confirmados problemas na interação nos diálogos D2 e D7, devido ao índice elevado do tempo de resposta do usuário. Nos outros diálogos, este tipo de usuário (USR3), mantém uma certa coerência no tempo de resposta, levando-se em consideração o seu perfil, que é o típico usuário leigo.

Usuário esporádico, com experiência:

O usuário esporádico, com experiência (USR4), da mesma forma que os outros usuários, demonstrou também dificuldades nos diálogos dois e sete. Na figura 5.10, verifica-se que, no diálogo D7, foi registrada a pior interação.

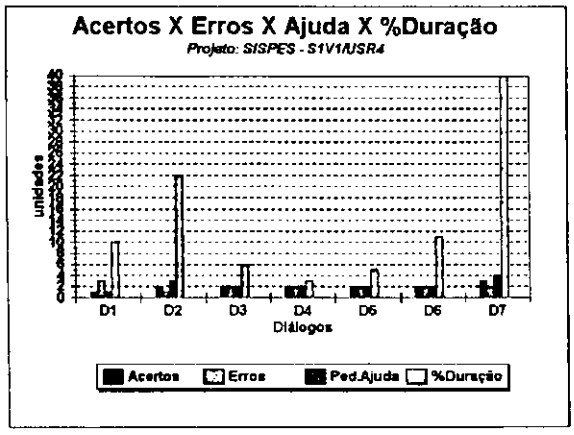


Figura 5.10

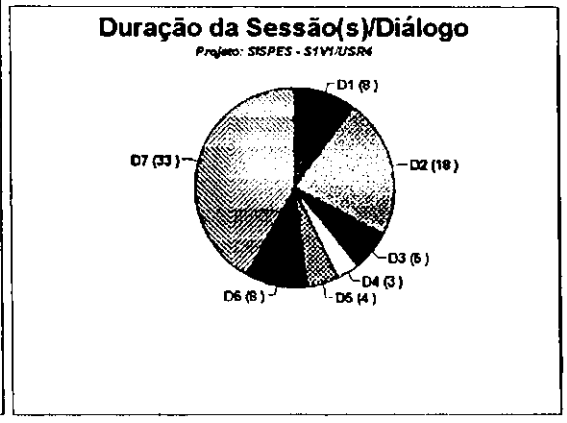


Figura 5.11

Na figura 5.11, duração da sessão por diálogo, constata-se que o tempo de resposta deste usuário foi maior nos diálogos dois e sete. No diálogo sete, pode-se verificar que, realmente, a figura anterior já tinha detectado o maior gargalo dessa interação.

Nesta primeira sessão, os dados apresentados pela ferramenta FAI ofereceram subsídios para identificar os diálogos-problema: D2 e D7.

Sessão S2 de coleta dos dados:

Como havia sido planejado no experimento, visando consolidar os resultados da primeira coleta e ainda observar a evolução na interação com a interface, foi feita uma segunda coleta, com um dia de intervalo.

Para o usuário USR1 - inexperiente, na segunda sessão da primeira versão, pode-se notar uma pequena melhoria nos índices da figura 5.12 e, quanto à duração da sessão (figura 5.13), o tempo total baixou, mas nos

diálogo D2 e D7 houve um pequeno acréscimo no tempo, o que leva a concluir que os outros diálogos tiveram um pequeno índice de aprendizado, enquanto que os diálogos D2 e D7, não.

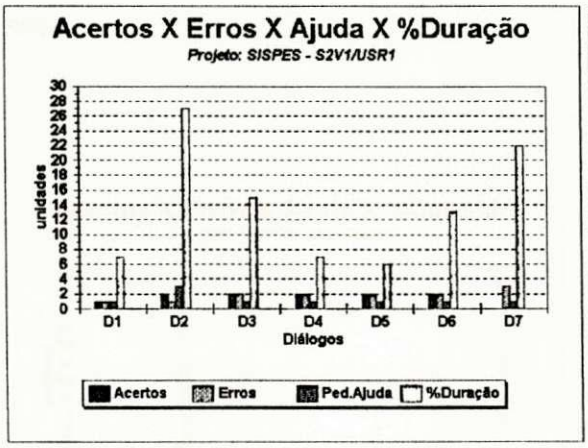


Figura 5.12

Figura 5.13

Para o usuário assíduo, com experiência (USR2), os resultados da coleta de dados, figura 5.14 e figura 5.15, não existe presença de erros nem solicitação de ajuda, e o índice de acertos é harmonioso.

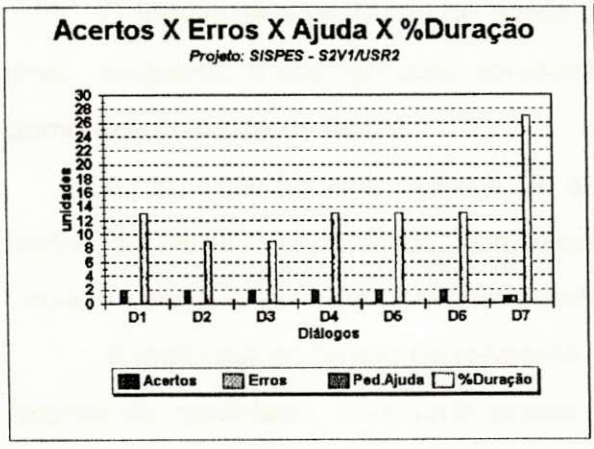


Figura 5.14

Figura 5.15

Na figura 5.15, a duração da sessão por diálogo tem um excelente tempo de resposta do usuário, confirmando, dessa forma, que é o usuário mais qualificado.

Conforme as figuras 5.16 e 5.17, o usuário esporádico, sem experiência (USR3) continua apresentando dificuldades de interação com os diálogos dois e sete.

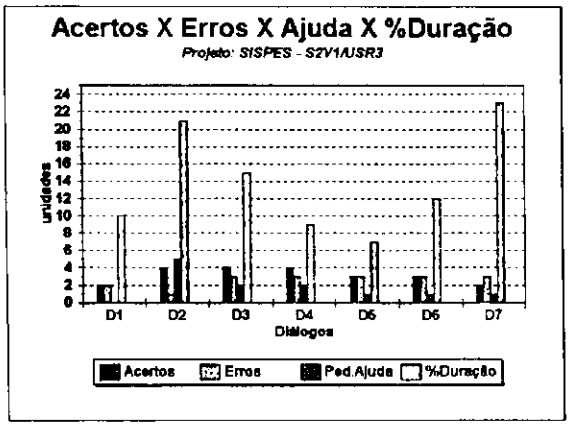


Figura 5.16

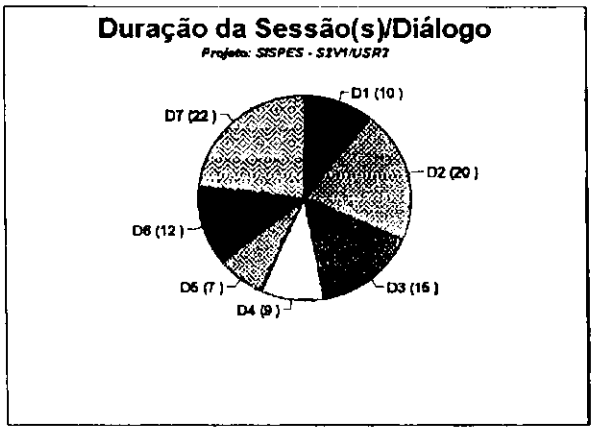


Figura 5.17

O índice de solicitação de ajuda ainda é bastante considerável no diálogo dois, enquanto o diálogo sete apresenta uma redução, mas, em compensação aumentou o índice de erros.

A partir das figuras 5.18 e 5.19, que apresentam os resultados da coleta de dados do usuário esporádico, com experiência (USR4), pode-se dizer que este usuário continua também com dificuldades nos diálogos dois e sete.

A melhoria do tempo de resposta, junto com as taxas indicativas dos outros fatores de qualidade, teve uma proporção muito semelhante para os usuários assíduo, sem experiência, e esporádico, sem experiência.

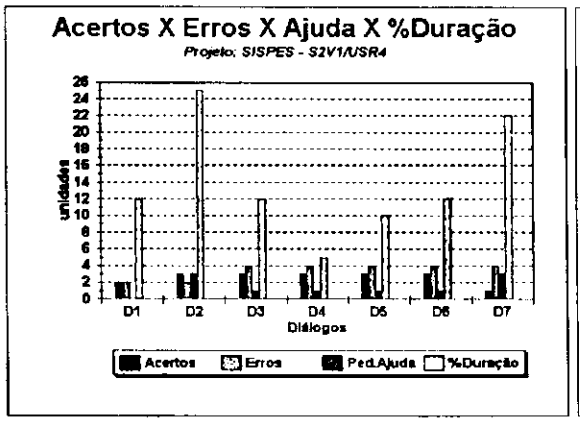


Figura 5.18

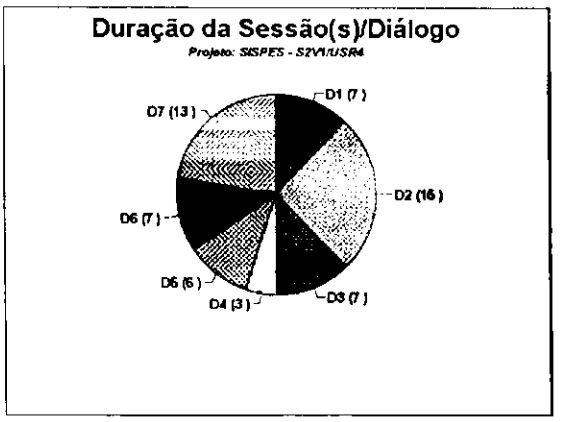


Figura 5.19

Da análise dos dados coletados nas duas sessões, com os quatro tipos de usuários, resultam informações suficientes para propor uma segunda versão deste sistema com modificações nos diálogos D2 e D7 de interface, tendo em vista que há evidências de que a maioria dos problemas existentes está concentrada nestes dois diálogos.

5.4.2 Modelo da interface para a segunda versão e sua análise

Tendo como base os diagnósticos apresentados na coleta de dados das duas sessões, na primeira versão, será apresentada a seguir, a interface modificada nos diálogos dois e sete, (figura 5.4.2.1) com alterações na quantidade de diálogos na versão dois do SISPES. Na primeira versão, havia sete diálogos e, nesta versão, existem oito diálogos disponíveis para executar as tarefas modeladas.

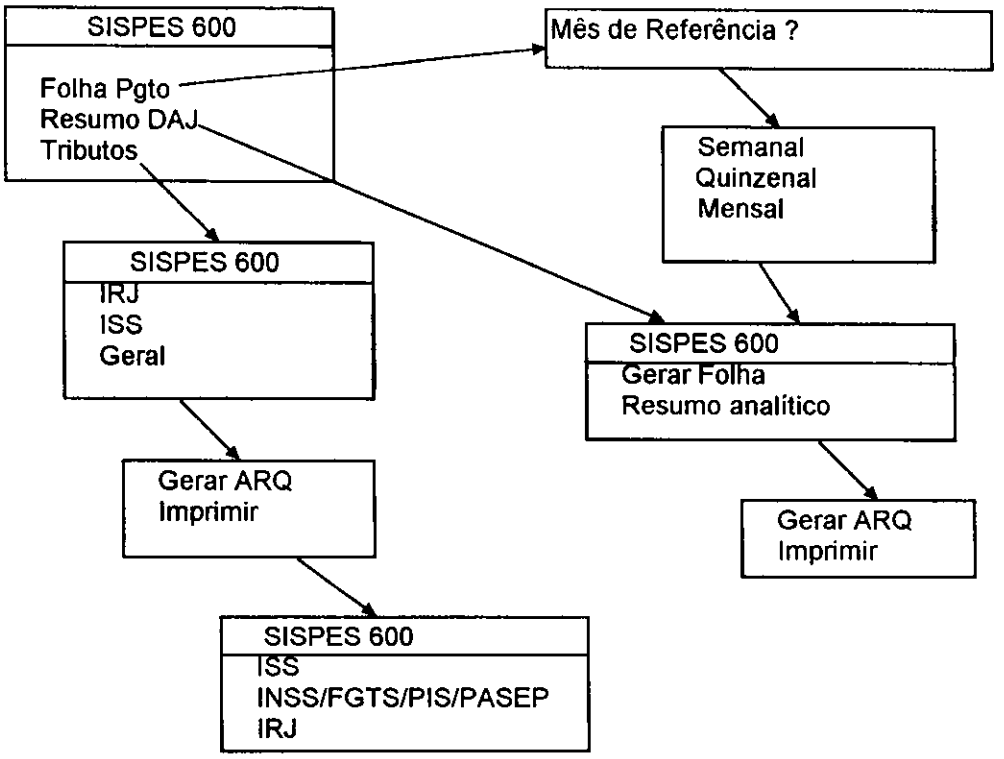


Figura 5.20 - Interface SISPES, segunda versão

Com estas pequenas modificações, a nova versão da interface foi submetida a testes de avaliação, com as mesmas tarefas e o mesmo número de sessões, com os quatro usuários. Essa nova versão tem um diálogo a mais, porque o "D2", da versão V1, teve sua sintaxe modificada, acrescentando-se um menu, de modo a melhorar a compreensão do usuário quanto ao sequenciamento da tarefa.

Findas as sessões, montou-se o gráfico 5.21, que retrata os erros, acertos e ajuda do USR1 ao longo das quatro sessões e das duas versões.

Neste gráfico 5.21, nota-se nitidamente que a segunda versão leva a um desempenho global bem melhor do que a primeira versão.

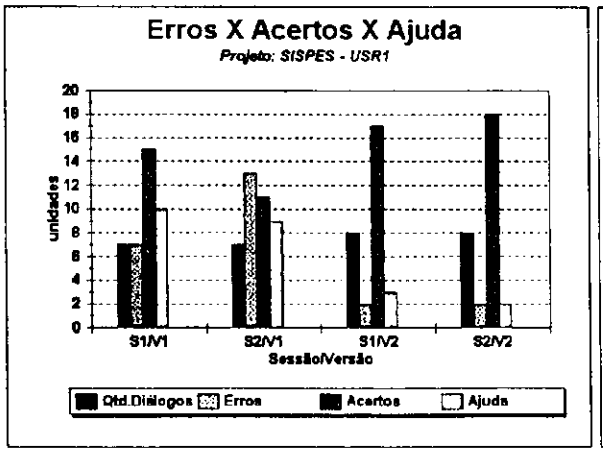


Gráfico 5.21

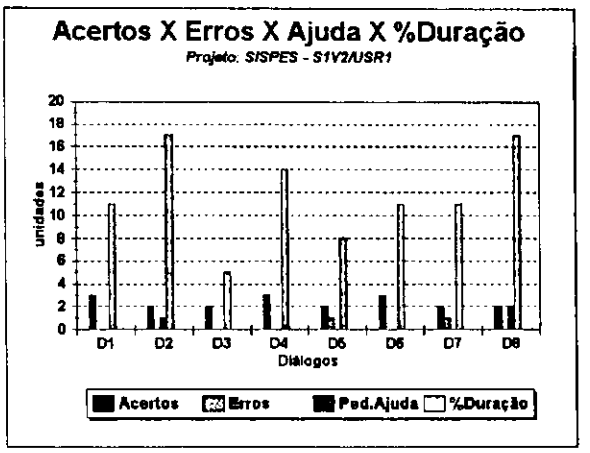


Gráfico 5.22

Pode-se verificar que o índice de acertos do USR1 aumentou na segunda versão, solicitando pouca ajuda em relação à primeira versão.

Na figura 5.22, são apresentados dados mais detalhados por diálogo, para o USR1, que também comprovam uma melhoria na distribuição do percentual do tempo de resposta, com uma certa estabilidade nos outros parâmetros de qualidade. Nota-se que os diálogos dois e sete, que foram os gargalos na primeira versão, já não se destacam nesta segunda versão.

O gráfico 5.23, que apresenta os resultados do USR2, permaneceu quase que inalterado em relação à versão V1, com um pequeno ganho na segunda sessão da segunda versão. Neste caso específico, o USR2 é o usuário mais qualificado para operar o SISPES.

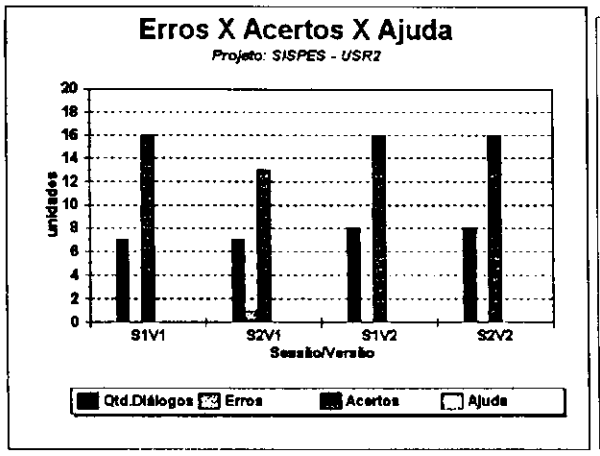


Gráfico 5.23

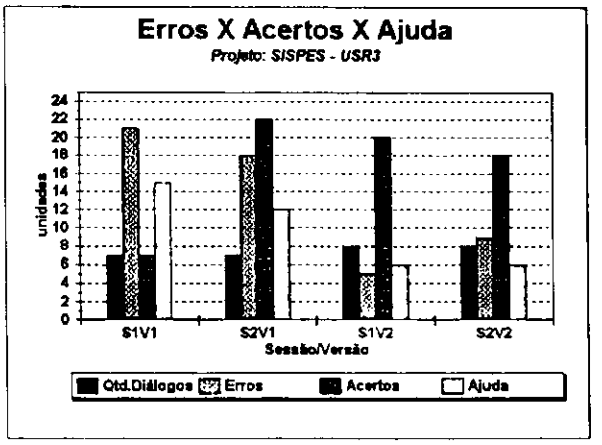


Gráfico 5.24

O USR3, que é o pior caso da versão V1, obteve um bom rendimento nas sessões da segunda versão. Na figura 5.24, observa-se uma queda nas taxas de erros e de solicitação de ajuda.

O USR4 evoluiu na segunda versão, tanto na taxa de erros quanto na de solicitação de ajuda, conforme se pode observar na figura 5.25.

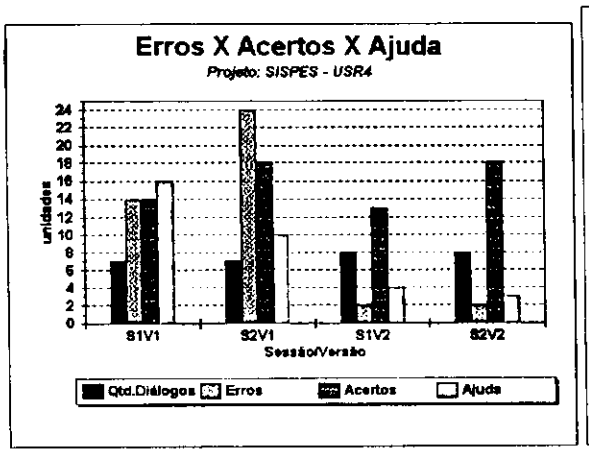


Gráfico 5.25

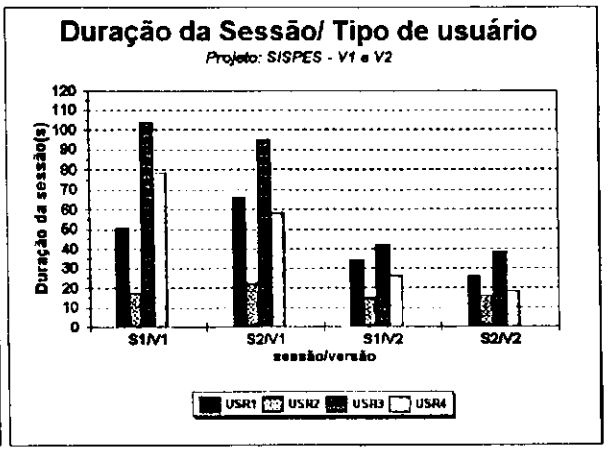


Figura 5.26

Para concluir, confirma-se, através da figura 5.26, a melhoria da interface na segunda versão, O tempo de resposta dos usuários USR1,USR3 e USR4, foi bem menor nas duas sessões da segunda versão do que na primeira versão. O usuário USR2 foi o único que manteve a duração das sessões bem estável.

Estas informações dão respaldo à afirmação de que a interface SISPES, segunda versão, ficou bem mais produtiva, particularmente para os usuários menos qualificados.

CAPÍTULO 6

CONCLUSÕES

CONCLUSÕES

Acredita-se que, ao concluir uma primeira versão do desenvolvimento da ferramenta de avaliação aqui descrita, se atingiu o principal objetivo do trabalho, ou seja, complementar o desenvolvimento do Ambiente AGILE, que, agora, passa a oferecer uma plataforma de estudos de interface para a área.

6.1 Discussão dos Resultados

Apesar de a proposta inicial da ferramenta ter como escopo a avaliação de todos os estilos de diálogo prototipados no ambiente AGILE, só foi possível avaliar dois desses estilos: Menu e Pergunta-e-resposta. A razão para esta limitação foi a dificuldade na interpretação do arquivo ".PTP" no trecho correspondente a esses diálogos. A estrutura deste arquivo, que é gerado pelo prototipador do AGILE (este armazena todos os dados relativos a um protótipo), foi resultado do desenvolvimento de duas dissertações e representa dados relativos ao código de dois programadores que já não recordam mais o significado de todos os itens, particularmente aqueles relativos aos dois estilos de interação excluídos na ferramenta de avaliação: Comando e Formulário.

Uma outra limitação da ferramenta é a falta de um gerenciador de arquivos que permitisse a correlação entre os dados das várias sessões de coleta. Na versão atual, isto é possível em um nível "macro", graças aos recursos de geração de gráficos na planilha eletrônica, mas ainda é muito limitado.

A apresentação dos resultados da coleta de dados em forma tabular no relatório de coleta é de difícil interpretação para os propósitos de comparação entre as sessões e detecção dos gargalos da interação. No entanto, o uso das planilhas exige uma certa familiarização do analista com a funcionalidade e

potencialidade da FAI e, ainda, com o uso da planilha para explorar melhor as informações disponíveis para análise.

Para uma avaliação adequada, é necessário um planejamento prévio do experimento, antes de iniciar a coleta de dados: especificar quais os aspectos da interface a serem modelados e determinar em que condições do ponto de vista do usuário, serão coletados estes dados. Por exemplo: com conhecimento ou sem ele ? Com a presença ou ajuda do analista, ou sem elas? Haverá limitação no tempo das sessões de interação? Qual o intervalo entre as sessões? Estas e outras perguntas presisam ser formuladas antes do início da avaliação, para que se tenha clareza na interpretação dos dados coletados.

Por conta da limitaçao no tempo disponível para coleta de dados, não foi possível expandir o número de sessões de coleta, de modo a permitir uma análise estatística ou a aplicação de um formalismo matemático sobre os dados coletados.

6.2 Testes

Os testes realizados com a FAI cobriram desde os testes dos módulos até os testes de integração com o ambiente AGILE. Na fase integração, foi onde surgiram os maiores problemas, devido à diferença de linguagem de programação: AGILE, escrito em C convencional, e a FAI, em C++.

Um dos fortes motivos para se migrar para "C++" foi a informação adquirida através do manual da BORLAND C++, versão 3.1, de que os dois códigos seriam integrados sem dificuldades, mas, na prática, houve grandes problemas, particularmente os de compartilhamento de dados de uma mesma estrutura de dados e os de passagem de parâmetros entre os módulos C e C++.

No entanto, esses problemas foram resolvidos com o compartilhamento de áreas entre os módulos, e os resultados dos testes efetuados até agora com os dois tipos de diálogos prototipados no AGILE foram positivos.

Além das dificulades de implementação, decorrentes da linguagem, uma outra dificuldade foi o entendimento do código AGILE, implementado em linguagem "C", por duas pessoas [SANTOS 92] e [PROCÓPIO 92], e cuja documentação não foi suficiente para permitir um desenvolvimento mais rápido e abrangente, como se pretendia.

Acrescentem-se a esses, problemas de memória, decorrentes do carregamento das estruturas do AGILE e da FAI. Estes problemas foram resolvidos com o compartilhamento das estruturas e com a implementação de um trecho de código para otimização da utilização de memória. Este código detecta a disponiblidade de memória e, quando esta atinge um nível crítico, ele salva em arquivo temporário as estruturas que não estão sendo utilizadas, permitindo sua recuperação, caso voltem a ser necessárias (ver Apêndice **B**).

6.3 Ambiente de instalação da FAI.

Para instalação da ferramenta, é necessária a instalação do AGILE na versão modificada, onde são feitas as alterações para ligação com a FAI. O código do AGILE ocupa cerca de 250KB de memória, enquanto que a FAI, 140KB. Juntos, correspondem a 7500 linhas de código compilado.

A configuração mínima necessária para instalar o ambiente AGILE, com o avaliador, é a seguinte: PC 286 ou 386, com 640kB de memória, monitor monocromático, mouse, disco rígido de 80MB. Para representar graficamente os resultados da coleta, é necessária uma planilha eletrônica, a exemplo da QUATRO-PRO versão 1.0, e do Windows, versão 3.1.

6.4 Sugestões para trabalhos futuros

Como o AGILE e, consequentemente, a FAI, ainda se encontram na fase embrionária no que diz respeito a se tornarem um produto de mercado,

ainda há muito trabalho a ser realizado. A seguir, são listadas algumas sugestões:

- A migração do AGILE/FAI para ambiente "WINDOWS", aproveitando toda a filosofia existente.

- Implementar recursos no coletor de dados, para coletar dinamicamente dados da interface de outro ambiente que não seja do AGILE.

- Expandir o conjunto de dados coletados pela FAI, de modo a considerar um conjunto mais abrangente de fatores de qualidade, passíveis de quantificação. Por exemplo, a inclusão do coletor de comandos para detectar o grau de utilização de cada comando projetado na interface. Esta informação permite verificar a adequação do conjunto de comandos da interface.

- Implementar um coletor de dados do tempo de resposta do sistema a uma ação do usuário.

- Implementar o coletor dos comandos utilizados em um contexto da interação, para permitir a avaliação do dimensionamento do conjunto de comandos projetado para a interface sob análise.

- Acrescentar características de inferência à ferramenta, permitindo uma análise assistida dos dados coletados, apoiando o bom senso do projetista na tomada de decisão de validar, ou não, a interface .

- Completar o desenvolvimento da nova versão (já iniciada neste trabalho) para o prototipador do ambiente AGILE. Esta nova versão, escrita em linguagem C++, seria mais facilmente integrada ao código da FAI do que a original utilizada neste trabalho.

• Incluir, no processo de avaliação, mais especificamente na análise dos dados coletados, métodos e ferramentas de análise estatística.

Apesar das limitações da ferramenta discutidas neste Capítulo, acredita-se que este trabalho tenha contribuído como um primeiro passo, para avaliações quantitativas de interfaces baseadas na coleta automática de dados.

REFERÊNCIAS BIBLIOGRÁFICAS

REFERÊNCIAS BIBLIOGRÁFICAS

BRANCO, S. O. **Especificação de um Sistema de Prototipagem Rápica de Interfaces Usuário-Computador**. Dissertação de Mestrado, COPIN- UFPB, 1990.

BEZERRA, J. **Avaliação Qualitativa de Interfaces Bancárias, com Projeto de Diretrizes de Projeto.** Dissertação de Mestrado, COPIN-UFPB, Junho/1992.

BEZERRA, J.; TURNELL, M. **Avaliação Qualitativa dos Bancos Eletrônicos - A opinião dos Usuários** . Anais do X SINAPE, 1992, p. 181-186.

COX, K.; WALKER, D. **User-Interface Design.** Prentice Hall, 1993.

DOWNTON, A. **Engineering the Human-Computer Interface.** McGraw-HILL Book Company, 1992.

ESTEVAM, R. C. **O Estudo sobre Desenvolvimento de Interfaces : Definição de Técnicas de Classificação e de Avaliação Baseadas na Satisfação do Usuário**. Dissertação de Mestrado da COPPE/UFRJ, Abril/1990.

GALITZ, W. O. **Handbook of Screen Format Design.** North-Holland, 2nd Ed. 1981.

GREEN, M. **The University of Alberta User Interface Management System.** Computer Graphcs, 19, 3, 1985, pp.205-213.

HilNNAMT, D. F. **Performance Measures.** Unix Review, vol.8, número 12, 1990, pp. 34-40.

KERNIGHAN, B. W.; RITCHIE, D. M. **C - A linguagem de Programação.** 1978.

KLEBER, C.; TURNELL, M. **Módulo Coletor de Dados de uma Ferramenta de Avaliação de Interfaces - FAI.** Relatório técnico do DEE- UFPB , junho 1993.

LAWSON, H. W. Jr.; BERTRAN, M. & SANAGUSTIN, J. **The Formal Definition of Humam/Machine Comunication.** Software-Practice and Experience, January/February 1978, pp. 51-58.

LOWGREN, J. **History, State and Future of User Interface Management Systems**. SIGCHI Bulletin, vol.,20, número,1, 1988, pp.38-44.

MARTIN, J. **Design of Man-Computer Dialogues.** Prentice-Hall, Englewood Cliffs, N.J. 1973

MORAN, P.T.; Card, K. **The Psychology of Humam - Computer Interaction.** Ed. Allen Nowell Laurence Erlbaum Associates, publishers, 1983

ORTIZ, S. **Sistemas Amigáveis.** XX Congresso de informática - 1987., pp. 1126-1130.

ORTH, A.I. **Controle de qualidade em Interfaces do usuário.** Relatório técnico do DCC-UFRGS, Agosto de 1989.

PROCÓPIO, C. **Desenvolvimento do Gerenciador de Diálogos e de Ferramentas de Prototipagem do Sistema AGILE.** Dissertação de Mestrado, COPIN-UFPB, Abril 1992.

QUEIROZ , J. E. **Validação de uma Metodologia de Projeto de Interfaces Usuário-Computador.** Dissertação de Mestrado, COPELE- UFPB. 1994.

QUEIROZ, J. E.; TURNELL, M. F. **Diretrizes para o Projeto de Interfaces**. Relatório técnico da COPELE, Junho 1992.

ROCHA, A. R. C. **Análise e Projeto Estruturado de Sistemas.** Editora Campus, 1987.

ROYBAL, R. **AIM Tecnology - Procuremente Guide.** AIM Tecnology 1990.

RUBIN, T. **User Interface Design for Computer Systems.** Ellis Horwood Limited, 1988.

RODNEY, M. C. B.; WALLERS, J. A. **A case study of user interface management system development and aplication.** Procedings of CHI'89 Human Factors in Computing Systems, ACM New York 1989, pp. 127-132.

SANTOS, R. **Um Gerenciador de Apresentação para o Sistema Agile de Prototipagem Rápida Usuário-Computador.** Dissertação de Mestrado, COPELE-UFPB, 1992.

SILVA, E. L. **Análise Ergonômica da Interface do AGILE e de suas Ferramentas (FAI) com base em Diretrizes de Programação Visual** . Relatório Técnico do DEE, Agosto 1994.

SMITH, J. E. **Characterizing Computer Performance With a Single Number.** Communications of the ACM, Outubro 1988, Vol. 31, Número 10, pp. 18 - 24.

SHNEIDERMAN, B. **Designing the User Interface: Strategies for Effective Humam-Computer Interaction.** Addison Wesley Publishing Company, USA 1987.

SHNEIDERMAN, B. **Human Factors Experiments in Designing Interactive Systems.** Universy of Maryland, IEEE dec. 1979.

TULLIS, T. **Designing a menu based interface for an operating system.** Procedings of CHI'85 Human Factors in Computing Systems, ACM New York 1985.

VAERTHIER, H.; ORTIZ, S. **Fatores Ergonômicos em Sistemas de Comunicação Homem-Máquina.** XVI Congresso Nacional de Informática, 1983., pp. 29-33.

WYRICK, R. **Benchmark Distributed Systems; Objective And Techniques.** Performance of Computer Installations ed. D. Ferrari, 1978, pp. 83-101.

WILLIGES, R. C. **The Use of Models in Humam-Computer Interface Design.** Ergonomics, vol 30, Número 3, 1987.

APÊNDICES

APÊNDICE A: DESCRIÇÃO DO ARQUIVO ".PTP" AGILE

Definição da apresentação dos diálagos AGILE [PROCÓPIO 92]

6 → Número de janela existente (diálogo)
"jtrab" → Identificação da janela (diálogo)

0 11 639 187 1 0 1 → Caractere de controle na apresentação
→ Coordenadas da janela no monitor de vídeo
0 → Número de Títulos

0 → Número de mensagens

"JINST"
0 0 0 0 0 0 0
0
0
"jmsg"
0 188 639 199 1 0 1
0
0
"jm1"
11 14 132 60 1 0 1 → Localização da Janela no monitor
0
0
"jm33"
183 60 383 110 1 3 15
0
0
"jp1"
102 29 384 48 1 0 1
0
0

Definição das Janelas

3 → Número de Menus

"ma1" → Identificação do Menu

1 3 1 1 0 1c0d 0 → Apresentação da tela (diálogo)

5 → Números de opções do Menu

"Prototipar" → ïtem do Menu

"1" → Número de opção do ítem

0 1 0 → Código de código de apresentação

```
"Simular"
"2"
0 1 0
"Avaliar (FAI)"
"3"
0 1 0

"Documentar"
"4"
0 1 0

"Sair"
"5"
0 1 0

"ma3"
1 3 1 1 0 1c0d 0
4

"Menu"
"1"
0 1 0

"Comando"
"2"
0 1 0

"Pergunta e Resposta"
"3"
0 1 0

"Formulario"
"4"
0 1 0

"MINST"
1 3 1 1 0 1c0d 0
0
0
```

1 ——→ Número de Pergnta e Resposta

"pa1" ——→ Identificação da Pergunta

1c0d 0 "Prototipo: " ——→ Ítem da Pergunta

1 ——→ Número de Formulário

"fsim" ——→ Identificação do Formulário

1d 0 ——→ Apresentação

2 ——→ Número de campos do Formulário

"Estilo de Interacao:" ⟶ Campo do Formulário

60 15 0 1 15
60 183 0 1 15 1
"Id. do Dialogo.....:"
120 15 0 1 15
120 183 0 1 15 1

⟶ Apresentação,Definição dos Campos e

1
"e5"
0 1 15
"Nao Existe Apresentacao Definida"

Definição das mensagens

---

2 ⟶ Número de Menus

"m1" ⟶ Identificação do Menu

"ma1" ⟶ Iidentificação da apresentação do Menu

"jm1" ⟶ Janela onde o Menu será apresentada

20 7 ⟶ Posição na Janela

5 ⟶ Número de opções

1 Número de ações
9 ⟶ Ação
0 Continuar diálogo

⟶ Opção 1

Gerenciador de diálogos

1 ⟶ Número de ações

1
3 0 "p3" ⟶ Ação ativar diálago

⟶ Opção 2

1 ⟶ Número de ações

13 ⟶ Chamar Procedimento

5 ⟶ Posição do vetor de Procedimentos

⟶ Opção 3

1 Número de ações

9
Continuar Diálogo
0

⟶ Opção 4

1 Número de ações

3

Encerra simulação

0

Opção 5

0
0
"m33"
"ma3"
"jm33"
8 8
4
4
13
3
2
0
13
4
10
4 1 "fsim"
4
13
3
2
0
13
4
10
4 1 "fsim"
4
13
3
2
0
13
4
10
4 1 "fsim"
4
13
3
2
0
13
4
10
4 1 "fsim"
0

```
1
"t1"
1
0 27
```
→ Identificação da Tecla aplicável (ESC)

```
1
6
0 0 ""
0
```
→ Ação da Tecla

1 → Número da pergunta e Resposta

"p3" → Identificação da Pergunta e Resposta

"pa1" → Identificação da apresentação da P/R

"jp1" → Identificação da Janela para apresentação

5 10 → Localização dentro da Janela

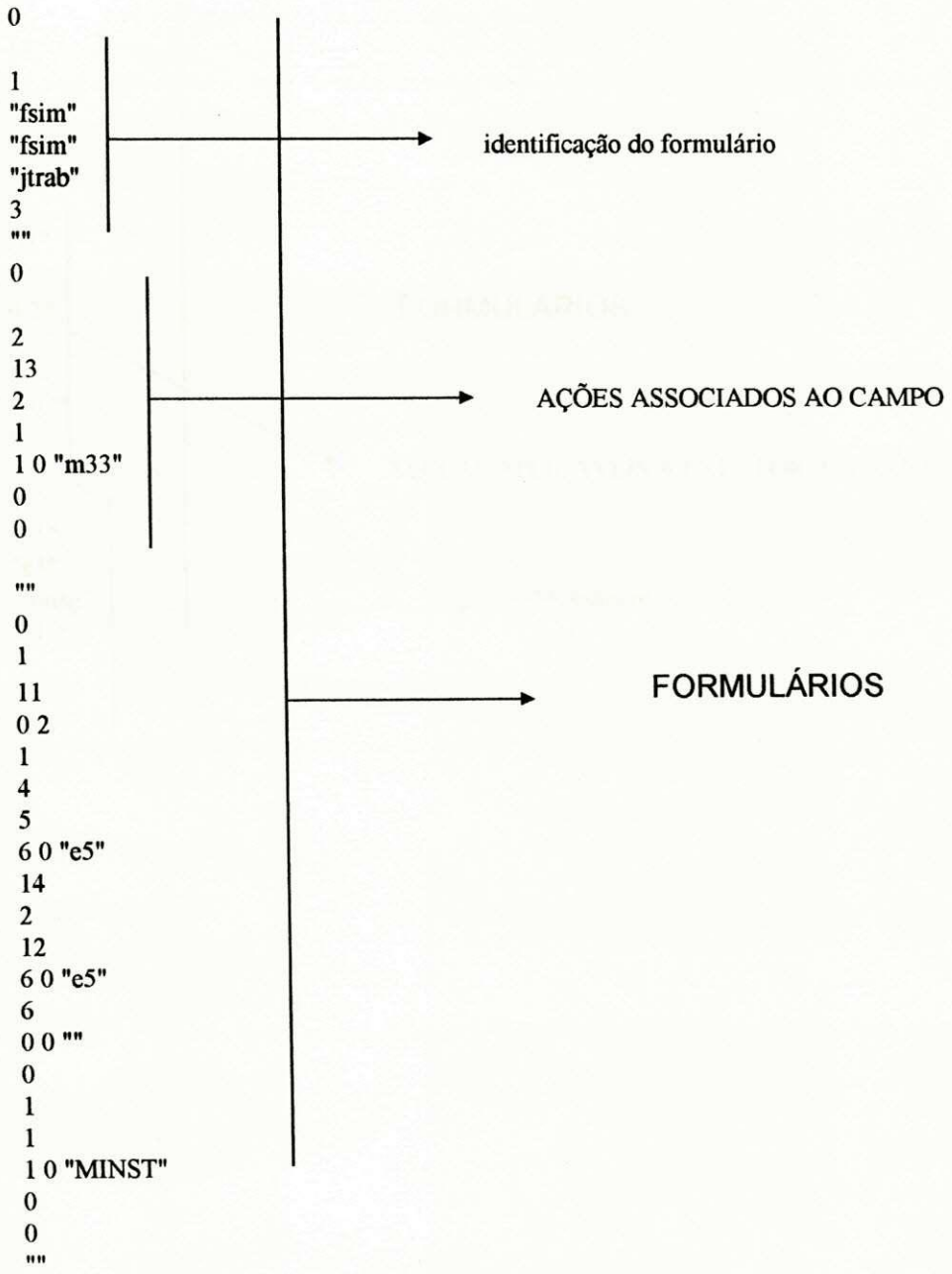
1 → Número de Resposta

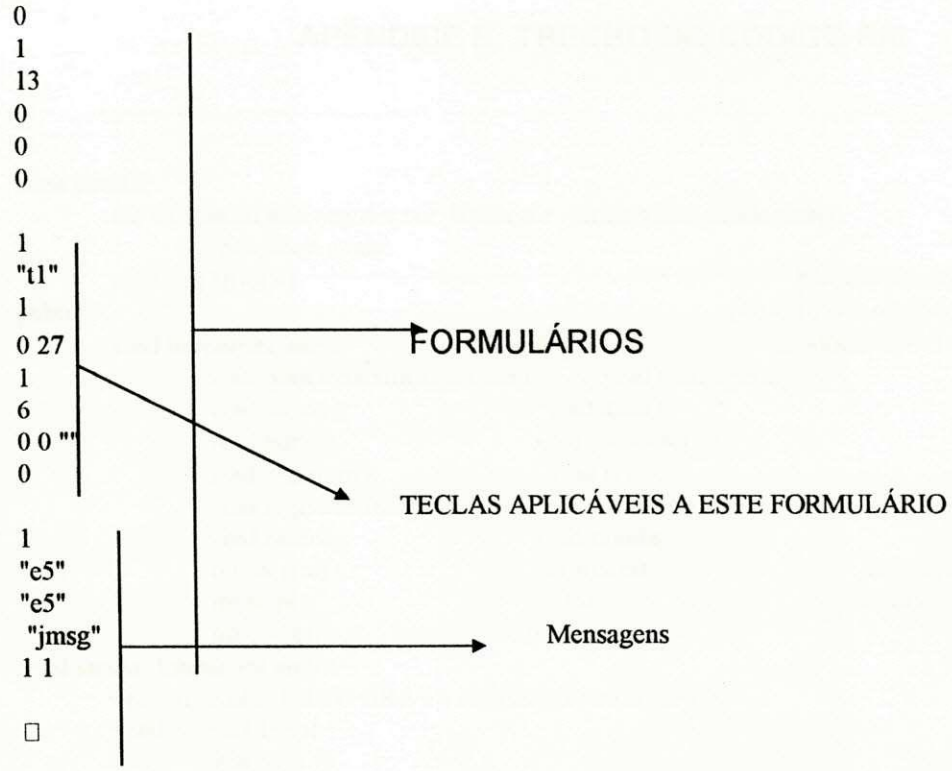
"r1" → Identificação da Resposta

"" Tipo de Resposta que o diálogo aguarda

```
3
13
1
2
0
1
4 0 "fsim"
```
→ Ações associadas a Resposta - Número de ações,Tipo da ação

```
1
"t1"
1
0 27
1
6
0 0 ""
```
→ Teclas aplicáveis


0
1
"fsim"
"fsim"
"jtrab"
3
""
0
2
13
2
1
1 0 "m33"
0
0
""
0
1
11
0 2
1
4
5
6 0 "e5"
14
2
12
6 0 "e5"
6
0 0 ""
0
1
1
1 0 "MINST"
0
0
""
identificação do formulário
AÇÕES ASSOCIADOS AO CAMPO
FORMULÁRIOS


0
1
13
0
0
0
1
"t1"
1
0 27
1
6
0 0 ""
0
1
"e5"
"e5"
"jmsg"
1 1
FORMULÁRIOS
TECLAS APLICÁVEIS A ESTE FORMULÁRIO
Mensagens

APÊNDICE B: TRECHO DO CÓDIGO FAI

```
class janela{
        int x,y,funcao,min,seg,dec,cor_borda,cor_caracter,cor_fundo,tempo,
                erros,acert,ajuda;
        char vet[3][total];
public:
        void limpar(int amb);              void tela();
                void posicionar(int fc,int amb);    void mensagem();
                void menu();                 void data();
                void hora();                 void cronometro();
                void executar();             void reset();
                void registrar(int tmp);       void rerros();
                void racert();               void rajuda();
                int merros();                int macert();              int majuda();
                int tmp();                   int func();              int corf();
                int corb();                  int corc();};
void janela::limpar(int amb){
        int f; textcolor(LIGHTGRAY); textbackground(BLACK);
        for(f=2;f<24;f++){
                gotoxy(1,f);

        if(amb)cprintf("%s","________________________________________________________
________________");
                else  cprintf("%s","                                                   ");}}

void janela::tela(){
        textcolor(cor_borda); textbackground(cor_fundo);
        gotoxy(x,y);   cprintf("+--------------+");
        gotoxy(x,y+1);  cprintf("¦              ¦_");
        gotoxy(x,y+2);  cprintf("¦              ¦_");
        gotoxy(x,y+3);  cprintf("¦              ¦_");
        gotoxy(x,y+4);  cprintf("+--------------+_");
        gotoxy(x+1,y+5); cprintf("________________");}
void janela::posicionar(int fc,int amb){
        char o,*sd; int cor; min=0; seg=0; dec=0; tempo=0;
        funcao=fc; x=32; y=10; erros=0; acert=0;
        if(fc==1||fc==2) for(o=0;o<3;o++) {
           gotoxy(10,24); printf("%s%d%s"," Entre com a mensagem da ",o+1,"a. linha (15 letras):
");

           gotoxy(58,24); sd=readstr(15); strcpy(vet[o],sd);
           while(strlen(vet[o])<15) strcat(vet[o]," "); vet[o][16]='\0';}
        do{ gotoxy(10,24); printf("%s"," Entre cor borda (0..15):                         ");
           gotoxy(36,24); cor_borda=readint(2);} while(!(cor_borda>=0&&cor_borda<=15));
        do{ gotoxy(10,24); printf("%s"," Entre cor caracter (0..15):                      ");
           gotoxy(39,24); cor_caracter=readint(2);} while(!(cor_caracter>=0&&cor_caracter<=15));
        do{ gotoxy(10,24); printf("%s"," Entre cor fundo (0..7):                          ");
           gotoxy(35,24); cor_fundo=readint(1);} while(!(cor_fundo>=0&&cor_fundo<=7));
        gotoxy(10,24); printf(" Posicione a janela.                                 ");
        _setcursortype(_NOCURSOR);
        while(o!=13){
```

(Anotação junto a `char vet[3][total];`: → Vetor de objeto, onde cada posição do vetor corresponde a um Diálogo com todos os seus atributos)

```
                limpar(amb); gotoxy(x,y); tela(); o=getch();
                if(o==72&&y>2)y--; if(o==80&&y<18)y++;
                if(o==75&&x>1)x--; if(o==77&&x<63)x++;}}
void janela::mensagem(){
        textcolor(cor_caracter); textbackground(cor_fundo);
        gotoxy(x+1,y+1); cprintf("%s",vet[0]);
        gotoxy(x+1,y+2); cprintf("%s",vet[1]);
        gotoxy(x+1,y+3); cprintf("%s",vet[2]);
        while(!kbhit());}
void janela::menu(){
        int o=0,f=0,k=0;
        while(1){
                for(k=0;k<3;k++){
                        textcolor(cor_caracter); textbackground(cor_fundo);
                        if(k==f){textcolor(cor_fundo); textbackground(cor_caracter);};
                        gotoxy(x+1,y+k+1); cprintf("%s",vet[k]);}
                o=getch(); if(o==72&&f>0)f--; if(o==80&&f<2)f++;
                if(o==13||o==32||o==27) break;
                if(o==59) { rajuda(); break;};
                if(o>=48 & o<=57) break;}}
void janela::data(){
        struct date d; getdate(&d); textcolor(cor_caracter);
        textbackground(cor_fundo); gotoxy(x+4,y+1); cprintf("%s","Data..");
        gotoxy(x+4,y+2); cprintf("%02d/%02d/%d",d.da_day,d.da_mon,d.da_year);
        while(!kbhit());}
void janela::hora(){
        struct time t; textcolor(cor_caracter); textbackground(cor_fundo);
        gotoxy(x+4,y+1); cprintf("%s","Hora..");
        while(!kbhit()){
        gettime(&t); gotoxy(x+4,y+2);
        cprintf("%02d:%02d:%02d",t.ti_hour,t.ti_min,t.ti_sec);}}
void janela::cronometro(){
        textcolor(cor_caracter); textbackground(cor_fundo);
        gotoxy(x+2,y+1); cprintf("%s","Cronometro..");
        while(!kbhit()){ delay(10);
        dec++; if(dec>=100){dec=0;seg++;} if(seg==60){seg=0;min++;}
        gotoxy(x+4,y+2); cprintf("%02d:%02d:%02d  ",min,seg,dec);}}
void janela::executar(){
        tela();
        switch(funcao){
                case 1: mensagem();  break;
                case 2: menu();      break;
                case 3: data();      break;
                case 4: hora();      break;
                case 5: cronometro(); break;}}
void janela::reset(){
        tempo=0; erros=0; acert=0; ajuda=0;}
void janela::registrar(int tmp){
        tempo+=tmp;}
void janela::rerros(){
        erros++;}
void janela::racert(){
        acert++;}
void janela::rajuda(){
        int tp; textcolor(WHITE); textbackground(RED);
        for(tp=10;tp<=15;tp++){ gotoxy(10,tp);
```

```
                cprintf("                                ");}
        caixa(10,10,15,50,1);
        gotoxy(12,12); printf("   Voce solicitou ajuda (Help).");
        gotoxy(12,13); printf(" Porem ainda nao foi implementado.");
        while(!kbhit()); ajuda++;}
int janela::merros(){
        return erros;}
int janela::macert(){
        return acert;}
int janela::majuda(){
        return ajuda;}
int janela::tmp(){
        return tempo;}
int janela::func(){
        return funcao;}
int janela::corf(){
        return cor_fundo;}
int janela::corb(){
        return cor_borda;}
int janela::corc(){
        return cor_caracter;}

// funcao para abrir o modo grafico
void opengraph(void) {
  int gd=DETECT,gm,ec; initgraph(&gd,&gm,""); ec=graphresult();
  if (ec!=grOk) {
    printf("Erro no modo grafico: %s\n",grapherrormsg(ec));
    printf("Qualquer tecla continua."); getch(); exit(1);}}

// informa pagina ativa do video
int dizpg(){
        union REGS regs; regs.h.ah=0x0f;
        int86(0x10,&regs,&regs); return(regs.h.bh);}

// informa o atributo do caracter na posicao do cursor
unsigned char atributo(){
        union REGS regs; regs.h.bh=dizpg(); regs.h.ah=8;
        int86(0x10,&regs,&regs); return(regs.h.ah);}

// informa o caracter presente na posicao do cursor
unsigned char caracter(){
        union REGS regs; regs.h.ah=8; regs.h.bl=dizpg();
        int86(0x10,&regs,&regs); return(regs.h.al);}

// muda o atributo dos caracteres iniciando posicao do cursor
void attr(unsigned char atrib,unsigned int vezes){
        union REGS regs;
        while(vezes--){
                regs.h.ah=9; regs.h.al=caracter(); regs.h.bh=dizpg();
                regs.h.bl=atrib; regs.x.cx=1; int86(0x10,&regs,&regs);
                gotoxy(wherex()+1,wherey());}}

// coloca o caracter c na posicao atual do cursor e repete
void replicate(char c,unsigned int vezes){
        union REGS regs;
        regs.h.ah=10; regs.h.al=c; regs.h.bh=dizpg();
```

```
        regs.h.bl=atributo(); regs.x.cx=vezes; int86(0x10,&regs,&regs);}

// faz uma janela
void caixa(int l1,int c1,int l2,int c2,int tipo){
        int i;
        static int quina[2][4]={{218,192,191,217},{201,200,187,188}};
        static int linha[2][2]={{196,179},{205,186}}; tipo--;
        gotoxy(c1,l1); replicate(quina[tipo][0],1);
        gotoxy(c1,l2); replicate(quina[tipo][1],1);
        gotoxy(c2,l1); replicate(quina[tipo][2],1);
        gotoxy(c2,l2); replicate(quina[tipo][3],1);
        gotoxy(c1+1,l1); replicate(linha[tipo][0],c2-c1-1);
        gotoxy(c1+1,l2); replicate(linha[tipo][0],c2-c1-1);
        for(i=l1+1;i<l2;++i){
                gotoxy(c1,i);   replicate(linha[tipo][1],1);
                gotoxy(c1+1,i); replicate(' ',c2-c1-1);
                gotoxy(c2,i);   replicate(linha[tipo][1],1);}}

// semelhante ao achoice do Clipper menu(vetor,x,y,itens,tipo_de_janela)
int menu(char *ver[],int x,int y,int h,int t){
        int marc=0,ini=0,cont,op=0;
        // verifica o tamanho da caixa de menu
        for(cont=0;ver[cont];cont++){
                if(strlen(ver[cont])>op) op=strlen(ver[cont]);}
        // providencia a caixa
        caixa(y-1,x-1,y+h,x+op,t);
        // faz o menu rolante e faz retornar o valor correspondente
        op=0; while(op!=13){
                for(cont=ini;(cont<=h+ini && ver[cont-ini]);cont++){
                        if (cont!=marc+ini) {
                                textcolor(WHITE);
                                textbackground(BLACK);
                                gotoxy(x,y+cont-ini);
                                cprintf("%s",ver[cont]);
                        } else {
                                textcolor(BLACK);
                                textbackground(WHITE);
                                gotoxy(x,y+cont-ini);
                                cprintf("%s",ver[cont]);}}
                op=getch(); switch(op){
                        case 72: if (marc>0) {marc--;} else
                                 if (marc==0 && ini>0) {ini--;}; break;
                        case 80: if (marc<h && ver[marc+ini+1]){marc++;} else
                                 if (marc==h && ver[h+ini+1]) {ini++;}; break;
                        case 27: return 27;
                        case 75: return 75;
                        case 77: return 77;}}
        return (marc+ini+1);}

int ver(struct time tp1,struct time tp2){
        int l1,l2;
        l1=tp1.ti_sec; l1+=tp1.ti_min*60; l1+=tp1.ti_hour*360;
        l2=tp2.ti_sec; l2+=tp2.ti_min*60; l2+=tp2.ti_hour*360;
        return l2-l1;}

void mud_amb(int modo,int i){
```

```
        int f; if(modo==1){
                if(!i){
                        textcolor(BLACK); textbackground(WHITE); gotoxy(1,1);
                        cprintf("%s","   Arquivos  Testes   Ambiente  Modo
");
                        textcolor(LIGHTGRAY); textbackground(BLACK);
                        for(f=2;f<24;f++){ gotoxy(1,f);
                        cprintf("%s","                                                    ");}
                } else {
                        textcolor(BLACK); textbackground(WHITE); gotoxy(1,1);
                        cprintf("%s","   Arquivos  Testes   Ambiente  Modo
");
                        textcolor(LIGHTGRAY); textbackground(BLACK);
                        for(f=2;f<24;f++){ gotoxy(1,f);

        cprintf("%s","_____________________________________________________________
_____");}}
                gotoxy(10,24);cprintf("%s","                                                    ");}}


int readint(int tam){
        int i=-1,c,cont1; char *s;
        int x=wherex(), y=wherey();
        strcpy(s,""); cont1=0;
        while (1){
                c=0; gotoxy(x,y); c=getch();
                if(c>=48 && c<=57 && cont1<tam){ s[cont1]=c; cont1++; s[cont1]=NULL; x++;
printf("%c",c);};
                if(c==8 && cont1>0){ s[cont1]=NULL; cont1--; x--; printf("\b%s\b"," ");};
                if(c==13){if(strlen(s)>0){i=atoi(s); return(i);}
                                else return(-1);};
                if(c==27){return(-1);};}}

char *readstr(int tam){
        int c,cont1; char *s;
        int x=wherex(), y=wherey();
        strcpy(s,""); cont1=0;
        while (1){
                c=0; gotoxy(x,y); c=getch();
                if(c>=32 && c<=125 && cont1<tam){ s[cont1]=c; cont1++; s[cont1]=NULL; x++;
printf("%c",c);};
                if(c==8 && cont1>0){ cont1--; s[cont1]=NULL; x--; printf("\b%s\b"," ");};
                if(c==13){return(s);}}}

int prototipar(janela vet[],int amb){
        int total,cont,resp,lin,col;
        textcolor(WHITE); textbackground(BLACK); _setcursortype(_NORMALCURSOR);
        do {gotoxy(10,24); printf("%s"," Total de dialogos (0..9):          ");
        gotoxy(37,24); total=readint(1);} while(total<0|total>9);
        if(!total) return 0;
        for(cont=0;cont<total;cont++){
                _setcursortype(_NOCURSOR);
                textcolor(LIGHTGRAY); textbackground(BLACK);
                for(resp=5;resp<=22;resp++){
                        gotoxy(10,resp); cprintf("%s","                                                    ");}
                caixa(5,10,22,73,1);
```

```
                gotoxy(12, 7); cprintf("%s"," Defina os dialogos a seguir.");
                gotoxy(12, 8); cprintf("%s"," Opcoes para funcao de dialogo..");
                gotoxy(17,10); cprintf("%s"," 1. Mensagem");
                gotoxy(17,11); cprintf("%s"," 2. Menu");
                gotoxy(17,12); cprintf("%s"," 3. Data");
                gotoxy(17,13); cprintf("%s"," 4. Hora");
                gotoxy(17,14); cprintf("%s"," 5. Cronometro");
                gotoxy(12,17); cprintf("%s"," Cores..");
                for(lin=0;lin<=1;lin++)
                        for(col=0;col<=7;col++){
                                gotoxy(col*5+27,lin*2+17);
                                textcolor(lin*8+col); cprintf("%s","_____");}
                textcolor(WHITE); gotoxy(13,17);
                for(lin=0;lin<=1;lin++)
                        for(col=0;col<=7;col++){
                                gotoxy(col*5+30,lin*2+18);
                                cprintf("%2d",lin*8+col);}
                _setcursortype(_NORMALCURSOR);
                do{ gotoxy(10,24);
                printf("%s%d%s"," Funcao do dialogo ",cont+1," (1..5):     ");
                gotoxy(40,24); resp=readint(1);} while(resp<1|resp>5);
                vet[cont].posicionar(resp,amb);}
        return total;}

void simular(janela vet[],int total,int amb){
        struct time tp1,tp2; int op1=0,cont=0;
        _setcursortype(_NOCURSOR);
        while(op1<total){vet[op1].reset(); op1++;}
        while(1){
                textcolor(WHITE); textbackground(BLACK);
                gotoxy(10,24); cprintf("%s%d%s","Visualizacao dialogo ",cont+1," <Espaco> Termina");
                vet[cont].limpar(amb); gettime(&tp1);
                vet[cont].executar(); op1=getch(); gettime(&tp2);
                vet[cont].registrar(ver(tp1,tp2));
                if(op1>=49&&op1<=total+48) cont=op1-49;
                if(op1==59) {vet[cont].rajuda();}
                if(op1==13) {vet[cont].racert(); cont++; if(cont>=total) cont=0;}
                if(op1==27) {vet[cont].rerros(); cont--; if(cont<0) cont=total-1;}
                if(op1==32) break;}
        while(!kbhit());}

void avaliar(janela vet[],int total){
        int op1=0,cont,tmp=0,terros,tacert,tajuda;
        FILE *pr; char op2,*nome;
        char *s[5]={" Assiduo sem experiencia    ",
                    " Assiduo com experiencia    ",
                    " Exporadico sem experiencia ",
                    " Exporadico com experiencia "};
        //char *fc[5]={"Mensagem","Menu","Data","Hora","Cronometro"};
        gotoxy(10,24); printf("%s","Entre com o tipo de usuario da interface.");
        textcolor(LIGHTGRAY); textbackground(BLACK);
        for(tmp=3;tmp<=7;tmp++){gotoxy(3,tmp); cprintf("%s","                          ");}
        op1=menu(s,4,4,4,1);
        gotoxy(10,24); printf("%s","                                    ");
        if(op1==27|total==0)return;
        textcolor(LIGHTGRAY); textbackground(BLACK);
```

```
for(tmp=4;tmp<=22;tmp++){gotoxy(5,tmp); cprintf("%s","                                                    ");}
caixa(4,5,22,65,1);
gotoxy(7,5); cprintf("%s","Avaliacao da ultima Simulacao..:");
gotoxy(7,7); cprintf("%s%s",   "Tipo de usuario   :",s[op1-1]);
gotoxy(7,8); cprintf("%s%3d",  "Total de dialogos  : ",total);
terros=0; for(cont=0;cont<total;cont++) terros+=vet[cont].merros();
gotoxy(7,9); cprintf("%s%3d",  "Erros cometidos   : ",terros);
tacert=0; for(cont=0;cont<total;cont++) tacert+=vet[cont].macert();
gotoxy(7,10); cprintf("%s%3d", "Acertos registrados: ",tacert);
tajuda=0; for(cont=0;cont<total;cont++) tajuda+=vet[cont].majuda();
gotoxy(7,11); cprintf("%s%3d", "Total Ajuda solict.: ",tajuda);
tmp=0; for(cont=0;cont<total;cont++) tmp+=vet[cont].tmp();
gotoxy(7,12); cprintf("%s%3d%s","Tempo de simulacao : ",tmp," seg.");
gotoxy(7,13); cprintf("%s","Grau de interesse..: ");
gotoxy(7,14); cprintf("%s","Dialogo   :");
gotoxy(7,15); cprintf("%s","Funcao    :");
gotoxy(7,16); cprintf("%s","Cores (FBC):");
gotoxy(7,17); cprintf("%s","Acertos   :");
gotoxy(7,18); cprintf("%s","Erros     :");
gotoxy(7,19); cprintf("%s","Ped. Ajuda :");
gotoxy(7,20); cprintf("%s","Tempo     :");
gotoxy(7,21); cprintf("%s","Porcent (%):");
for(cont=0;cont<total;cont++){
        textcolor(LIGHTGRAY);
        gotoxy(cont*4+21,14); printf("%3d",cont+1);
        gotoxy(cont*4+21,15); printf("%03d",vet[cont].func());
        gotoxy(cont*4+21,17); printf("%03d",vet[cont].macert());
        gotoxy(cont*4+21,18); printf("%03d",vet[cont].merros());
        gotoxy(cont*4+21,19); printf("%03d",vet[cont].majuda());
        gotoxy(cont*4+21,20); printf("%03d",vet[cont].tmp());
        gotoxy(cont*4+21,21); printf("%03d",tmp?vet[cont].tmp()*100/tmp:0);
        gotoxy(cont*4+21,16);
        textcolor(vet[cont].corf()); cprintf("%s","_");
        textcolor(vet[cont].corb()); cprintf("%s","_");
        textcolor(vet[cont].corc()); cprintf("%s","_");}
gotoxy(10,24); printf("%s"," Saida do relatorio em (I)mpressora ou (A)rquivo. ");
op2=getch(); op2=toupper(op2);
while (op2=='I'||op2=='A'){
  if(op2=='I'){
        if((pr=fopen("PRN","w"))==NULL){
                gotoxy(10,24); printf("%s"," Nao posso abrir impressora..                    ");
                delay(2000); while(!kbhit()); break;}
                textcolor(BLACK);
                fprintf(pr,"\n%s","Avaliacao da ultima Simulacao..:");
                fprintf(pr,"\n\n%s%s", "Tipo de usuario   :",s[op1-1]);
                fprintf(pr,"\n%s%3d",  "Total de dialogos  : ",total);
                fprintf(pr,"\n%s%3d",  "Erros cometidos   : ",terros);
                fprintf(pr,"\n%s%3d",  "Acertos registrados: ",tacert);
                fprintf(pr,"\n%s%3d",  "Total Ajuda solict.: ",tajuda);
                fprintf(pr,"\n%s%3d%s","Tempo de simulacao : ",tmp," seg.");
                fprintf(pr,"\n\n%s","Grau de interesse..: ");
                fprintf(pr,"\n%s","Dialogo   :");
                for(cont=0;cont<total;cont++){
                        fprintf(pr,"%4d",cont+1);}
                fprintf(pr,"%s","\nFuncao    :");
                for(cont=0;cont<total;cont++){
```

```
                    fprintf(pr,"%c%03d",' ',vet[cont].func());}
                fprintf(pr,"%s","\nCores (FBC):");
                for(cont=0;cont<total;cont++){
                    textcolor(vet[cont].corf()); fprintf(pr," %s","_");
                    textcolor(vet[cont].corb()); fprintf(pr,"%s","_");
                    textcolor(vet[cont].corc()); fprintf(pr,"%s","_");}
                fprintf(pr,"%s","\nAcertos   :");
                for(cont=0;cont<total;cont++){
                    fprintf(pr," %03d",vet[cont].macert());}
                fprintf(pr,"%s","\nErros     :");
                for(cont=0;cont<total;cont++){
                    fprintf(pr," %03d",vet[cont].merros());}
                fprintf(pr,"%s","\nPed. Ajuda :");
                for(cont=0;cont<total;cont++){
                    fprintf(pr," %03d",vet[cont].majuda());}
                fprintf(pr,"%s","\nTempo (seg):");
                for(cont=0;cont<total;cont++){
                    fprintf(pr," %03d",vet[cont].tmp());}
                fprintf(pr,"%s","\nPorcent (%):");
                for(cont=0;cont<total;cont++){
                    fprintf(pr," %03d",tmp?vet[cont].tmp()*100/tmp:0);}
                fprintf(pr,"\n\n\n\n\n");
        fclose(pr); break;}
if(op2=='A'){
        gotoxy(10,24); printf("%s"," Entre com o nome do arquivo:                    ");
        gotoxy(40,24); nome=readstr(12);
        if((pr=fopen(nome,"w"))==NULL){
                gotoxy(10,24); printf("%s"," Nao posso abrir arquivo..                ");
                delay(2000); while(!kbhit()); break;}
                fprintf(pr,"\n%s","Avaliacao da ultima Simulacao..:");
                fprintf(pr,"\n\n%s%s", "Tipo de usuario   :",s[op1-1]);
                fprintf(pr,"\n%s%3d",  "Total de dialogos  : ",total);
                fprintf(pr,"\n%s%3d",  "Erros cometidos    : ",terros);
                fprintf(pr,"\n%s%3d",  "Acertos registrados: ",tacert);
                fprintf(pr,"\n%s%3d",  "Total Ajuda solict.: ",tajuda);
                fprintf(pr,"\n%s%3d%s","Tempo de simulacao : ",tmp," seg.");
                fprintf(pr,"\n\n%s","Grau de interesse..: ");
                fprintf(pr,"\n%s","Dialogo   :");
                for(cont=0;cont<total;cont++){
                    fprintf(pr,"%4d",cont+1);}
                fprintf(pr,"%s","\nFuncao    :");
                for(cont=0;cont<total;cont++){
                    fprintf(pr,"%c%03d",' ',vet[cont].func());}
                fprintf(pr,"%s","\nCores (FBC):");
                for(cont=0;cont<total;cont++){
                    textcolor(vet[cont].corf()); fprintf(pr," %s","_");
                    textcolor(vet[cont].corb()); fprintf(pr,"%s","_");
                    textcolor(vet[cont].corc()); fprintf(pr,"%s","_");}
                fprintf(pr,"%s","\nAcertos   :");
                for(cont=0;cont<total;cont++){
                    fprintf(pr," %03d",vet[cont].macert());}
                fprintf(pr,"%s","\nErros     :");
                for(cont=0;cont<total;cont++){
                    fprintf(pr," %03d",vet[cont].merros());}
                fprintf(pr,"%s","\nPed. Ajuda :");
                for(cont=0;cont<total;cont++){
```

```
                        fprintf(pr," %03d",vet[cont].majuda());}
                fprintf(pr,"%s","\nTempo (seg):");
                for(cont=0;cont<total;cont++){
                        fprintf(pr," %03d",vet[cont].tmp());}
                fprintf(pr,"%s","\nPorcent (%):");
                for(cont=0;cont<total;cont++){
                        fprintf(pr," %03d",tmp?vet[cont].tmp()*100/tmp:0);}
                fprintf(pr,"\n\n\n\n\n");
            fclose(pr); break;}}
        gotoxy(10,24); printf("%s"," <Espaco> Termina.                    ");
        while(!kbhit());}

// rotinas para salvacao e recuperacao de prototipos do agile 2

void salvar(janela vet[],int total){
        if(!total) return;
        FILE *fp; char *ss; int cont; _setcursortype(_NORMALCURSOR);
        gotoxy(10,24); printf(" Entre com o nome do prototipo:      ");
        gotoxy(42,24); ss=readstr(12); _setcursortype(_NOCURSOR);
        if((fp=fopen(ss,"wb"))==NULL){ gotoxy(10,24);
                printf("%s", " Nao posso abrir arquivo..    ");
                getch(); return;}
        for(cont=0;cont<total;cont++)
                fwrite(&vet[cont],sizeof(vet[cont]),1,fp);
        fclose(fp);}


int ler(janela vet[]){
        FILE *fp; char *ss; int cont=0; _setcursortype(_NORMALCURSOR);
        gotoxy(10,24); printf(" Entre com o nome do prototipo:      ");
        gotoxy(42,24); ss=readstr(12); _setcursortype(_NOCURSOR);
        if((fp=fopen(ss,"rb"))==NULL){ gotoxy(10,24);
                printf("%s", " Nao posso abrir arquivo..    ");
                getch(); return 0;}
        while(fread(&vet[cont],sizeof(janela),1,fp))cont++;
        fclose(fp); return cont;}

void main(){
        int total=0,mod=1,amb=1,posic=1,f,jan=0,erros;
        janela vet[10]; char op; int hm[4]={3,3,2,2};
        char *m[5][5]={{" Salvar   ",
                       " Ler      ",
                       " Sair     "},{
                       " Prototipar ",
                       " Simular  ",
                       " Avaliar  ",},{
                       " Limpo    ",
                       " Saturado "},{
                       " Texto    ",
                       " Grafico "}};
        char *vop[4]={" Arquivos "," Testes  "," Ambiente "," Modo    "};
        clrscr(); while (1){
                _setcursortype(_NOCURSOR); mud_amb(mod,amb);
                textcolor(BLACK); textbackground(GREEN);
                gotoxy(posic*11-8,1); cprintf("%s",vop[posic-1]);
                gotoxy(10,24); printf(" Selecione a opcao.");
```

```
if(jan){
        op=menu(m[posic-1],posic*11-8,3,hm[posic-1],1);
        if(posic==1){
                switch(op){
                        case 1: salvar(vet,total); break;
                        case 2: total=ler(vet); break;
                        case 3: _setcursortype(_NORMALCURSOR);
                        textbackground(BLACK); textcolor(LIGHTGRAY);
                        clrscr(); puts("_ Projeto Agile.\n"); exit(1);
        }} else if(posic==2){
                switch(op){
                        case 1: total=prototipar(vet,amb); break;
                        case 2: if(total)simular(vet,total,amb); break;
                        case 3: avaliar(vet,total); break;
        }} else if(posic==3){
                switch(op){
                        case 1: amb=0; break;
                        case 2: amb=1; break;
        }} else if(posic==4){
                switch(op){
                        case 1: if(mod==2){closegraph();mud_amb(1,1);}; break;
                        case 2: /*if(mod==1){opengraph();mud_amb(2,1);};*/ break;
}}} else op=getch(); switch(op){
        case 75: if (posic>1) posic--; break;
        case 77: if (posic<4) posic++; break;
        case 13: if (jan==0) jan=1; break;
        case 27: if (jan==1) jan=0; break;}}}
```

APÊNDICE C: DESCRIÇÃO DAS ESTRUTURAS DE DADOS DO AGILE

A ferramenta FAI, nesta versão, utiliza as estruturas de dados do AGILE [PROCÓPIO 92] e [SANTOS 92], compartilhando área comum de memória. As estruturas de dados utilizados na FAI são:

1 - Pilha de diálogos

Para que sejam apresentadas as ações de um diálogo que definem o seu sequenciamento, torna-se necessário conhecer a estrutura de dados da pilha de diálogos (figura 1), que auxilia na execução dessas ações.

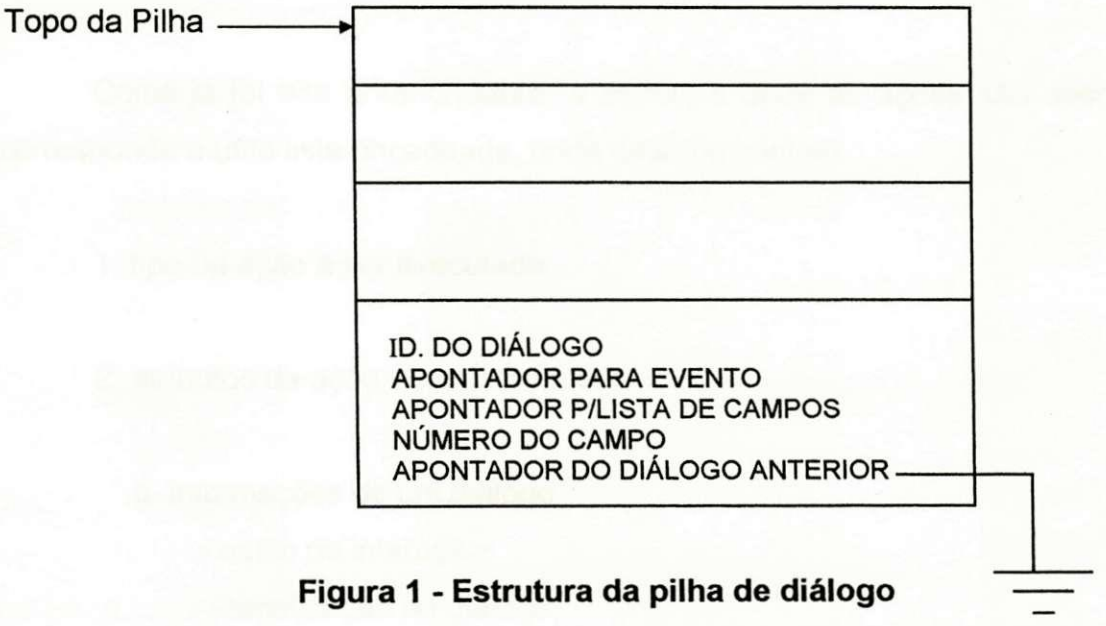


Figura 1 - Estrutura da pilha de diálogo

Esta estrutura tem como objetivo guardar o caminho percorrido para se chegar ao diálogo que se encontra ativo naquele momento. Dessa forma, pode-se reativar diálogos ativados anteriormente.

Cada elemento da pilha corresponde a um diálogo, e contém as seguintes informações:

- identificação do diálogo;
- apontador para a estrutura EVENTO, que foi alocada quando da ativação do diálogo;
- Número do campo do formulário que estava sendo preenchido quando este diálogo era o ativo. Esta informação só é preenchida para o diálogo de estilo formulário.
- Apontador para uma lista, onde cada nó contém o dado digitado pelo usuário final, para cada campo do formulário que está sendo sequenciado.

A seguir apresentam-se as operações desta pilha, juntamente com a descrição das ações.

2 - Descrição das ações

Como já foi dito anteriormente, a estrutura onde as ações são descritas corresponde a uma lista encadeada, onde cada nó contém:

1. tipo da ação a ser executada;

2. atributos da ação, que podem ser:

   a. informações de um diálogo:
      - estilo de interação;
      - identificação do diálogo;
      - número do campo do formulário, caso o estilo de interação seja formulário.

   b. informações de uma rotina da aplicação: número da rotina, para a estrutura RETORNOS.

c. valor inteiro.

Na figura 2, pode-se observar a estrutura correspondente a uma ação.

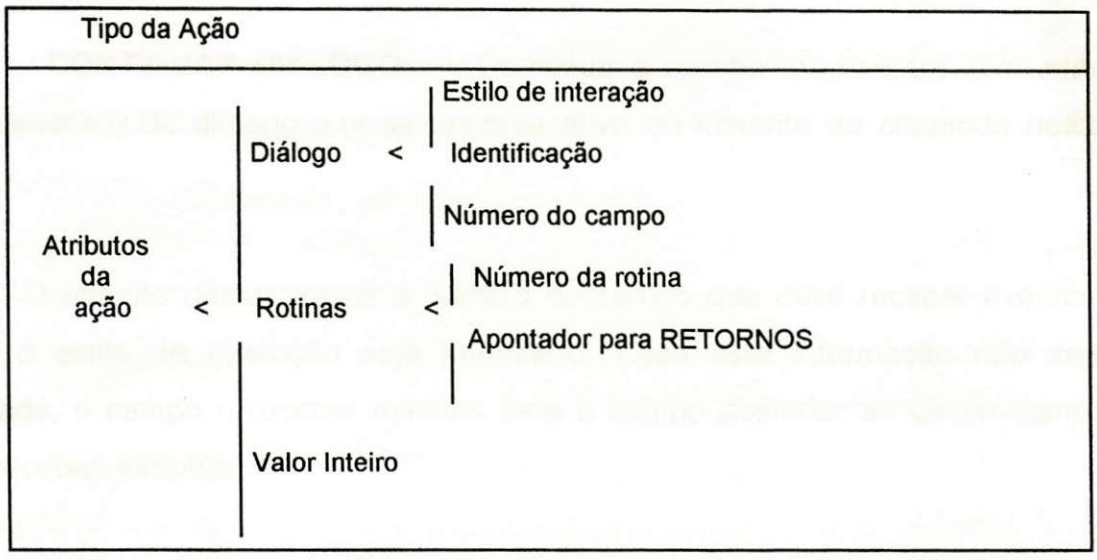


Figura 2 - Estrutura de uma ação

- **ATIVAR DIÁLOGO :>** Tornar ativo um novo diálogo, ou seja, associar os eventos (entradas do usuário) com o diálogo ativado.

Os atributos desta ação são:

. Estilo de interação: Menu, comando, pergunta-e-resposta e formulário;

. Identificação do diálogo;

. Número do campo que deve receber eventos, caso o estilo de interação seja formulário. Caso este dado não seja fornecido, o campo considerado será o primeiro campo do formulário.

- **VOLTAR DIÁLOGO: >** Ativar um diálogo que já foi ativado anteriormente.

Os atributos desta ação são:

. Estilo de interação;

. Identificação do diálogo.

. Número do campo que deve receber eventos, caso o estilo de interação seja formulário. Caso este campo não seja fornecido, os eventos serão gerados para o campo posterior ao último campo preenchido.

- CONTINUAR DIÁLOGO: > Continuar a receber os eventos (entradas dos usuários) do diálogo que se enconta ativo no instante da chamada desta ação.

O atributo desta ação é o número do campo que deve receber eventos, caso o estilo de interação seja formulário. Caso esta informação não seja passada, o campo a receber eventos será o campo posterior ao último campo que recebeu eventos.

- CONTINUAR A LER DIÁLOGO: > Terminar de digitar um entrada de um diálogo que, por algum evento, foi interrompido.

Os atributos desta ação são:

. Estilo de interação;

. Identificação do diálogo;

. Número do campo que deve receber eventos, caso o estilo de interação seja formulário. Se esta informação não foi definida, o campo a receber eventos será o último que os tinha recebido anteriormente.

- ENCERRAR AGILE: > Encerra a interação com AGILE e volta ao Sistema Operacional.

- ENCERRAR SIMULAÇÃO: > Encerra a simulação que está sendo realizada, retornando o controle para o AGILE.

- ENCERRAR DIÁLOGO: > Fecha o diálogo que se encontra ativo.

- APRESENTAR MENSAGEM: > Apresenta em determinada posição da tela uma mensagem.

- RETIRAR MENSAGEM: > Retira da tela uma mensagem.

- SUSPENDER EXECUÇÃO: > Suspende a simulação da interface por um intervalo de tempo.

O atributo desta ação é valor inteiro, informando o tempo, em segundos, que a simulação deve ser suspensa.

A estrutura **RETORNO** é uma lista encadeada, onde cada nó contém os possíveis valores retornados por uma função da aplicação, associando, a cada um dos valores, as ações que devem ser ativadas quando um deles for retornado.

A figura 3 apresenta esta estrutura.

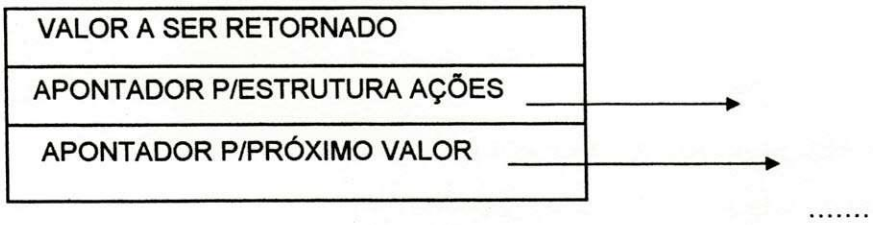


Figura 3 - Estrutura RETORNOS