UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SGC - UM AMBIENTE PARA A AUTOMAÇÃO DE PROCEDIMENTOS DE CARACTERIZAÇÃO E TESTE.

por

GILSON INÁCIO WIRTH.

Dissertação submetida como requisito parcial
para a obtenção do grau de
mestre em ciência da computação.

Prof. Sergio Bampi.
Orientador

Porto Alegre, dezembro de 1994.

CIP -CATALOGAÇÃO NA PUBLICAÇÃO

Wirth, Gilson Inácio

SGC - Um ambiente para a automação de procedimentos de caracterização e teste / Gilson Inácio Wirth.- Porto Alegre: CPGCC da UFRGS, 1994.

132p.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1989. Orientador: Bampi, Sergio.

Dissertação: Sistemas de Informação e Medida, Teste Auxiliado por Computador, Automação de Medidas, Circuitos Integrados.

AGRADECIMENTOS

Agradeço a todos aqueles que tornaram este trabalho possível: ao professor Sergio Bampi, pela orientação, apoio e amizade inestimáveis. A Ricardo Wartchow e James Petterson, bolsistas de Iniciação Tecnológica e Industrial - ITI, pela dedicação e brilhantismo demonstrados durante o trabalho de implementação do ambiente SGC, bolsistas que jamais arrefeceram, apesar dos constantes contratempos ocorridos no pagamento de suas bolsas.

Agradeço aos colegas e professores deste curso de pós-graduação, em especial àqueles que formam o grupo de microeletrônica, pela amizade e comentários realizados, que em muito auxiliaram na definição do rumo a ser seguido no decorrer deste trabalho. Dentre estes é indispensável destacar o colega Rosalvo E. Streit, pela colaboração durante o trabalho de definição da base de dados do ambiente SGC, o bolsista ITI Christian Burger, pelos trabalhos realizados durante o projeto de uma placa GPIB para o controle dos motores de passo do micromanipulador MCS 8806, ao colega Luis Fernando Ferreira, pela grande colaboração na fase inicial deste trabalho, e ao Luis Felipe Uebel, colega em muitos trabalhos e companheiro na luta contra editores de texto e equipamentos (como impressoras), que insistiam em não funcionar como esperávamos.

Foi também muito valiosa a colaboração dos professores que constituíram a banca do seminário de andamento deste trabalho, Prof. Sergio Bampi, Prof. Flávio R. Wagner, Prof. Tiaraju V. Wagner e Prof. Ricardo A. L. Reis, ao realizarem comentários fundamentais para definição dos trabalhos de finalização necessários, e ao transmitir o apoio e estímulo necessários à conclusão desta dissertação.

É necessário agradecer também a José Otávio Simões, do LAC - Laboratório Central de Eletrotécnica e Eletrônica (COPEL/UFPR), pelo apoio prestado durante a realização deste trabalho, e pela oportunidade proporcionada na introdução do ambiente SGC junto ao LAC.

Agradeço também ao CNPq e ao RHAE, pelo auxílio financeiro, e aos funcionários do Instituto de Informática, em especial à Eliane Iranco e ao Luis Otávio Soares, sempre dispostos a auxiliar na solução dos problemas organizacionais que surgiram, como o limitado horário de acesso ao laboratório de microeletrônica.

Finalmente agradeço, mas de modo algum por último, aos meus pais, irmãos, amigos e familiares, pelo apoio sempre certo e espontâneo.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Este trabalho trata de ambientes de *software* para a realização de teste e caracterização de dispositivos, componentes ou circuitos eletro-eletrônicos, de forma automatizada. Ênfase especial é dada à problemática relacionada ao teste e caracterização automatizados de dispositivos e circuitos integrados.

O assunto é tratado sob dois pontos de vista distintos e complementares:

i) Sob o ponto de vista do engenheiro de teste e caracterização, que realiza experimentos físicos, que são as medidas e aquisições de dados, processa, visualiza e analisa dados.

ii) Sob o ponto de vista do projetista de ferramentas de *software*, que desenvolve programas de computador para automatizar as tarefas rotineiramente realizadas durante o teste e a caracterização.

Após a análise do assunto em questão, um ambiente de *software* (*Framework*), chamado SGC, é proposto e implementado. O SGC foi implementado em ambiente *MS-Windows*™ através de um paradigma de orientação a objetos, e pretende atender as necessidades inerentes ao teste e caracterização automatizados, quando tratados sob os dois pontos de vista citados.

O ambiente SGC é um sistema aberto, a fim de permitir o fácil acoplamento de novas facilidades, bem como mostra-se um sistema prático para suportar rotinas de teste e caracterização em laboratório.

PALAVRAS-CHAVE: Teste Auxiliado por Computador, Caracterização Auxiliada por Computador, Automação de Medidas, Ambientes de Teste e Caracterização, Circuitos Integrados, Sistemas de Informação e Medida, Sistemas Abertos.

ABSTRACT

This work deals with software environments for automatic test and characterization of electro-electronical devices, components and circuits. Special attention is paid to the features of testing and characterizing integrated devices and circuits.

The subject is treated in two different and complementary views:

i) The needs of the test and characterization engineer are addressed. The test engineer carries out physical experiments, which embody measurements and data acquisitions, data processing, visualization and analysis.

ii) The needs of the software tools developer, who develops computer programs for the automation of the procedures that are usually carried out during test and characterization, are also addressed.

After the analysis of the subject under study, a software framework, called SGC ("Sistema de Gerenciamento e Controle"), is proposed and implemented. The SGC Framework was implemented under *MS-Windows*™ using a object oriented approach. The SGC framework aims to fulfill the needs inherent to the automatic test and characterization, when treated using the approaches mentioned above.

The SGC Framework is a open system, supporting the easy integration of new software functions to the environment, as well as a practical system for test and characterization laboratory routines.

KEY WORDS: Computer Aided Test (CAT), Computer Aided Characterization, Automated Measurement, Test and Characterization Frameworks, Integrated Circutis, Measurement Information Systems, Open Systems.

1 INTRODUÇÃO.

1.1 Motivação.

Atualmente, um dos fatores importantes para a competitividade industrial é a capacidade de assegurar-se a qualidade dos produtos oferecidos. Para que se possa assegurar a qualidade da produção, modernas técnicas de controle de qualidade são empregadas. Todas estas técnicas necessitam que medidas rápidas e precisas sejam realizadas para testar e/ou caracterizar os produtos de uma linha de produção e/ou protótipos. Na Microeletrônica não é diferente.

É necessário que se disponha de um ambiente para automatizar os procedimentos de caracterização e teste dos dispositivos integrados, para que se possa realizar os procedimentos necessários a assegurar a qualidade e confiabilidade dos dispositivos produzidos economicamente. Este ambiente pode ser chamado de "sistema de informação e medida" (SIM) /MUR93/. Um SIM é classificado como caso de particular de Sistema de Informação porque neste existe a necessidade de realizar-se um experimento físico, que é a medida. Outros chamam estes sistemas que permitem a realização de teste e caracterização de maneira automatizada de "sistemas de teste auxiliado por computador" (CAT Systems - Computer Aided Test Systems) /MIL87/.

Para automatizar a caracterização e teste dos dispositivos integrados várias ferramentas são necessárias: ferramentas para controlar os equipamentos de teste e realizar a aquisição de dados, para visualizar e analisar os dados adquiridos, para realizar a extração de parâmetros, para gerar vetores de teste, e assim por diante. Visando incrementar a produtividade é indispensável que estas ferramentas estejam integradas no mesmo ambiente de engenharia.

Atualmente, na maioria dos ambientes de teste/caracterização de dispositivos, estas ferramentas constituem-se em "ilhas de automação". Geralmente não existe uma maneira padrão de tratar os dados com os quais estas ferramentas trabalham e os resultados por ela gerados como fazendo parte de um todo, porque faltam funções como a intercomunicação entre ferramentas, gerenciamento dos dados em uma base de dados comum, entre outras características importantes. Via de regra, cada equipamento de teste utilizado incorpora um *software* de CAT próprio, específico, sem interfaces com outros sistemas de CAT.

Quando não há intercomunicação entre estas ferramentas, tempo precioso é perdido fazendo-se, manualmente, com que os dados fluam de uma ferramenta para outra. Quando não há como repetir automaticamente os procedimentos de caracterização e teste, igualmente tempo é perdido repetindo-se manualmente estes procedimentos. Quando funções comuns a todas as ferramentas, como tratamento de erros, interfaces com usuário e *hardware* não são providas, mais tempo é perdido, desenvolvendo-se estas funções para cada uma das ferramentas, e quando alterações em alguma destas funções comuns são necessárias, as mesmas devem ser feitas em cada uma das ferramentas.

O ambiente SGC, objeto desta dissertação, foi projetado para ser um ambiente eficiente, auxiliando o projetista de *software* a desenvolver ferramentas que poderão ser facilmente integradas ao sistema, e oferecendo ao engenheiro de testes um ambiente no qual estas poderão ser eficientemente usadas em procedimentos de rotina.

1.2 Terminologia Utilizada.

No decorrer deste trabalho, vários termos serão utilizados com um significado particular e bem definido, formando uma terminologia própria à área de caracterização e teste elétrico automáticos. Para que o trabalho realizado e as idéias aqui propostas possam ser apresentados de maneira clara e concisa, descreveremos os principais termos desta terminologia, exaustivamente empregados nos demais capítulos desta dissertação.

Em um processo de teste ou caracterização, vários agentes tomam parte: pessoas, instrumentos de medida, e ferramentas de computação. Chamaremos de **procedimento** a descrição da seqüência de ações recíprocas entre estes agentes, com a finalidade de obter informações a respeito de quantidades inerentes ao objeto sob teste ou objeto de caracterização.

Vamos tomar como exemplo a caracterização elétrica de osciladores formados por um anel de inversores, disponíveis em uma pastilha de teste não encapsulada. Suponha-se que desejamos caracterizar estes osciladores em anel para vários valores da tensão de alimentação (VCC). Durante esta caracterização, vários ações devem ocorrer, a saber:

1. Primeiramente, deve-se posicionar as micro ponteiras sobre os PADs apropriados, para que possamos contactar os terminais dos osciladores em anel.

2. Ajuste das escalas de um multímetro e/ou osciloscópio, para que posteriormente possamos adquirir os dados desejados. Provavelmente optar-se-á por utilizar a opção de auto-escala, se estes a oferecerem.

3. Conexão à alimentação elétrica do circuito através de uma fonte de tensão programável, com o valor de tensão de alimentação desejado.

4. Deve-se utilizar o osciloscópio para realizar a leitura de dados como freqüência e período de oscilação, tensão pico a pico e rms, tempos de subida e descida, entre outros.

5. Leitura da corrente consumida pelo circuito através de um multímetro.

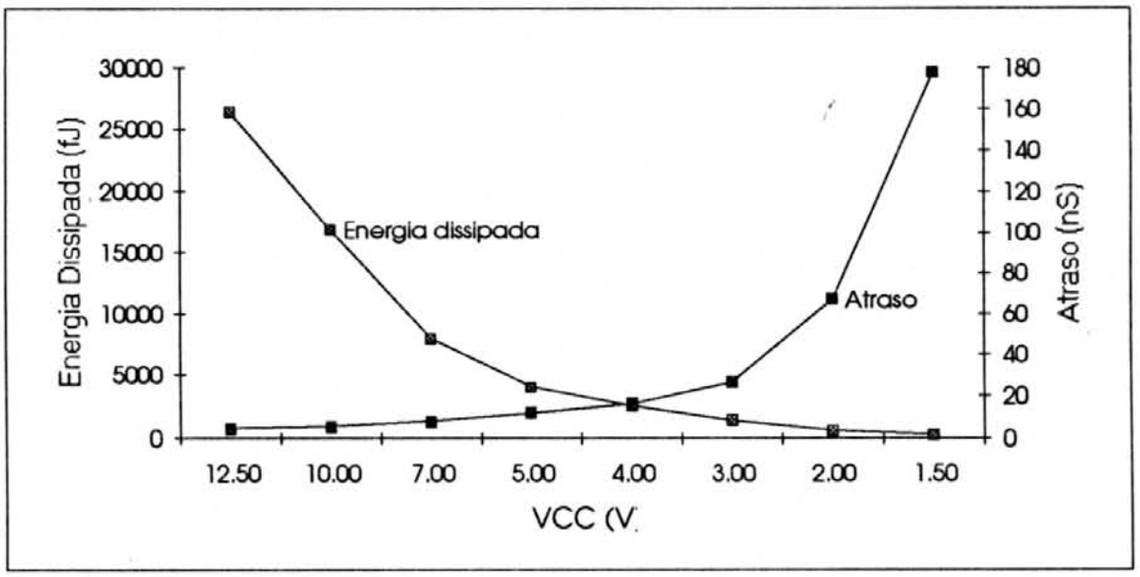


Fig. 1.1 Energia dissipada por ciclo de oscilação e atraso do sinal, em função da tensão de alimentação (VCC).

6. Quando deseja-se visualizar os dados que estão sendo adquiridos, deve-se por exemplo, exibir na tela do computador os dados que estão sendo lidos, sob a forma de uma curva corrente *versus* tensão (IxV).

7. Se é desejada a visualização da curva da energia dissipada por ciclo de oscilação e do atraso do sinal, ambos *versus* tensão de alimentação (VCC), deve-se multiplicar o valor da tensão aplicada pela fonte de tensão pelo valor da corrente lida pelo multímetro, multiplicando-se então o valor obtido pelo valor do período de oscilação medido pelo osciloscópio.

8. Plota-se o valor obtido no passo anterior em uma curva, onde a coordenada *x* é o valor da tensão aplicada, cuja coordenada *y1* é o de energia dissipada calculado no passo anterior e cuja coordenada *y2* é o valor do período de oscilação medido pelo osciloscópio, como exemplificado na figura 1.1.

9. Deve-se verificar se todo o espectro de valores de tensão de alimentação para os quais deseja-se caracterizar o oscilador em anel já foi coberto. Caso negativo, deve-se definir o novo valor da tensão de alimentação e retornar ao passo 3.

Esta seqüência de ações pode ser denominada de procedimento.

Como exemplificado, cada procedimento pode ser subdividido em uma seqüência de ações ou eventos. Cada ação será chamada de **passo**. Estes por sua vez, podem ser classificados em *setups*, atividades, fluxo de dados, passos aritméticos, e passos condicionais.

Os ***setups***, como o próprio nome já diz, definem os diferentes ajustes das ferramentas durante o procedimento (como a opção de utilizar-se a escolha de escala automática, para o osciloscópio e multímetro).

As **atividades**, definem as tarefas a serem realizadas pelas ferramentas (como no exemplo anterior, a leitura da corrente consumida pelo oscilador em anel).

Os passos classificados como **fluxo de dados** são aqueles em que existe troca de dados entre as ferramentas utilizadas durante a execução destes. No exemplo anterior, é possível que a ferramenta que irá mostrar a curva IxV na tela do computador não é a mesma que faz a leitura do multímetro. Neste caso, é necessário que a ferramenta que faz a leitura do multímetro envie o valor lido à ferramenta de visualização da curva IxV na tela. Este passo do procedimento, de agora em diante, será chamado de transferência de dados.

Os **passos aritméticos** são aqueles em que realiza-se operações com os dados relativos a um procedimento, como, no exemplo acima, a multiplicação da tensão de alimentação pelo valor medido para a corrente consumida e pelo valor medido para o período de oscilação do circuito.

Os **passos condicionais** são aqueles em que decisões devem ser tomadas. Estas decisões podem vir a alterar a seqüência em que um procedimento é executado, como no exemplo anterior, quando verifica-se se todo o espectro de tensões de alimentação sob as quais o circuito deve ser caracterizado já foi varrido, definindo-se então qual o próximo passo a ser executado.

Logo, um procedimento é formado por passos, que podem ser atividades, *setups*, fluxo de dados, passos aritméticos ou passos condicionais.

Para algumas ferramentas, pode ser possível definir-se passos dos diversos tipos existentes. Por exemplo, para o osciloscópio pode-se definir as escalas (Volts por divisão no eixo Y, tempo por divisão no eixo X) e o acoplamento (AC ou DC), portanto, é possível definir-se um *setup* do osciloscópio. Também pode-se definir a aquisição da forma de onda, a medida da freqüência de oscilação do sinal, entre outros, portanto, também podemos definir uma atividade a ser realizada pelo osciloscópio. Além disso, pode-se desejar que a forma de onda adquirida pelo osciloscópio seja transferida para um programa que mostre-a na tela, definindo-se então um passo fluxo de dados para o osciloscópio.

Durante a caracterização e/ou teste de dispositivos integrados, geralmente deseja-se executar o mesmo procedimento sobre várias amostras. Cada execução do procedimento será chamada de **execução**. Ou seja, uma execução é uma "rodada" de um procedimento, quando todos os passos que o formam são realizados pelas ferramentas correspondentes.

Quando uma atividade é executada, ela pode gerar dados. Os dados gerados pela execução das atividades de um procedimento serão chamados de ***ActData***. No nosso exemplo, a freqüência de oscilação do oscilador em anel, medida com o osciloscópio, é uma *ActData*.

1.3 Objetivos.

O objetivo principal deste trabalho foi o desenvolvimento de um ambiente de *software* para a automação dos procedimentos de caracterização e teste, executados rotineiramente em laboratórios de instituições de pesquisa e ensino, bem como por indústrias, incluindo-se neste ambiente o suporte para o desenvolvimento e integração de novos módulos de *software*, da maneira mais genérica e flexível possível.

Tendo-se este objetivo principal em mente, pode-se analisar as características de um sistema desta natureza. Estas características podem ser divididas em dois grandes grupos: as que dizem respeito às necessidades do engenheiro de teste/caracterização e as que dizem respeito ao projetista de ferramentas para a automação dos procedimentos de teste/caracterização.

1.3.1 A automação vista pelo engenheiro de caracterização e teste.

O engenheiro de caracterização e teste deve ser assistido ao desempenhar as suas tarefas. Necessidades de automação específicas podem ser enumeradas, de maneira que o SGC deve prover:

- Definição de procedimentos de caracterização/teste: O SGC deve prover meios de o usuário definir novos procedimentos de teste/caracterização ou modificar procedimentos já existentes de maneira fácil e rápida, através de uma interface amigável, que não exija um treinamento demorado para a sua utilização.

- Documentação dos procedimentos e das execuções: Todos os dados que caracterizam uma procedimento ou uma execução devem ser armazenados e estar disponíveis para consulta posterior. Assim, é necessário que sejam armazenadas informações como as condições em que os passos foram executados. Além de informações como o nome da pessoa que definiu o procedimento, da pessoa que o modificou, bem como as respectivas datas, comentários que usuário deseje fazer, entre outros.

- Repetição dos procedimentos: este é um aspecto que diferencia o SGC dos ambientes de CAD/CAE convencionais. Nestes não nos interessa repetir todos os passos que levaram ao produto final, porque obtém-se sempre o mesmo resultado. Muitas vezes entretanto, os procedimentos de teste/caracterização são executados sobre amostras diferentes, podendo portanto levar a resultados diferentes. Nos procedimentos de caracterização, é importante realizá-los sobre um grande número de amostras, para garantir que os parâmetros obtidos sejam representativos da tecnologia caracterizada. Nos procedimentos de teste, o ideal seria que estes fossem repetidos sobre todas as amostras, antes que estas fossem colocadas no mercado. Assim, é vital que o engenheiro de teste/caracterização possa repetir as procedimentos fácil e rapidamente.

- Análise dos resultados de maneira rápida e simples: Facilidades devem ser oferecidas para que a análise dos dados gerados durante a execução de um procedimento seja realizada de maneira adequada.

- Geração de uma biblioteca de procedimentos de caracterização/teste: Os procedimentos definidos devem estar organizados de maneira a formarem um biblioteca de procedimentos, permitindo a sua rápida localização, identificação e acesso. Além disso, deve ser possível utilizar conjuntos de Passos definidos previamente quando da edição de novos procedimentos.

- Supervisão da execução dos procedimentos: Deve ser possível a supervisão dos procedimentos, através da monitorização dos passos em execução em um determinado momento e dos dados que estão sendo gerados/adquiridos. Além disso, também deve ser possível suspender ou abortar a execução do procedimento.

- Operação inter-ferramentas: Deve ser possível realizar procedimentos utilizando-se múltiplas ferramentas. Uma maneira simples de realizar-se a transferência de dados entre ferramentas, garantindo-se a adequação dos dados à ferramenta, deve ser provida. Um meio simples de realizar-se a escolha da ferramenta que deve realizar o passo, e a especificação do passo, deve ser provida. Isto deve ser válido também para os procedimentos nos quais se utilize ferramentas diferentes, desenvolvidas por pessoas diferentes, em centros de pesquisa diferentes.

- Fácil integração de ferramentas: Deve-se prover um meio para que o usuário possa facilmente integrar novas ferramentas ao sistema. Todos os procedimentos previamente definidos devem continuar válidos. Não devem ser necessárias modificações nas ferramentas já integradas ao sistema.

- Fácil substituição de ferramentas: Deve ser possível substituir as ferramentas facilmente, por exemplo, quando novas versões de ferramentas são implementadas, ou quando deseja-se substituí-las por outras, que realizam tarefas semelhantes. Esta substituição de ferramentas não deve requerer alterações nos Procedimentos já definidos. Estes Procedimentos devem continuar válidos (passíveis de repetição, usando as ferramentas que substituíram as anteriores). Além disto, não devem ser necessárias modificações em nenhuma ferramenta integrada ao SGC, quando da substituição de outras ferramentas.

- Acesso à base de dados de maneira fácil, coerente e consistente: Os dados devem ser armazenados de maneira organizada, permitindo ao usuário um acesso fácil e rápido aos dados e das relações entre eles existentes. Deve-se prover mecanismos que não permitam ao usuário levar a base de dados à um estado inconsistente.

- Interface com o usuário padronizada e amigável: Todas as ferramentas devem interagir com o usuário de uma maneira padrão, evitando que o usuário tenha que reaprender a metodologia de interação com a ferramenta a cada nova ferramenta utilizada. A interface para análise e visualização dos dados gerados/adquiridos durante a execução de um procedimento deve ser amigável.

- Independência de tecnologia: O sistema não deve impor limitações no universo de aplicações, dentro do possível. Ele deve ser genérico a ponto de permitir sua

utilização em todas as áreas que necessitam realizar a aquisição de dados (medidas) e posteriormente processá-los. Como requisito mínimo, ele deve ser apropriado para o teste/caracterização das mais diversas tecnologias utilizadas na microeletrônica (CMOS, Bipolar, SOI, InGaAsP, etc.).

1.3.2 A automação vista pelo projetista de ferramentas de *software*.

O SGC também deve auxiliar o projetista de *software* na tarefa de desenvolver ferramentas para a automação dos procedimentos de caracterização e teste. Necessidades específicas podem ser enumeradas. O SGC deverá prover as funções comuns a diversas ferramentas:

- Definição de um modelo para a base de dados: Um modelo para o armazenamento dos dados na base de dados deve ser provido. Estes modelo deve permitir que: i) as ferramentas compartilhem os dados armazenados; ii) implemente-se a troca de dados entre ferramentas; iii) implemente-se a documentação dos procedimentos.

- Funções de acesso a Base de Dados: Para garantir-se a integridade da base de dados e esconder do projetista de ferramentas a complexidade e os detalhes da implementação da base de dados, devem ser providas funções que permitirão às ferramentas o acesso a base de dados, sem que estas necessitem acessar diretamente os arquivos que formam a base de dados. Além disto, desta maneira será possível realizar modificações na implementação da base de dados (como extensões no seu modelamento) sem que sejam necessárias modificações nas ferramentas.

- Funções de acesso às interfaces de *hardware* existentes: As ferramentas integradas não devem acessar as facilidades de *hardware* disponíveis diretamente. Este acesso deve ser implementado através de *drivers* específicos. O interfaceamento deve ser realizado desta forma a fim de simplificar-se a escrita e a manutenção das ferramentas de *software*. Além de garantir-se que eventuais problemas serão tratados, e se for o caso, reportados ao usuário de maneira uniforme. Como exemplos de drivers que devem ser inicialmente implementados podemos citar os drivers para as interfaces GPIB e RS232.

- Intercomunicação entre ferramentas: O SGC deve prover funções de intercomunicação entre ferramentas. Estas funções permitirão a troca de dados e sincronização das ferramentas quando da execução de procedimentos com múltiplas ferramentas, inclusive muitos equipamentos de teste. Estas funções devem esconder das ferramentas os detalhes e a complexidade da implementação da troca de dados entre as mesmas.

- Modularidade: O sistema deve ser modular, a fim de permitir uma fácil manutenção.

- Interface com o Usuário: Deve haver a definição de uma maneira padrão para a interação com o usuário, e funções de uso comum (como help-on-line, barras para mensagens de status, etc.) devem ser providas, como módulos de *software* de uso comum a todas as ferramentas.

1.4 Escolha da Plataforma de *Hardware* e Ambiente de *Software*.

Todo o ambiente SGC foi desenvolvido para plataforma IBM-PC com sistema operacional Microsoft Windows.

A opção pela plataforma IBM-PC foi realizada em virtude de esta plataforma ser usualmente encontrada em laboratórios de medida de todas as instituições de ensino/pesquisa com as quais interagiu-se no decorrer desta dissertação e estar a mesma amplamente difundida na indústria nacional. Além disto, acredita-se que esta plataforma continuará disponível no mercado nacional e internacional por muitos anos ainda.

A opção pelo sistema operacional Microsoft Windows foi feita porque este oferece as facilidades básicas indispensáveis para que se possa atingir os objetivos descritos acima, como a implementação de uma interface com o usuário uniforme e amigável e a troca de mensagem entre ferramentas.

A linguagem de programação escolhida foi o C++, devido a sua flexibilidade e também porque o paradigma orientado a objetos ajuda a garantir a modularidade do sistema, além do C ser uma linguagem de programação bem difundida nos meios acadêmicos e industriais.

1.5 Organização do texto da dissertação.

Nos seguintes capítulos, estão descritos: i) o trabalho teórico desenvolvido durante a fase de análise da problemática relacionada à automação dos procedimentos de caracterização e teste, e desenvolvimento de ferramentas para tal fim; ii) a implementação do ambiente SGC; iii) as facilidades oferecidas aos engenheiros de caracterização e teste, para a realização destes procedimentos de caracterização e teste; iv) as facilidades oferecidas ao projetista de *software* para a automação de teste e

caracterização e, finalmente, v) os trabalhos que pretendemos desenvolver no futuro, estendendo as potencialidades do ambiente SGC, são descritos.

No capítulo 2 as alternativas até o momento propostas na literatura são analisadas e discutidas, apontando-se suas lacunas, deficiências e acertos. Esta discussão servirá de ponto de partida para a definição da estrutura global do ambiente SGC, que é proposta no capítulo 3. Depois de definidos os módulos que devem estar presentes nesta estruturação do ambiente SGC, passamos à discussão detalhada de cada um deles. No capítulo 4 discute-se o módulo de gerenciamento dos procedimentos, com os seus diversos componentes, como a linguagem implementada a fim de permitir ao usuário a edição dos procedimentos de caracterização e teste, e as facilidades que permitem o gerenciamento destes procedimentos pelo usuário. No capítulo 5 discute-se o módulo que permite a intercomunicação (troca de dados) entre os diversos componentes do ambiente SGC. Intercomunicação que permite o acesso à base de dados, às interfaces de *hardware*, e a troca de dados entre as ferramentas integradas. No capítulo 6 as características que a base de dados deve possuir são analisadas, passando-se em seguida ao modelamento e à subseqüente descrição da sua implementação. No capítulo 7 as linhas que devem ser observadas ao planejar-se e implementar-se ferramentas e *drivers* a serem integrados ao ambiente SGC são discutidas, além de apresentar-se as ferramentas e *drivers* já integrados. No capítulo 8 apresenta-se as facilidades oferecidas para a integração de novas ferramentas e *drivers*, com destaque especial para os esqueletos de ferramenta e *driver* disponíveis. No capítulo 9 os trabalhos que serão realizados em um futuro próximo são analisados, pois representam uma evolução natural do ambiente SGC. Finalmente, no capítulo 10, são apresentadas as conclusões às quais chegou-se durante o desenvolvimento deste trabalho.

2 AMBIENTES DE AUTOMAÇÃO: UMA REVISÃO.

2.1 Introdução.

Neste capítulo são analisados os Sistemas de Informação para Medidas (SIMs) já desenvolvidos e as alternativas já propostas na bibliografia disponível para a problemática da automação dos procedimentos de caracterização e teste.

Tendo em vista as necessidades descritas no capítulo 1, vários SIMs foram analisados /DOE84/, /RIN88/, /AUD89/, /FER90/, /FLU90/, /RUS90/, /SOE90/, /CAD92/, /DAP92/ e /MUR93/. Três destes (/MUR93/, /FER90/ e /DAP92/) serão aqui discutidos. Esta escolha deu-se porque estes formam o conjunto que melhor representa o universo estudado e atendem às necessidades apontadas no capítulo anterior em maior grau.

2.2 O sistema METIS.

O primeiro SIM analisado chama-se METIS e está descrito em /MUR93/. Este sistema foi idealizado a partir da análise de outros três sistemas, a saber: CALIBRAT /RIN88/, MET/CAL /FLU90/ e *System One* /AUD89/.

Para a escrita dos procedimentos, o METIS utiliza uma linguagem procedural, mas que em alguns pontos procura aproximar-se das linguagens declarativas. Pretende-se que o conjunto estável das combinações entre palavras (termos) normalmente utilizado quando definindo-se um procedimento esteja representado no *vocabulary of a metrologist* (VM) /MUR93/. O conjunto de variantes que formam o VM seria definido a partir de uma análise dos métodos mais amplamente aceitos e aplicados nos procedimentos de caracterização e teste. As variantes do VM formarão o conjunto de comandos (termos) disponíveis ao usuário para a definição do procedimento. O VM, juntamente com as regras que regulamentam a combinação (relação) entre estas variantes, formam a linguagem para a programação dos procedimentos. Esta linguagem é chamada de *Formalized Natural Language* (FNL) /MUR93/, e pretende ser uma espécie de linguagem natural limitada.

As ferramentas são integradas ao METIS ao criar-se uma biblioteca de ferramentas. A função destas ferramentas é realizar os procedimentos descritos a partir da FNL. Há um gerente encarregado de formar o procedimento a partir da descrição em FNL e executá-lo. Os parâmetros necessários são passados como argumentos quando da chamada pelo gerente dos demais módulos da biblioteca.

Em alguns casos, no momento da execução do procedimento, o gerente pode chamar rotinas existentes na biblioteca que armazena imagens gráficas dos painéis frontais de alguns instrumentos utilizados, a fim de permitir a entrada dos dados necessários a execução do procedimento, aproximando a FNL das linguagens visuais. É importante observar que estas imagens gráficas não são montadas pela ferramenta que manipulará os dados informados através delas adquiridos, e que a chamada a estas ferramentas deve ser programada de forma procedural. É o gerente que se encarrega de mostrar a imagem ao usuário, aguarda que o usuário informe os dados solicitados, em seguida repassando-os à ferramenta que deles necessita, no momento da execução do procedimento.

O METIS pode ser subdividido em três módulos principais:

- O *Meta System*, utilizado para configurar o sistema para o âmbito da aplicação em que será utilizado. Esta configuração consiste basicamente na definição do *vocabulary of a metrologist* (VM), que constitui-se da base léxica da FNL. No artigo em discussão não fica claro como isto é realizado. Se uma linguagem própria para esta configuração é implementada, ou se é necessário que se programe com linguagens procedurais convencionais (como C, por exemplo).

- O *System Level*, que oferece as facilidades para a edição e execução do procedimento. A edição do procedimento pode ser realizada utilizando-se um menu hierárquico, que permite a seleção dos comandos definidos no VM. A definição dos dados a serem armazenados também é realizada neste nível.

- O *Work Procedure Level*, que é invocado pelo gerente durante a (ou antes da) execução do procedimento, para que o usuário possa informar ao sistema os parâmetros necessários à execução deste. Para algumas ferramentas isto é realizado através de uma interface que imitam os painéis frontais dos equipamentos a serem utilizados.

2.3 O sistema PI.

Segundo o SIM definido em /FER90/, aqui chamado PI, quando um computador pessoal é empregado para processar dados, controlar equipamentos para a aquisição de dados/realização de medidas (incrementando as potencialidades destes), processar estes dados e mostrá-los ao usuário, além de assistir o usuário durante o

processo de teste e caracterização em uma maneira amigável, teremos um *Personal Instrument* (PI).

Os conceitos apresentados e os esforços despendidos tem como objetivo a obtenção de um PI.

O PI pode ser dividido nos seguintes módulos principais:

- Módulo de aquisição de sinais: Este módulo permite ao usuário o controle dos equipamentos utilizados para a aquisição de dados. Também é neste módulo que todos os parâmetros dos equipamentos que realizam a aquisição de sinais são definidos.

O acesso às facilidades oferecidas, neste e nos demais módulos, é orientado a menus.

As programações realizadas podem ser recuperadas e utilizadas em aquisições posteriores. A recuperação deve ser realizada manual e seqüencialmente. Os dados adquiridos são transferidos a disco.

- Módulo de visualização e elaboração: Este é o ponto mais forte do sistema PI. As funções deste módulo estão disponíveis, durante a execução de um procedimento, depois que a aquisição de dados estiver completa, podendo também serem utilizadas para processamento de dados adquiridos em procedimentos executados anteriormente.

Os dados são apresentados ao usuário de uma maneira semelhante ao display de formas de onda em uma tela de osciloscópio. São oferecidas facilidades para tratamento dos dados adquiridos, como: medida de período, valores de pico e RMS, diferença de fase entre sinais e espectro de freqüência do sinal.

- Módulo de processamento dos sinais: Realiza o processamento dos dados adquiridos/medidos permitindo a extração das informações desejadas.

- Módulo que permite "shell" para outros programas: Permite que o usuário retorne ao sistema operacional e execute outras ferramentas, através da execução de comandos, na linha de comandos do sistema operacional.

2.4 O Sistema AOB.

O Sistema descrito em /DAP92/, aqui chamado de AOB, é implementado a partir de uma paradigma orientado a objetos.

Neste sistema, objetos físicos (instrumentos, controladores de interface, dispositivos de teste, e o próprio procedimento de teste) são representados por objetos de *software* que comunicam-se entre si. O sistema de *software* é formado por uma coleção de objetos de *software*. A idéia básica é que os objetos de *software* sejam a imagem de objetos físicos. Pretende-se que o sistema para o gerenciamento dos procedimentos de teste seja construído através da escolha dos objetos de *software* apropriados, da conexão destes de acordo com as conexões existentes entre os objetos físicos que estes representam, e através da apropriada inicialização das estruturas de dados internas dos mesmos.

Assim, cada instrumento utilizado em um procedimento de teste é modelado por basicamente dois objetos de *software*: o primeiro modela as facilidades oferecidas pelo instrumento; o segundo modela a forma utilizada para controlar o respectivo objeto (por exemplo o tráfego de mensagens através de uma interface GPIB). Este segundo objeto de *software* é responsável por mapear comandos do primeiro objeto nas mensagens que devem trafegar na interface que controla o instrumento. Logo, para integrar novas facilidades ao sistema, novos objetos de *software* devem ser programados e acoplados ao sistema. Nenhum suporte é oferecido para a realização desta tarefa.

O procedimento de teste também é modelado por um objeto de *software*, chamado supervisor. Este objeto de *software* é responsável por inicializar os demais objetos e coordenar a execução do procedimento. Durante a inicialização destes objetos, eles solicitam ao usuário que os parâmetros necessários a execução do procedimento sejam informados. Estes parâmetros devem ser informados todas as vezes que um procedimento é executado (repetido).

Toda vez que o usuário desejar incluir um novo procedimento ao sistema, ele deverá programar um novo objeto de *software* supervisor que represente este procedimentos. Para modificar um procedimento, o código do objeto supervisor que o representa necessita ser alterado.

2.5 Deficiências encontradas nos SIM's analisados:

Os diferentes SIM's acima descritos podem ser analisados de acordo com os critérios (objetivos) estabelecidos no capítulo 1. Ao realizar-se esta análise, os seguintes fatos podem ser observados:

- Nenhum dos sistemas estudados oferece uma boa maneira de se realizar a definição dos procedimentos e repetir sua execução, sempre que necessário.

No METIS, há a necessidade do usuário aprender a sintaxe do VM e as regras que permitem a combinação dos seus termos. Além disto, como os termos do VM formam frases prontas, e devem ser um conjunto mínimo de termos cobrindo todos os possíveis problemas na área de aplicação /MUR93/, o VM estará sempre intimamente ligado à aplicação final, limitando assim a gama de procedimentos passíveis de implementação. Assim, quando desejar-se definir procedimentos para novas aplicações, é necessário realizar alterações na definição do VM, tarefa esta que pode vir a exigir a alterações no código fonte do sistema.

Já no sistema AOB, não é possível repetir o procedimento de teste com novas amostras, após ter-se encerrado o procedimento, porque na fase de inicialização dos objetos, o usuário é chamado a realizar a entrada de dados que não são armazenados de forma permanente, sendo assim perdidos quando o procedimento é encerrado. Esta é uma grande falha também a nível de documentação do procedimento, porque não é possível a recuperação de todas as condições em que o procedimento foi executado.

Outro inconveniente do sistema AOB é o fato de o procedimento ser modelado por um objeto de *software*, necessitando assim que o usuário proceda a programação deste objeto utilizando uma linguagem de programação, como Pascal ou C. Isto exige do usuário bons conhecimentos de programação e um profundo conhecimento da estrutura interna do sistema. Além disto, sempre que uma pequena modificação no procedimento for necessária, é necessária a recompilação de todo este módulo.

No sistema PI, não existe a noção de procedimento. É possível realizar-se o *Setup* do sistema e a análise dos dados adquiridos, salvando-se em disco os dados e o *Setup*. Mas, ao armazenar-se em disco um conjunto de aquisições/análise de dados, nenhuma informação referente à seqüência em que este foi realizados é armazenada, e para repetir-se todo o procedimento, é necessário recuperar do disco cada *Setup* individualmente, na seqüência desejada, e a execução das aquisições de dados deve ser manualmente disparada pelo operador, porque nenhuma informação sobre as Atividades realizadas é armazenada. Além disto, nenhuma relação entre os dados adquiridos/gerados e o *Setup* do ambiente no momento da aquisição/geração é mantida em disco. Assim, depende-se da boa memória e organização do operador quando quisermos recuperar as condições em que um determinado conjunto de dados foi gerado/adquirido.

Já no ambiente SGC, todos estes problemas são resolvidos no módulo de gerenciamento dos procedimentos, como será descrito no capítulo 4.

- Nenhum dos sistemas analisados provê a transferência automática de dados entre ferramentas. Normalmente o operador despende um tempo precioso fazendo com que os dados fluam entre as ferramentas. Muitas vezes há incompatibilidade nos formatos utilizados pelas diversas ferramentas, já que nenhum tipo de padronização é adotado, o que torna necessário a utilização de conversores. No ambiente SGC, graças aos mecanismos de comunicação entre ferramentas (capítulo 5) e do modelamento dos dados (capítulo 6), esta transferência automática pode ser realizada. Através destes mecanismos é possível realizar-se a troca automática de dados também entre ferramentas desenvolvidas por grupos de engenharia distintos. A ausência destes mecanismos nos sistemas analisados também impede que se realize a verificação da adequação dos dados às ferramentas, para evitar que se designe a uma ferramenta dados que ela não pode tratar. O SGC implementa esta verificação, conforme descrito no capítulo 6.

- Nenhum dos sistemas analisados se preocupa com a necessidade de documentar os procedimentos, suas execuções e os dados gerados/adquiridos. Também não é dada atenção suficiente à necessidade de armazenar os dados gerados durante a execução do procedimento de maneira coerente e oferecer ao usuário facilidades para a localização dos dados gerados/adquiridos durante a execução do procedimento. Este problema é resolvido no ambiente SGC, a partir da definição de uma Base de Dados, que está descrita no capítulo 6, e das funções que permitem ao usuário o acesso e navegação através desta.

- Nenhum dos sistemas apresentados oferece a possibilidade de visualizar os dados gerados pelo procedimento, à medida que este é executado, ou seja, monitoração em tempo real. Nestes sistemas isto não é possível porque nenhum deles apresenta um modelo de dados pré-definido, que em conjunto com funções para troca de dados em tempo real pode resolver este problema. No SGC é possível indicar, por exemplo, a transferência dinâmica de dados entre uma ferramenta que gera/adquire dados e outra encarregada de mostrar estes dados ao usuário, à medida que estes são gerados/adquiridos. Este mecanismo é implementado através da troca dinâmica de

mensagens entre as ferramentas integradas e o SGC, descrita no capítulo 5, e das facilidades oferecidas para definição do fluxo de dados, descritas no capítulo 4.

- Nenhum dos sistemas se preocupa com a necessidade de manter a compatibilidade dos procedimentos definidos pelo usuário, quando da alteração das ferramentas integradas ou integração de novas facilidades ao sistema. No ambiente SGC toma-se o cuidado de manter-se esta compatibilidade. Esta tarefa é facilitada pelo paradigma orientado a objetos adotado e pela estruturação do sistema em diversos níveis hierárquicos, conforme descrito no capítulo 3.

- Excetuando-se o sistema AOB, nenhum dos sistemas analisados é suficientemente modular. Apenas o sistema AOB e o SGC possuem módulos bem definidos, permitindo assim uma mais fácil manutenção e extensão do sistema. Tanto o SGC quanto o AOB são desenvolvidos a partir de paradigmas orientados a objeto. O sistema AOB deve toda a sua modularidade a este paradigma. O SGC, além do paradigma orientado a objetos, foi desenvolvido a partir de um modelo dividido em níveis hierárquicos, descrito no capítulo 3, com funções bem definidas, o que vem a reforçar a sua modularidade. Apesar do sistema AOB possuir módulos bem definidos, o interfaceamento entre os diversos módulos não é adequado, já que este não possue nenhum componente (módulo) dedicado a esta tarefa. No SGC, o módulo que gerência a intercomunicação (capítulo 5), trata este interfaceamento de maneira adequada, permitindo que os diversos módulos sejam desenvolvidos independentemente, em programas separados, sem a necessidade de estarem ligados (*linked*) em um mesmo programa executável.

- Nenhum dos sistemas se encarrega de gerenciar os recursos de uso comum a todos, como o gerenciamento das interfaces de hardware (como a interface com o banco de dados e as interfaces com barramentos como GPIB ou RS232), base de dados comum e gerenciamento dos mecanismos de troca de mensagem.

No ambiente SGC, o nível hierárquico mais baixo de sua estrutura engloba todas as funções necessárias para o gerenciamento dos recursos comuns, conforme descrito no capítulo 3. Assim, ao desenvolver-se ferramentas para integração no ambiente SGC, não é necessário que se escreva o código correspondente ao gerenciamento destes. Além disto, quando alterações nas funções de gerenciamento

destes recursos comuns são necessárias, não há necessidade de alterar-se o código das ferramentas integradas. Quando alterações de configuração são necessárias, estas não precisam ser repetidas para cada uma das ferramentas.

Esta divisão em níveis hierárquicos também garante a portabilidade das ferramentas, já que as mesmas estão separadas dos módulos que realizam a sua interface com os recursos disponíveis em uma instalação em particular. Assim, é possível, por exemplo, utilizar em duas instituições diferentes uma ferramenta desenvolvida para realizar a aquisição de dados via interface GPIB, a partir de um dado instrumento, mesmo que cada uma delas esteja equipada com placas GPIB que exijam *drivers* GPIB incompatíveis entre si.

- Nenhum dos sistemas apresenta uma maneira padrão para a integração de ferramentas, sem a necessidade de alterações no código fonte de outros módulos do sistema, e permitindo que ferramentas desenvolvidas em centros de pesquisa diferentes sejam facilmente integradas no sistema. No ambiente SGC existe um módulo desenvolvido especificamente para permitir a integração de ferramentas (capítulo 8). Busca-se, assim, permitir que um conjunto diversificado de ferramentas venha a ser integrado sob o ambiente SGC.

- Nenhum dos sistemas preocupa-se em evitar que as mesmas facilidades sejam incluídas em ferramentas distintas, como por exemplo a necessidade de incluir-se facilidades para visualização de dados em diversas ferramentas. No SGC a implementação da troca de mensagens entre ferramentas, modelamento dos dados e a divisão das ferramentas integradas em grupos distintos, de acordo com a sua aplicação (capítulo 3), faz com que cada ferramenta possa ser programada considerando-se apenas o conhecimento específico (facilidades) que deseja-se agregar ao sistema. O objetivo é que cada ferramenta atenda uma determinada necessidade, e a utilização do conjunto de ferramentas interativas irá suprir todas as necessidades do usuário.

2.5 Conclusão.

As alternativas propostas na literatura foram estudadas, analisando-se as suas características, para que se possa utilizar no desenvolvimento do SGC os méritos por estas atingidos, e evitar que as mesmas fragilidades nestas encontradas apareçam

também no SGC. As deficiências encontradas nos ambientes estudados na literatura foram apontadas, e maneiras de equacioná-las foram sugeridas.

3 ESTRUTURA DO AMBIENTE SGC.

3.1 Introdução:

Um sistema de *software* complexo como o ambiente SGC necessita ser estruturado em módulos bem definidos, a fim de viabilizar a sua implementação e manutenção. O ambiente SGC encontra-se subdivido em 3 níveis hierárquicos, que por sua vez são formados por diversos elementos. Neste capítulo os critérios adotados nesta subdivisão e a estruturação implementada são descritos.

3.2 A subdivisão em 3 níveis de hierarquia.

Um ambiente de teste/caracterização que atenda aos pré-requisitos mencionados nos capítulos anteriores, a nível de usuário e a nível de projetista de ferramentas, deve ter pelo menos 3 níveis hierárquicos, como ilustrado na fig. 3.1.

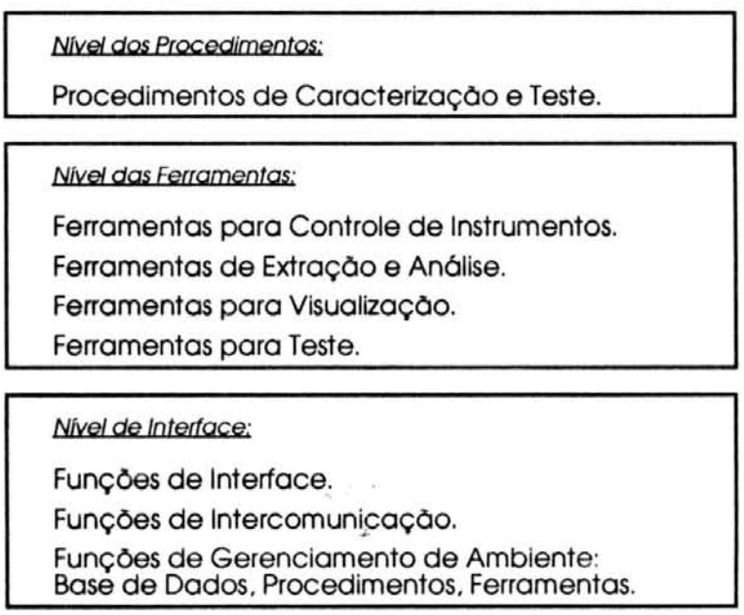


Fig. 3.1 Hierarquia dos Módulos do SGC.

Esta subdivisão em níveis hierárquicos foi definida de maneira a permitir que os elementos de um nível mais alto sejam implementados utilizando-se os elementos existentes nos níveis inferiores. Assim, os procedimentos, que estão no nível hierárquico

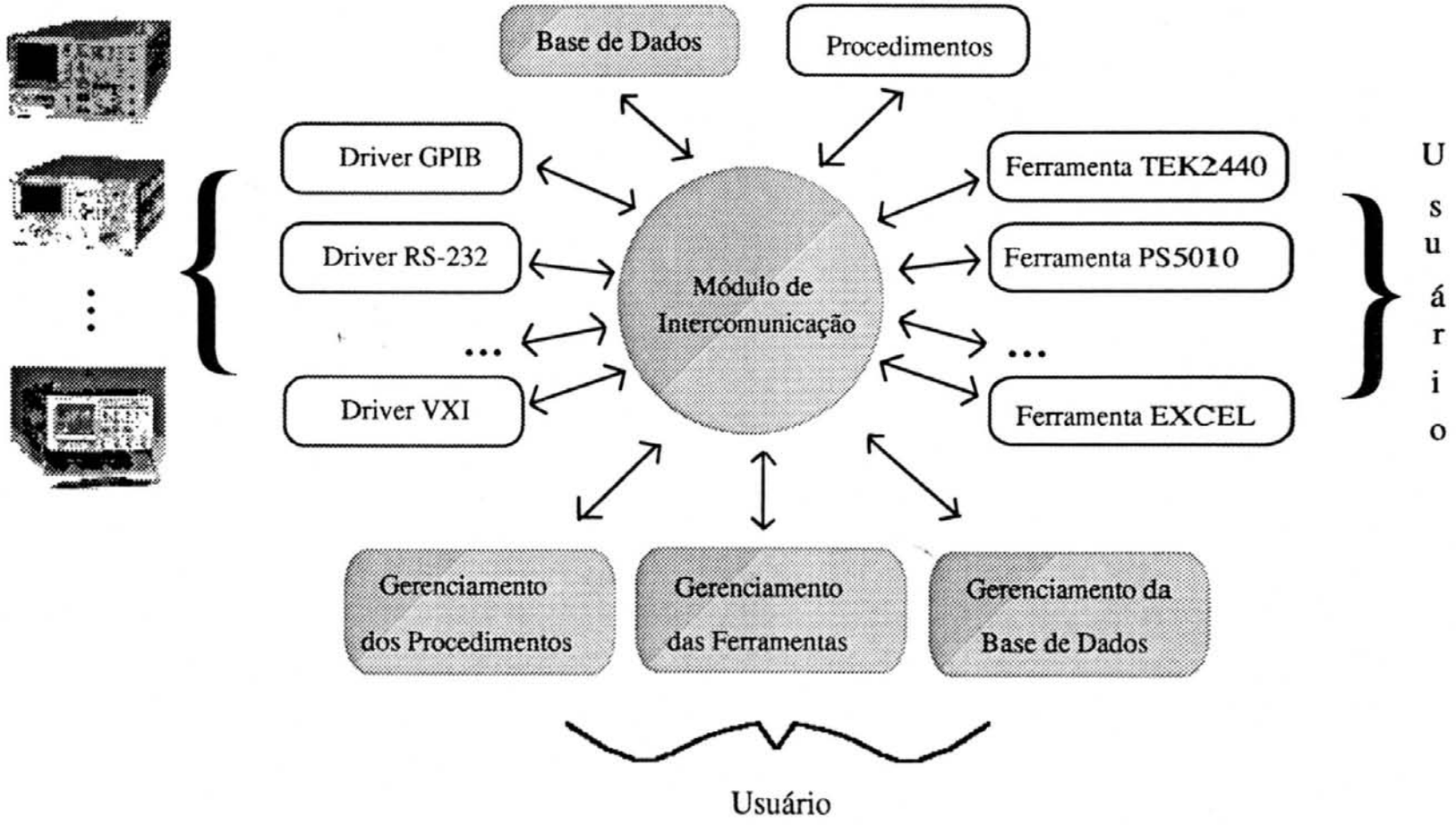
mais alto, são implementados pelo engenheiro de teste e caracterização através da utilização sistemática das ferramentas que compõem o nível intermediário, e das facilidades de gerenciamento oferecidas a ele oferecidas no nível inferior. As ferramentas, por sua vez, são implementadas pelos projetistas de ferramentas com o auxílio das funções de interface e gerenciamento de ambiente, presentes no nível hierárquico inferior.

Outro objetivo a ser alcançado através desta estruturação é desvincular ao máximo os elementos que estão no nível das ferramentas e dos procedimentos das particularidades inerentes ao *hardware* em que estes encontram-se instalados. Assim evita-se também que funções para o gerenciamento destes recursos de *hardware*, utilizados por todos os componentes dos níveis superiores, necessite ser implementado em todos eles. Além disto, são oferecidos meios para que os projetistas de ferramentas (que atuam ao nível das ferramentas), os engenheiros de teste/caracterização (que atuam ao nível dos procedimentos) e aqueles que desejarem trabalhar ao nível das funções de interface, possam incluir novas facilidades, ou realizar modificações nas já existentes, sem que ocorra a perda de compatibilidade entre os mais diversos módulos do sistema.

Cada um dos níveis hierárquicos é formado por diferentes módulos, conforme mostrado na fig. 3.2. A composição de cada um dos níveis hierárquicos é definida a partir de um paradigma de orientação a objetos /BOO91/, /MEY88/, /CHA92/ e /CHA92b/. Estes diversos módulos estão interconectados através das facilidades de intercomunicação oferecidas pelo SGC e descritas no capítulo 5. A orientação a objetos foi adotada porque esta se adequa muito bem à problemática de desenvolvimento de ambientes e ferramentas para teste e caracterização automatizados, ao facilitar a sua manutenção e auxiliar no desenvolvimento, evitando que as mesmas necessidades sejam endereçadas em diferentes módulos do sistema. Além disso, conforme descrito a seguir, os vários objetos que formam um ambiente como o SGC são facilmente identificáveis.

Os módulos que encontram-se hachuriados na fig. 3.2 são aqueles módulos cuja implementação é idêntica em todas as instalações do sistema SGC. Os módulos que não estão hachuriados são aqueles que podem ou não existir em uma instalação em particular, ou podem possuir implementações distintas, isto é, que estão intimamente relacionados aos equipamentos disponíveis no laboratório ou a metodologia de teste ou caracterização adotada. Como exemplo destes podemos citar as ferramentas para controle de instrumentos remotos, os procedimentos de teste e caracterização definidos pelo usuário, os *drivers* para interfaces de *hardware* utilizados, ou ferramentas para extração de parâmetros, que implementam um algoritmo específico.

Fig. 3.2. Os Módulos do Ambiente SGC



A seguir encontram-se descritos os elementos que compõem cada um dos níveis hierárquicos apresentados.

3.3 - Funções de Interface e Gerenciamento de Ambiente.

Os elementos pertencentes ao nível das funções de interface e gerenciamento de ambiente formam a base a partir da qual os demais módulos são construídos e gerenciados. É a partir das facilidades oferecidas neste nível que as ferramentas, que são os módulos que formam o nível das ferramentas, são implementadas. Estas facilidades incluem por exemplo: i) as funções básicas para comunicar com equipamentos remotos (como *drivers* para o gerenciamento dos barramentos GPIB, VXI e da interface serial); ii) comunicação entre ferramentas; iii) acesso a base de dados por parte das ferramentas; iv) acesso a base de dados por parte do usuário; v) gerenciamento das ferramentas por parte do usuário; vi) edição e gerenciamento dos procedimentos por parte do usuário e (vii) facilidades para a construção da interface com o usuário das ferramentas, entre outros.

As funções existentes neste nível podem ser vistas como funções de gerenciamento de ambiente, gerenciando o *hardware* e o *software* disponível ao disciplinar o acesso das ferramentas e do usuário a estes, além de realizar o tratamento dos erros que eventualmente venham a ocorrer.

O SGC oferece a possibilidade de integrar-se novos módulos a este nível a qualquer momento. Assim, *drivers* para gerenciar interfaces de *hardware* não previstas inicialmente podem ser facilmente integrados posteriormente. As facilidades oferecidas para integração de novos *drivers* estão descritas no capítulo 8.

As funções pertencentes ao nível das funções de interface estão descritas em detalhe no capítulo 6, que apresenta a base de dados e os mecanismos de acesso a esta, e no capítulo 7, onde os *drivers* já implementados estão descritos; capítulo 5, que descreve as facilidades providas para comunicação entre ferramentas e entre estas e o SGC, e no capítulo 8, onde estão descritas as facilidades para o acoplamento de novas ferramentas e o esqueleto utilizado ao desenvolver-se novas ferramentas ou *drivers* para o ambiente SGC.

3.4 Ferramentas.

Os elementos que formam este nível são as ferramentas para controle de instrumentos, realização de teste, extração de parâmetros, análise gráfica e estatística de dados, entre outros. Estas ferramentas são providas pelos projetistas de ferramentas, que utilizam as facilidades oferecidas no nível das funções de interface durante a implementação das ferramentas.

As ferramentas para caracterização e teste auxiliado por computador, que compõem um SIM, podem ser divididas em quatro grandes grupos: i) ferramentas para controle de instrumentos e aquisição de dados; ii) ferramentas para extração de parâmetros, caracterização e análise de dados; iii) ferramentas para visualização; e iv) ferramentas para teste.

O objetivo desta subdivisão é evitar que facilidades semelhantes sejam implementadas em diversas ferramentas (como por exemplo ter-se facilidades para visualização dos dados em todas as ferramentas). Cada ferramenta é desenvolvida para atender a uma necessidade específica. Ao utilizar-se o conjunto das ferramentas interativamente, todas as necessidades do engenheiro de caracterização ou teste são atendidas. Este uso interativo das ferramentas é possível graças à integração destas no ambiente SGC.

Cada componente do nível de ferramentas necessita tratar (modelar, implementar) somente as características (conhecimento) particulares a uma finalidade específica. Por exemplo, quando queremos implementar a aquisição de dados através de um equipamento remoto, controlado via interface GPIB, o *sofware* (ferramenta) implementado necessita apenas tratar das particularidades necessárias a realizar a configuração do equipamento e realizar a medida/aquisição de dados. Esta ferramenta não precisa tratar (modelar, implementar) as facilidades necessárias a fazer com que os dados trafeguem corretamente pela interface GPIB, que sejam armazenados na base de dados os dados e as condições em que estes forma adquiridos, como também não precisa oferecer facilidades para a visualização destes dados. O gerenciamento da interface GPIB e a atualização da base de dados (armazenamento dos dados e passos realizados) será realizado pelas facilidades oferecidas pelo SGC no nível das funções de interface e gerenciamento de ambiente. O *display* será realizado por outra ferramenta, que trata deste problema em especial, e é utilizada para mostrar os dados gerados/adquiridos com as mais diversas ferramentas. A transferência dos dados entre as ferramentas pode

ocorrer em tempo real, isto é, os dados poderão ser mostrados à medida em que são adquiridos, conforme descrito no capítulo 5.

Esta filosofia visa garantir a maior flexibilidade possível ao sistema. O projetista de ferramentas poderá desenvolver cada ferramenta de *software* tomando como base apenas o conhecimento que esta representa, sem dependência na estrutura do ambiente de *software* como um todo ou de uma aplicação específica (de uma metodologia de teste em particular). O mesmo ocorre ao nível dos procedimentos, conforme descrito abaixo. Este paradigma de orientação a objetos visa auxiliar a implementação, utilização e manutenção do sistema /KIL91/, /LEW91/, /GOS90/, /HEL90/ e /HAY91/.

Além de maximizar os esforços no processo de desenvolvimento e integração de ferramentas, esta filosofia também vem de encontro às necessidades do engenheiro de caracterização e teste. Quando facilidades semelhantes são incluídas em diversas ferramentas, é necessário que se aprenda a utilizá-las em cada uma das ferramentas. Por exemplo, se facilidades de visualização são incluídas em todas as ferramentas de aquisição de dados, e se sua visualização for possível apenas através das facilidades oferecidas na própria ferramenta que os dados adquire, é necessário que o usuário conheça a interface para visualização de cada uma das ferramentas particularmente. Este processo de aprendizado exige que um tempo precioso seja desnecessariamente despendido.

Os componentes pertencentes ao nível das ferramentas, já disponíveis, estão descritos em detalhe no capítulo 7. Agora passa-se a uma breve discussão da subdivisão das ferramentas em grupos.

3.4.1 Ferramentas para controle de instrumentos:

Para cada instrumento disponível no laboratório de caracterização/teste, o SGC terá uma ferramenta de *software* para controlá-lo e realizar a aquisição dos dados através dele medidos.

Tarefas rotineiras e demoradas, relacionadas com o controle/operação dos instrumentos são automatizadas utilizando-se as ferramentas providas neste nível. Entre as operações automatizadas estão: o setup do equipamento, controle da medida/teste, manipulação de erros e a aquisição de dados de medida através dos instrumentos.

3.4.2 Ferramentas para extração, caracterização e análise de dados:

Estas são as ferramentas utilizadas para extração dos parâmetros necessários à caracterização de tecnologias e análise matemática de dados. Dentre elas podemos citar os extratores de parâmetros SPICE, como o SEPE, descrito em /BAM90/, programas para tratamento matemáticos de dados como o MatLab /MAT90/ e Mathematica /CRA91/, ou planilhas de cálculo como o Excel /MIC92a/, entre outras.

Em /BAM87/ é discutida uma metodologia de caracterização e extração de parâmetros para transistores MOSFET, por exemplo, que é implementada automaticamente.

3.4.3 Ferramentas para visualização de dados:

Ferramentas para análise visual de dados também podem ser integradas ao ambiente SGC. Entre estas ferramentas podemos citar programas para visualização de curvas, como o SIMDE /FER88/.

3.4.4 Ferramentas para teste funcional de sistemas:

Neste nível encontram-se as ferramentas necessárias ao teste de circuitos integrados, como geradores de vetores de teste. Os vetores de teste gerados por tal ferramenta poderiam ser transferidos a um analisador lógico, através de uma ferramenta para controle de instrumento, e utilizados para excitar o dispositivo em teste. Em seguida a resposta do dispositivo é adquirida, pela ferramenta para controle de instrumento, e comparada com resposta esperada, pela ferramenta para teste automatizado.

3.5 Procedimentos.

Neste nível estão os procedimentos definidos pelo engenheiro de caracterização e teste. Os procedimentos são definidos, modificados e executados ao usar-se repetidamente as facilidades oferecidas neste e nos demais níveis hierárquicos.

É através dos procedimentos que o engenheiro de teste/caracterização adiciona conhecimento ao sistema. Cada procedimento reflete uma maneira particular de realizar-se um processo de teste caracterização, o que é reflexo direto da aplicação de uma metodologia de teste/caracterização. Ao permitir-se a integração destes procedimentos ao ambiente SGC, está se permitindo a integração desta metodologia de teste/caracterização ao sistema.

A linguagem utilizada para a edição dos procedimentos é gráfica. Os procedimentos são construídos a partir do conjunto de ferramentas suportadas, utilizando-se o conjunto de facilidades (controle de instrumentos, aquisição e análise de dados, extração de parâmetros, visualização, etc.) por elas implementado, e o procedimento pode englobar uma combinação de sucessivos passos de teste/caracterização utilizando um ou mais instrumentos de *hardware* e ferramentas de *software*.

O engenheiro de teste/caracterização programa cada uma das ferramentas (*setups*, atividades a serem executadas, etc.) com base no conhecimento que esta representa, sem dependência da estrutura do ambiente de *software* como um todo ou de uma aplicação específica (de uma metodologia de teste em particular). Desta maneira garante-se a maior flexibilidade possível a linguagem de definição dos procedimentos, conforme será discutido no capítulo 4.

Após a definição de um procedimento, este pode ser repetido (executado) ou modificado, sempre através de uma interface gráfica, a partir das funções de gerenciamento de procedimentos oferecidas no nível hierárquico inferior. Além disto na definição de novos procedimentos, podemos indicar a inclusão de trechos de outros procedimentos pré-definidos.

A metodologia para a edição de procedimentos e as facilidades oferecidas estão detalhadas no capítulo 4.

3.6 Conclusão.

A estrutura hierárquica do ambiente SGC e a sua divisão em módulos bem definidos garante que quando modificações em componentes de um determinado nível hierárquico são realizadas, a compatibilidade com os demais componentes, pertencentes ao mesmo nível hierárquico ou a um dos demais níveis hierárquicos, será mantida. Tomemos como exemplo o módulo de interface GPIB, que está no nível das funções de interface. Quando este módulo for modificado, como por exemplo quando da substituição da placa GPIB por outra mais moderna, nenhuma modificação será necessária nas ferramentas já desenvolvidas. Também nenhuma modificação será necessária nos procedimentos de teste/caracterização definidos pelo usuário, ou nos demais *drivers* integrados.

Da mesma forma, quando modificações forem realizadas nas ferramentas, nenhuma modificação será necessária nos procedimentos de teste/caracterização definidos pelo usuário. Eles continuarão válidos e poderão ser utilizados com a nova versão da ferramenta.

Além disto, esta estrutura hierárquica permite que informações possam ser agregadas ao sistema de maneira organizada, evitando que os mesmos problemas sejam endereçados em diferentes módulos do sistema. O usuário agrega informações ao sistema definindo procedimentos de caracterização e teste, que são incorporados ao nível hierárquico mais elevado. O projetista de ferramentas agrega informação ao sistema desenvolvendo ferramentas destinadas a realizar determinadas tarefas, incluindo assim facilidades ao nível hierárquico intermediário. As pessoas encarregadas do desenvolvimento e manutenção do ambiente SGC agregam facilidades ao nível hierárquico inferior. Facilidades estas que são utilizadas pelos projetistas de ferramentas e engenheiros de teste e caracterização.

Assim, o SGC é um ambiente de trabalho cooperativo, definindo meios para que novos recursos possam ser adicionadas em cada um de seus níveis hierárquicos, e utilizadas nos níveis hierárquicos superiores.

4 MÓDULO DE GERENCIAMENTO DOS PROCEDIMENTOS.

4.1 Introdução.

Neste capítulo encontra-se a análise e descrição das facilidades oferecidas pelo SGC ao usuário para a visualização, edição e execução dos procedimentos de teste e caracterização. Primeiramente discute-se a linguagem utilizada para a definição dos procedimentos. A seguir as facilidades que visam a assistir o usuário durante o gerenciamento destes procedimentos são abordadas.

4.2 A linguagem implementada.

Um sistema que visa automatizar os procedimentos de rotina realizados em laboratórios de teste, caracterização ou medidas, só poderá ser considerado bem sucedido se: i) sua utilização representar um incremento na produtividade alcançada pela pessoa que os realiza; ii) o processo de aprendizado não representar um ônus exagerado, e principalmente, iii) se ele vir a ser efetivamente utilizado durante a execução destes procedimentos de rotina.

Um dos elementos que tem importância fundamental para que este sucesso possa ocorrer é, sem dúvida, a linguagem oferecida para a definição destes procedimentos de rotina.

Mas, além do requisito desejável de representar um fator que contribua de forma decisiva no sucesso do ambiente junto ao usuário, a linguagem também deve atender aos seguintes pré-requisitos: iv) impor o mínimo de restrição ao universo de aplicações (procedimentos com ela realizáveis); v) ser facilmente extensível, ou seja, permitir que novas primitivas sejam facilmente acrescentadas a sua base léxica sempre que necessário, sem que os procedimentos previamente definidos tornem-se incompatíveis com esta atualização.

Em relação aos ítens i), ii) e iii), pode-se afirmar que as linguagens visuais são as que têm maior chance de obterem sucesso /GLI90/.

Em relação aos ítens iv) e v) deve-se primeiramente analisar as limitações impostas pelas linguagens para definição de procedimentos de caracterização/teste tradicionais, como a apresentada em /MUR93/, que é procedural. Ao definir-se uma linguagem para programação dos procedimentos de teste/caracterização, normalmente

tem-se a preocupação de fazer com que os termos que formam a base léxica desta sejam um conjunto mínimo cobrindo todos os possíveis problemas na área de aplicação /MUR93/. Ou seja, a aplicação (procedimento utilizado, metodologia de teste ou caracterização que pretende-se implementar) tem um papel fundamental na fase de definição da linguagem, o que imporá uma limitação no universo de aplicações ao qual esta linguagem é adequada. Por outro lado, é difícil implementar extensões na base léxica destas linguagens, visto que para tanto teremos necessariamente que modificar o código do módulo que interpreta ou compila esta linguagem. Portanto, esta metodologia de definição da linguagem não atende aos ítens iv) e v) citados acima.

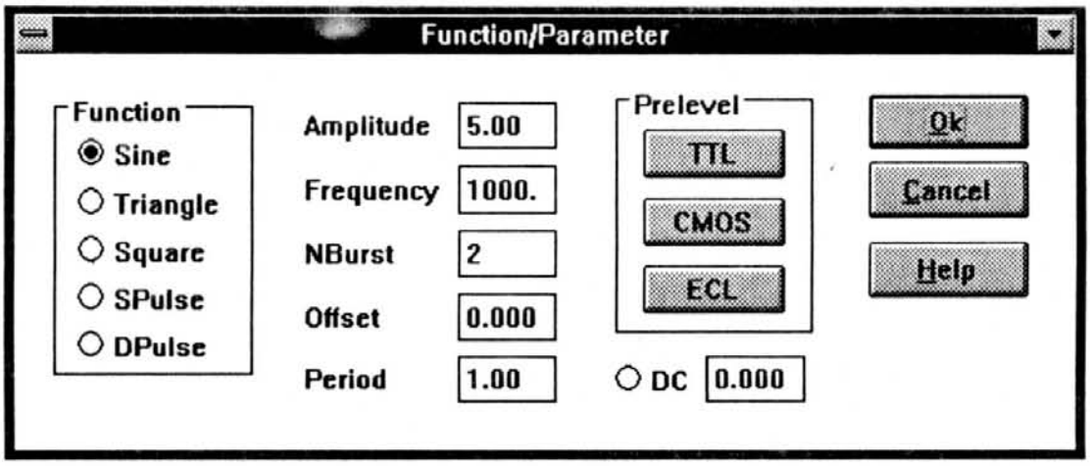


Fig. 4.1 Janela para programação de facilidades de uma ferramenta integrada (PFG5105).

A situação ideal é aquela em que cada objeto (equipamento, ferramenta de *software*, etc.) possa ser programado pelo usuário com base no conhecimento que ele representa, não dependendo da estrutura global do sistema ou de uma aplicação em particular. Se além disso, definir-se uma maneira fácil de realizar extensões na base léxica da linguagem, estar-se-á atendendo plenamente aos ítens iv) e v). Foi a partir destas considerações que a linguagem implementada no ambiente SGC foi definida.

No SGC os procedimentos são programados a partir das ferramentas integradas, com o auxílio das primitivas que permitem a definição de laços condicionais e definição do fluxo de dados entre ferramentas. Toda esta programação é realizada de maneira visual, como pode ser visto, por exemplo, em uma das janelas utilizadas para a programação das facilidades oferecidas por uma ferramenta para o controle remoto de um gerador de funções (PFG5105 da Tektronix), mostrada na fig. 4.1.

A implementação da programação (edição) dos procedimentos de maneira visual, a partir das ferramentas integradas, auxilia em muito no processo de aprendizagem. O trabalho do usuário é facilitado através da visualização e/ou edição de todos os passos de um procedimento através de interfaces gráficas, guiado por *help on-line*. As interfaces com o usuário das ferramentas são gráficas e possuem *help on-line*, como pode ser visto no capítulo 7. O Módulo de Gerenciamento dos Procedimentos também possue interface gráfica com *help on-line*, conforme mostrado no item 4.3 abaixo. As primitivas para a definição dos laços condicionais e fluxo de dados também são primitivas visuais. Estas primitivas são apresentadas nos ítens 4.3.2.1 e 4.3.2.2, abaixo.

A edição dos procedimentos está implementada de tal forma que, quando o usuário define um passo de um procedimento, todas as informações necessárias à documentação e repetição deste são repassadas pelas ferramentas ao SGC, e por este armazenadas na base de dados. Quando da execução de um procedimento, o SGC verifica as ferramentas envolvidas, então repassando-lhes as informações necessárias à repetição de cada passo, no momento apropriado. Destas informações armazenadas e transferidas entre o SGC e as ferramentas, o SGC processa apenas aquelas cujo conhecimento é necessário para proceder o gerenciamento dos procedimentos, como por exemplo a ferramenta a qual estas se referem, o tipo de dado armazenado, e a ordem em que os passos do procedimento devem ocorrer. Os dados relacionados ao objeto em particular, modelado pela ferramenta, são processados apenas pela ferramenta. Em relação a estes dados, o SGC é apenas o gerente da base de dados e implementador do fluxo de dados (transferências de dados entre ferramentas e entre estas e a base de dados). Ou seja, cada ferramenta conhece (modela, gerência) um objeto em particular (um instrumento remoto, um algoritmo de extração de parâmetros SPICE, um algoritmo para realizar transformadas de Fourier, etc.) e o SGC conhece (modela, gerencia) o objeto procedimento.

Assim, a base léxica da linguagem para programação dos procedimentos pode ser estendida sem que alterações no Módulo de Gerenciamento dos Procedimentos sejam necessárias. Esta extensão é realizada sempre que novas ferramentas são acopladas ao SGC. Este acoplamento, simples de ser realizado sem perda de compatibilidade entre quaisquer módulos do sistema, está descrito no capítulo 8. Portanto, pode-se dizer que o SGC possue uma base léxica "dinâmica" para a programação dos procedimentos, porque sempre que precisarmos incluir novas facilidades (como o controle remoto de novas equipamentos, incluir novas rotinas de extração de parâmetros, etc.), novas ferramentas

podem ser incluídas no SGC, estendendo-se a base a partir da qual os procedimentos são definidos, sem que alterações de qualquer natureza sejam necessárias nos ítens já integrados (procedimentos, ferramentas, etc.).

Esta implementação também faz com que nenhuma limitação no universo de aplicações seja imposta pela linguagem. O universo de aplicações será limitado apenas pela incompletude do conjunto de ferramentas integradas, ou seja, porque para o execução de um determinado algoritmo ou controle de um determinado equipamento, não há ferramenta integrada ao sistema.

4.2.1 As primitivas visuais para a definição de expressões condicionais.

A definição das expressões condicionais é feita a partir de primitivas visuais. Estas primitivas visam possibilitar a implementação de laços de passos, bem como a execução condicional de passos de procedimentos. Existem primitivas que permitem a definição das seguintes expressões:

- laço condicional FOR.

- expressão condicional IF THEN ELSE.

- laço condicional WHILE.

A definição adotada no SGC para estas expressões condicionais corresponde ao padrão definido para este tipo de expressão pela maioria das linguagens de programação convencionais encontradas na literatura (como C, PASCAL, FORTRAN, etc.). A diferença reside apenas na sua implementação, que é realizada de maneira visual e declarativa, conforme descrito no item 4.3.2.1 abaixo.

O laço condicional FOR é composto de 4 termos:

- A expressão de inicialização, normalmente utilizada para inicialisar variáveis.

- A expressão de teste. Esta expressão pode conter comparações (maior que, menor que, maior ou igual que, menor ou igual que, igual a, diferente de), as operações lógicas "E", "OU", "VERDADEIRO" e "FALSO", e inversões lógicas (negações).

- A expressão incremental, normalmente utilizada para realizar incremento de variáveis.

- A declaração ENDFOR, utilizada para delimitar o conjunto de passos a serem executados dentro do laço FOR.

Durante a execução de um laço FOR, a seguinte seqüência de ações ocorre:

1. A expressão de inicialização, caso exista, é executada.

2. A expressão de teste é avaliada. Se o resultado da avaliação for verdadeiro (não zero), os passos contidos dentro do laço serão executados. Se o resultado da avaliação for falso (zero), o laço FOR é encerrado.

3. A expressão incremental é executada.

4. O controle retorna ao item 2 acima.

O expressão condicional IF THEN ELSE é composto de 4 termos:

- A expressão de teste. Esta expressão pode conter comparações (maior que, menor que, maior ou igual que, menor ou igual que, igual a, diferente de), as operações lógicas "E", "OU", "VERDADEIRO" e "FALSO", e inversões lógicas (negações).

- A declaração ENDTHEN, utilizada para delimitar o conjunto de passos a ser executado quando a expressão de teste for verdadeira.

- As declarações ELSE e ENDELSE, utilizadas para delimitar o conjunto de passos a ser executado quando a operação de teste for falsa. Estas declarações são opcionais. Porém, se uma for utilizada, a utilização da outra será obrigatória.

O laço condicional WHILE é composto pelos seguintes termos:

- A expressão de teste. Esta expressão pode conter comparações (maior que, menor que, maior ou igual que, menor ou igual que, igual a, diferente de), as operações lógicas "E", "OU", "VERDADEIRO" e "FALSO", e inversões lógicas (negações).

- A declaração ENDWHILE, que delimita o conjunto de passos executado enquanto a expressão de teste for verdadeira.

Ao encontrar um laço WHILE, o SGC avalia a expressão de teste, se esta for verdadeira, o conjunto de passos delimitado por WHILE e ENDWHILE é executado, até que a expressão de teste seja falsa. Se a expressão de teste for falsa, o passo seguinte a ENDWHILE é executado.

4.2.2 As primitivas visuais para a definição do fluxo de dados.

Na linguagem implementada pelo SGC existem primitivas visuais para a definição do fluxo de dados entre ferramentas. Através destas primitivas é possível definir-se um fluxo de dados que é implementado pela troca de dados em tempo real, durante a execução do procedimento.

Existem basicamente cinco primitivas para a definição do fluxo de dados:

- GET FROM TOOL.
- GET FROM TOOL AND APPEND.
- SEND TO TOOL.
- ARITHMETIC.
- SEND TO DATABASE.

A primitiva GET FROM TOOL permite que o usuário defina ao SGC a obtenção de um objeto (dado) de uma ferramenta.

A primitiva GET FROM TOOL AND APPEND permite que o usuário defina ao SGC a obtenção de um objeto (dado) de uma ferramenta e sua concatenação a um objeto já existente na memória.

A primitiva SEND TO TOOL permite que o usuário peça ao SGC para que este envie um objeto a uma ferramenta.

A primitiva ARITHMETIC permite que operações aritméticas simples (soma, subtração, divisão e multiplicação) sejam realizadas sobre dois objetos existentes na memória do SGC.

A primitiva SEND TO DATABASE permite que se realize a transferência dos dados gerados/adquiridos durante um procedimento para a base de dados. Quando estes dados são salvos na base de dados, obviamente as relações destes com as ferramentas a partir das quais estes foram gerados e a execução (*run*) particular também são criadas. Maiores detalhes são incluídos no capítulo 6.

Quando um objeto é recebido de uma ferramenta, este é armazenado pelo SGC em uma área de memória (RAM ou disco, dependendo da disponibilidade) que este gerência. O usuário dá um nome a este objeto. Depois que este objeto foi batizado, ele poderá ser transferido a outras ferramentas.

Quando o SGC realiza o envio de objetos a uma ferramenta, ele verifica se o tipo (*data type*) destes correspondem ao tipo que esta ferramenta está aguardando. Vários objetos podem ser recebidos ou enviados a uma ferramenta em um único passo.

Sempre que o usuário acionar a primitiva visual que permite o recebimento de dados de uma ferramenta, o SGC envia uma mensagem a ferramenta, perguntando-lhe quais os dados que esta pode enviar. A seguir é exibida na tela uma caixa de diálogo onde são mostrados todos os dados que podem ser recebidos, permitindo a seleção do(s) dado(s) desejados. Da mesma forma, quando o usuário deseja enviar dados a uma ferramenta, o SGC envia a esta uma mensagem perguntando-lhe quais os dados que esta pode receber, que serão mostrados ao usuário para seleção. Juntamente com a informação de quais dados devem ser transferidos, também são repassadas entre as ferramentas e o SGC as informações referentes ao tipo desta, para que possa verificar-se a compatibilidade de tipos entre os dados transferidos.

A utilização conjunta destas primitivas e das expressões condicionais permite que a troca dinâmica de dados entre objetos seja implementada. Tomemos o exemplo da fig. 4.2, em que uma fonte de tensão, um multímetro e uma ferramenta para display gráfico de curvas são utilizados. No laço WHILE, fazemos que a variável "v" varie de 1 a 8, em passos de 0.1 Volts. No primeiro passo dentro do laço WHILE, esta variável é enviada à Fonte de Tensão (PS5010), fazendo-se assim o setup da tensão de saída deste (a alimentação de um circuito sob teste, por exemplo). No próximo passo, faz-se a leitura do Multímetro (a corrente consumida pelo circuito sob teste), transferindo-se este valor ao SGC, com o nome de "i". Finalmente, o par de objetos "v" (tensão de alimentação) e "i" (corrente medida) é transferido à ferramenta de display gráfico, que irá exibir na tela, a cada iteração do laço FOR, um ponto da curva VCCxICC. Assim, à medida que os valores da curva são gerados, eles são mostrados ao usuário.

A descrição da interface oferecida ao usuário para implementar este fluxo de dados está descrita no item 4.3.2.2.

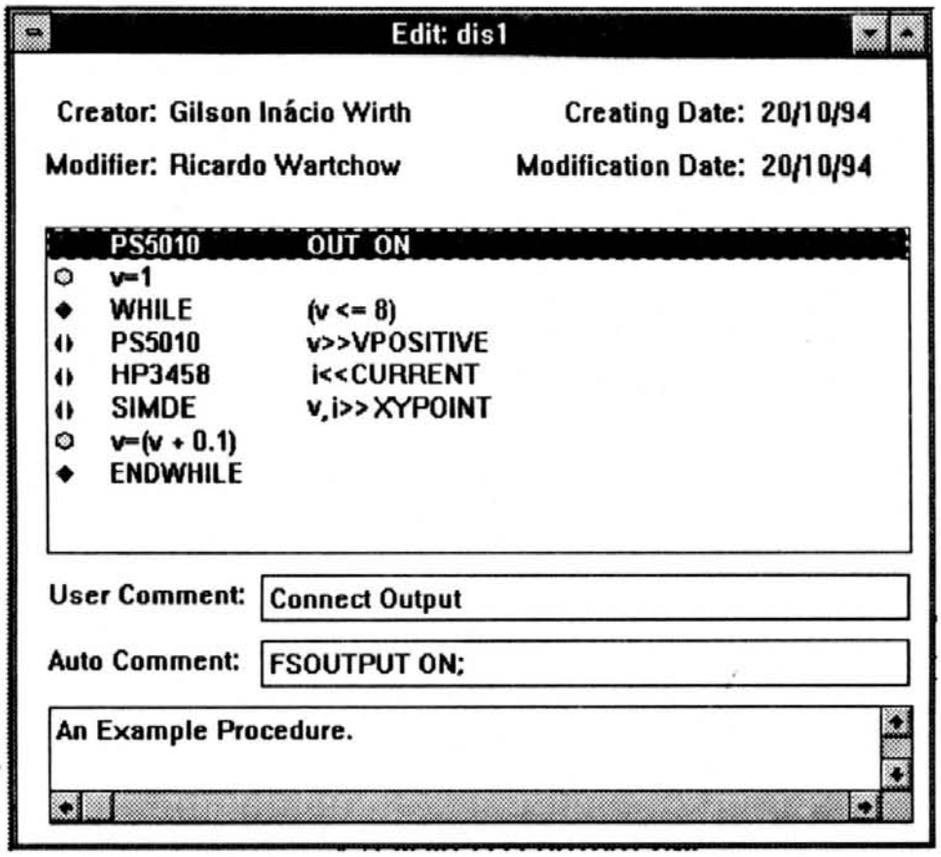


Fig. 4.2 Procedimento visualizado na janela de edição/visualização.

4.3 Gerenciamento dos Procedimentos.

O SGC oferece suporte ao usuário na tarefa de manipular os procedimentos. O usuário poderá recuperar procedimentos armazenados na base de dados, visualizá-los e editá-los, além de executá-los.

4.3.1 Recuperação dos Procedimentos.

Os procedimentos integrados ao SGC podem ser recuperados para edição ou para visualização. Quando um procedimento é recuperado apenas para a visualização, não serão permitidas alterações neste. A partir do Menu de Arquivos do SGC o usuário pode selecionar um procedimento já existente para edição, criar um novo procedimento, salvá-lo ou replicá-lo (com novo nome). Quando desejarmos apenas visualizar o procedimento, devemos utilizar a opção *ViewProcedure*, a partir do menu *DataBase*. Durante a busca por um procedimento, para edição ou visualização, os comentários dos procedimentos poderão ser visualizados, para auxiliar o usuário na identificação deste.

4.3.2 Definição de novos Passos.

Depois que um procedimento é recuperado da base de dados, ou a criação de um novo procedimento é solicitado, novos passos podem ser incluídos neste.

Para a inclusão de novos passos no procedimento, faz-se distinção entre as seguintes situações:

4.3.2.1 Definição das expressões condicionais.

A inclusão dos passos compostos por expressões condicionais é realizada a partir de um menu onde todas as primitivas condicionais são apresentadas. Esta inclusão é realizada de forma gráfica (visual). Depois que o usuário seleciona a inclusão de um determinado passo condicional, ele é assistido no processo de definição das expressões que formam este passo. Isto é feito a partir da edição visual dos seus componentes. Após a seleção do passo condicional desejado, a janela que auxilia na edição visual deste é apresentada.

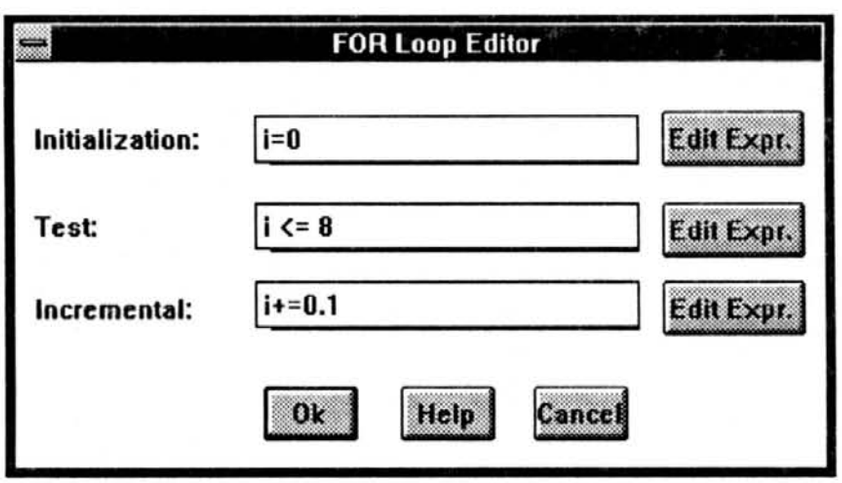


Fig. 4.3 Janela para edição do passo condicional laço FOR.

Para a inclusão de uma passo FOR, uma janela como a da fig. 4.3 é utilizada. A partir desta janela podemos editar as expressões de inicialização, teste e incremental, selecionando-se o botão *Edit Expr.* ao lado da expressão desejada.

A expressão de incialização pode ser omitida (não existirá). Caso exista, deverá ser necessariamente uma expressão do tipo fluxo de dados ou aritmética, definida

pelo usuário de acordo com o item 4.3.2.2. Em seguida o usuário é solicitado a definir a expressão de teste.

Ao selecionar-se o botão *Edit Expr.* localizado a direita do campo *Test* (fig. 4.3), podemos definir a expressão de teste, a partir da indicação dos objetos (operandos) e das operações que sobre estes devem ser realizadas. Primeiramente o usuário seleciona os operandos, a partir de uma lista de todos os objetos gerenciados pelo SGC e disponíveis em memória (RAM ou disco). Em seguida, faz a seleção da operação, selecionando o respectivo botão, conforme mostrado na fig. 4.4. É possível aninhar operações.

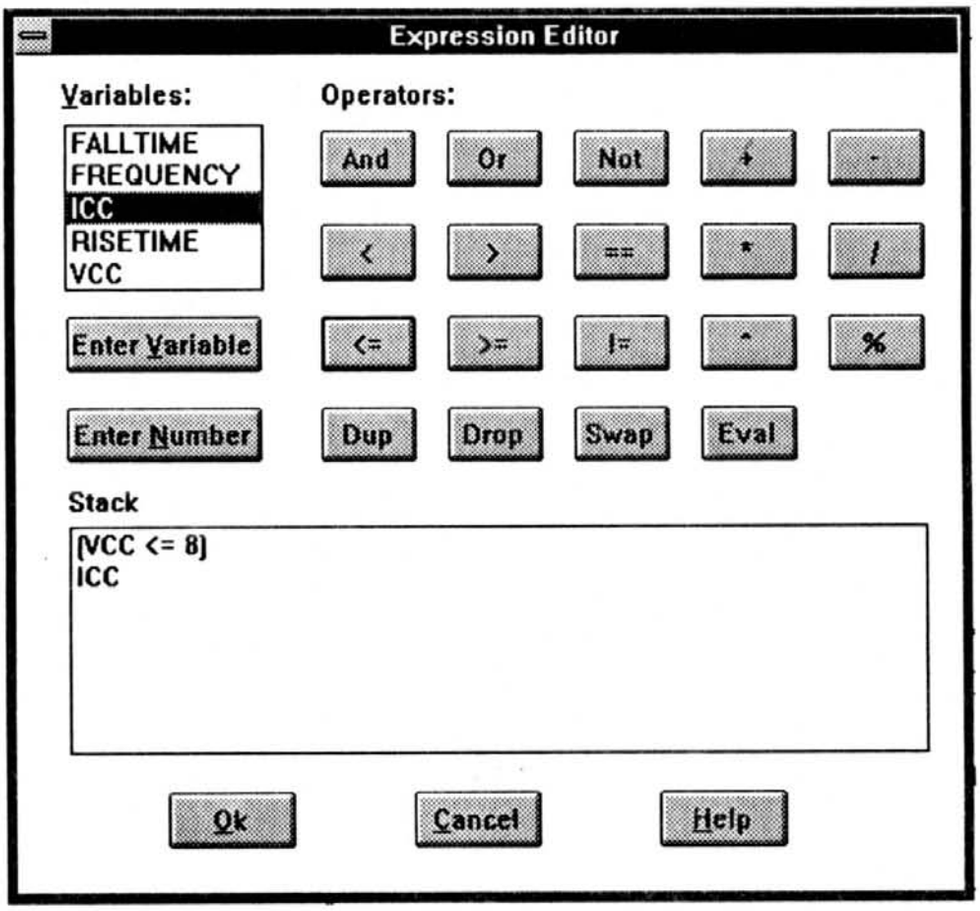


Fig. 4.4 O editor visual de expressões condicionais.

Finalmente, o usuário deve informar a operação incremental. Esta expressão incremental pode ser omitida (não existirá). Caso exista, deverá ser

necessariamente uma expressão de fluxo de dados, definida pelo usuário de acordo com o item 4.3.2.2.

A partir deste momento o usuário passa a definir os passos que farão parte do laço FOR, lembrando de incluir a indicação de final do laço, através da inserção no procedimento da declaração ENDFOR.

Na definição do laço WHILE, a expressão de teste é definida como no caso do laço FOR. O final do laço é indicado pela declaração ENDWHILE.

Na definição do laço IF THEN ELSE, a expressão de teste também é definida como no caso do laço FOR. Após a definição da expressão de teste, o usuário passa à definição do grupo de passos que serão executados quando a expressão de teste for verdadeira. Este grupo será delimitado pela expressão ENDIF. Opcionalmente um grupo de passos a ser executado se a expressão de teste for falsa pode ser definido. Este grupo de passos é selecionado através da seleção das primitivas visuais ELSE e ENDELSE.

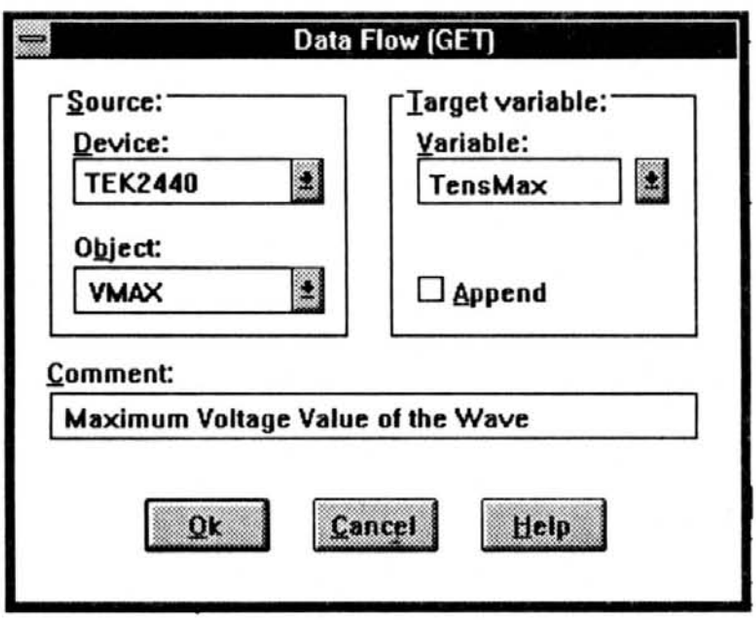


Fig. 4.5 Escolha de objeto a ser recebido de ferramenta.

4.3.2.2 Definição do Fluxo de Dados.

A definição dos passos em que ocorre a troca de dados entre o SGC e as ferramentas também é implementado de forma totalmente visual (gráfica).

Para a definição destes passos, primeiramente o usuário seleciona a primitiva visual que indica o recebimento ou o envio de dados ao SGC, a partir da janela apropriada. Após a seleção de uma destas primitivas, é exibida ao usuário uma caixa de diálogo onde estão disponíveis para seleção todas as ferramentas com as quais tal operação pode ser realizada, entre outros. É também através desta caixa de diálogo que o usuário indica os objetos a serem recebidos ou enviados.

No caso do recebimento pelo SGC de objetos a serem transferidos pela ferramenta, após a seleção da primitiva visual GET FROM TOOL, uma caixa de diálogo semelhante a da fig. 4.5 é utilizada para a escolha da ferramenta da qual o usuário deseja receber dados, e do objeto a ser transferido (ítens *Device* e *Object* do campo *Source* da caixa de diálogo da fig. 4.5, respectivamente). As alternativas disponíveis no campo *Object* são ajustadas de acordo com a ferramenta escolhida, porque após a seleção da ferramenta, o SGC envia a ela uma mensagem, perguntando quais os objetos que esta pode receber, conforme descrito no item 5.3.2. Assim, estas alternativas representam todos os objetos que uma ferramenta pode enviar, facilitando a integração das ferramentas.

No campo *Target variable* (variável destino) são mostrados todos os objetos (variáveis) presentes na memória do SGC, para que o usuário possa indicar ao SGC o nome da variável sob o qual este deve armazenar o objeto a ser recebido. Não é necessário que haja correspondência de tipos entre a variável destino e o objeto a ser recebido da ferramenta. O tipo de uma variável é definido sempre pelo tipo do último objeto que esta recebeu. Se o usuário definir a transferência para esta variável de um objeto com um tipo diferente ao anteriormente recebido, o SGC sobre-escreve o tipo da variável automaticamente. Também é possível ao usuário definir uma nova variável, bastando para isto a definição do seu nome.

No campo *Comment* um texto que auxilia o usuário na identificação da variável a ser recebida é exibido. Este texto é definido pela própria ferramenta e identifica a variável selecionada no campo *Source* item *Object*.

Quando deseja-se que o SGC receba objetos da ferramenta e os concatene a outros objetos já existentes na memória, basta que habilite-se a opção *Append* do campo *Target variable*. O novo objeto transferido será concatenado ao objeto já armazenado na variável destino.

É graças a este mecanismo que é possível, por exemplo, adquirir-se os pontos de uma curva IxV um a um, a partir de uma ferramenta que modela uma fonte de

tensão e outra que modela um multímetro, e posteriormente armazenar esta curva na base de dados, a partir da primitiva SEND TO DATABASE.

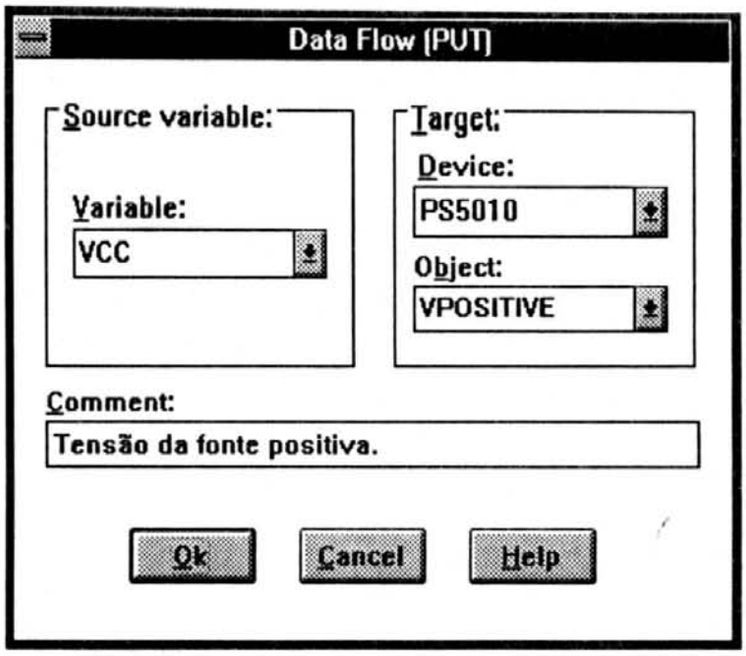


Fig. 4.6 Escolha de objeto a ser enviado a uma ferramenta.

No caso do envio de objetos pelo SGC a uma ferramenta, após a seleção da primitiva visual SEND TO TOOL e da indicação da ferramenta da qual o usuário deseja receber dados, o usuário deve, a partir de uma lista onde são mostrados todos os tipos de objetos que a ferramenta pode receber, indicar um objeto a ser transferido (esta lista é criada pelo SGC após comunicar-se com a ferramenta, perguntando-lhe quais objetos esta pode receber, conforme descrito no item 5.3.2). Assim, pode-se indicar, por exemplo, que deseja-se passar à ferramenta um par de pontos (x,y) que irá formar uma curva a ser mostrada. Neste momento o usuário não estará indicando qual o objeto que deve ser repassado, mas sim qual o tipo de objeto que a ferramenta destino deve receber e o que este objeto representa, do ponto de vista da ferramenta destino (o ponto x,y que deve ser mostrado na tela, no exemplo da fig. 4.2).

Após indicar à ferramenta quais os objetos que esta deve receber, o usuário define, através da seleção de um botão de controle, se os dados a serem transferidos são dados que estarão na memória do SGC, ou dados que estão armazenados na base de dados. Se o usuário solicitar a transferência de dados que estão na memória do SGC, serão mostrados todos os objetos presentes na memória do SGC

que correspondem ao tipo do objeto selecionado para ser recebido pela ferramenta, para que o usuário possa selecionar o objeto a ser transferido. Se o usuário optar pela transferência de dados que fazem parte da base de dados, ele poderá selecioná-los de acordo com a metodologia descrita no item 4.3.5.

O usuário poderá indicar a transferência de vários objetos a uma mesma ferramenta. Não existe limitação no número de objetos que podem ser transferidos.

A caixa de diálogo utilizada na definição da transferência de dados à ferramenta é exemplificada na fig. 4.6.

Quando deseja-se que o SGC realize operações simples sobre objetos já existentes na memória, deve-se selecionar a primitiva visual ARITHMETIC. O usuário poderá definir os objetos operando e a operação a ser realizada, bem como o objeto que receberá o resultado da operação. Toda esta programação é realizada de maneira visual. Pelo menos um dos operandos deve ser um objeto já existente na memória. Os demais podem ser constantes definidas pelo usuário. O objeto que recebe o resultado pode ser um novo objeto, criado no momento da definição do passo em questão. O SGC verificará a compatibilidade dos tipos dos operandos. O objeto que receberá o resultado terá automaticamente o tipo do objeto resultado da operação.

Para solicitar que objetos (variáveis) existentes na memória do SGC sejam transferidos à base de dados, o usuário deve selecionar a primitiva SEND TO DATA BASE. Todas os objetos previamente transferidos à memória do SGC são mostrados, sendo possível selecionar os objetos que se deseja armazenar na base de dados. Após esta seleção o usuário é convidado a informar um nome, que facilitará a sua posterior identificação, quando estiver navegando através da base de dados, conforme descrito em 4.3.5.

4.3.2.3 Definição de passos a partir de ferramentas integradas.

No Ambiente SGC dispensa-se o usuário do aprendizado de uma linguagem procedural específica para a definição dos passos relativos às ferramentas integradas. Estes podem ser definidos, visualizados e editados a partir da Interface com o Usuário de cada ferramenta. Esta implementação facilita o trabalho, já que ele poderá editar, visualizar e definir os procedimentos a partir de interfaces gráficas (as interfaces com o usuário das ferramentas são interfaces gráficas, como no exemplo da fig. 4.1).

Depois que o usuário ativou a(s) ferramenta(s) que são(serão) utilizada(s) durante o procedimento, ele passa a utilizá-las normalmente, lançando mão das facilidades que estas oferecem. As ferramentas são ativadas (executadas) a partir do

menu de *Devices*, *Analysis*, *Visualization* ou *Test* do SGC. Menus estes que abrigam as ferramentas para controle de instrumentos e aquisição de dados, para extração de parâmetros e análise de dados, visualização de dados, e ferramentas para teste, respectivamente. Através da comunicação implementada entre as ferramentas e o SGC, estas informam ao SGC as condições de execução de cada passo. O SGC se encarrega de formar o procedimento, de maneira totalmente transparente para o usuário. Não existe limite teórico para o número de ferramentas que podem estar ativas simultaneamente. Este número é limitado apenas pelos recursos da plataforma de *hardware* em que o sistema estiver instalado, dependendo basicamente da quantidade de memória disponível. Cada ferramenta é um programa independente, rodando sob o *MS-Windows*, que possue uma conexão DDE com o SGC, para comunicação com este, através do protocolo implementado no módulo de intercomunicação. O DDE é um protocolo utilizado para troca dinâmica de dados entre aplicativos, definido em /MIC90/.

4.3.2.4 Definição de passos a partir de ferramentas encapsuladas.

Para a definição dos passos a serem realizados por ferramentas encapsuladas (integração *black-box*), utiliza-se a Interface com o Usuário do *ToolPreProcessor* (descrito no capítulo 8). A interface do *ToolPreProcessor* tem as mesmas características das Interfaces com o Usuário das ferramentas integradas, e estas ferramentas encapsuladas aparecem no Menu do SGC da mesma maneira em que as ferramentas integradas aparecem, conforme descrito no item anterior. Mas, quando uma destas ferramentas é selecionada, o SGC primeiramente ativa o *ToolPreProcessor*, e não a ferramenta em si, como ocorre para as ferramentas integradas. O *ToolPreProcessor* é responsável pela interface com o usuário desta ferramenta, no momento da definição do procedimento. Esta interface permite a definição e/ou busca de dados de entrada para a ferramenta. O *Setup* desta ferramenta também pode ser feito através do *ToolPreProcessor*. O *ToolPreProcessor* também indicará a programação da geração dos dados de saída da ferramenta, de maneira que estes dados possam ser utilizados em passos posteriores deste mesmo procedimento. Além disto, o *ToolPreprocessor* prepara os argumentos a serem repassados a ferramenta encapsulada que será usada para executar o passo em questão. Estes argumentos serão repassados quando da execução do procedimento. Caso a ferramenta interaja necessariamente com o usuário durante a execução do procedimento (como para entrada de parâmetros que não podem ser passados pela linha de comando ou outro meio), esta interação ocorrerá durante a execução do procedimento, devido a impossibilidade de realizá-la durante a definição do procedimento.

4.3.3 Edição/Visualização de um Passo.

Após a seleção de um procedimento, ou à medida que passos são incluídos em um novo procedimento, o usuário poderá visualizar as informações mais importantes de cada passo, como pode-se observar na fig. 4.2. Cada passo será mostrado em uma linha. O primeiro elemento da linha é um ícone que identifica o tipo de passo em questão:

- *Setup*, identificado por um círculo azul.

- *Activity*, identificado por um triângulo vermelho.

- *DataFlow*, identificado por duas setas verdes justapostas.

- *Conditional*, identificado por um losângulo lilás.

- *Arithmetic*, identificado por um círculo amarelo.

O segundo elemento da linha identifica a ferramenta utilizada (exceto para o caso das primitivas condicionais ou aritméticas, em que não há ferramenta envolvida). Os comentários automaticamente gerados pela ferramenta (*Auto Comment*) e os comentários definidos pelo usuário (*User Comment*, para descrever o passo em questão) são mostrados na parte inferior da janela que exibe o procedimento, para a documentação do passo selecionado. O comentário do procedimento (*Procedure Comment*, definido pelo usuário para descrever o procedimento em questão) também é exibido. O *User Comment* e o *Procedure Comment* poderão ser editados pelo usuário. Além disto, também são mostrados o nome do usuário que criou o procedimento, a data em que o procedimento foi criado, o nome do usuário que realizou a última modificação, e a data desta modificação.

Selecionando-se com o *mouse* um passo de um procedimento, este poderá ser visualizado, editado ou excluído do procedimento. Para a inclusão ou exclusão de passos do procedimento, o usuário poderá utilizar-se das teclas *Delete* ou *Insert*, bem como das opções *Cut* e *Paste* (do menu *File* do SGC), respectivamente.

Para a edição e visualização, os diversos passos são tratados de maneira diferenciada:

- Quando tratar-se de um *Setup*, a ferramenta a que este se refere é invocada pelo SGC, para permitir a edição/visualização do passo. Quando o SGC invoca a ferramenta, ele passa à ferramenta to[illegible] dados referentes a este passo, permitindo

assim ao usuário visualizar o *setup* através da Interface com o Usuário da própria ferramenta (como no exemplo da fig. 4.1), donde poderá modificá-lo.

- Quando o usuário seleciona uma *Activity*, esta não será executada. O usuário será informado da atividade programada através do comentário por ele definido. Não existe opção para modificar atividades. Pode-se apenas realizar a exclusão ou inclusão de atividades. Não existe sentido em prover a modificação de atividades, porque estas serão sempre operações simples. A modificação corresponde sempre à definição de uma nova atividade, sendo portanto mais simples a exclusão da atividade em questão, após a definição da nova. A definição das atividades é realizada a partir da interface com o usuário da ferramenta a executá-la.

- Todos os dados relativos a uma primitiva *Conditional* são sempre mostrados na linha correspondente, como mostrado na fig. 4.2. A edição de passos formados por expressões condicionais se dá da mesma maneira utilizada para a sua definição, descrita no item 4.2.

- Todos os dados relativos a uma primitiva *DataFlow* são sempre mostrados na linha correspondente, como mostrado na fig. 4.2. As primitivas de definição do fluxo de dados também são editadas de maneira visual.

- Todos os dados relativos a uma primitiva *Arithmetic* são sempre mostrados na linha correspondente, como mostrado na fig. 4.2. As primitivas de definição de operações aritméticas são editadas de maneira visual.

4.3.4 Busca de passos.

O usuário pode inserir, excluir e modificar passos, bem como buscar passos armazenados na base de dados, pertencentes a este ou outros procedimentos, e incluí-los no procedimento. Isto pode ser feito para um passo em particular, ou para grupos de passos, já que funções que permitem a seleção e movimentação de blocos de passos estão disponíveis. Esta busca de passos pode ser realizada a partir de menus de Ferramentas ou Procedimentos, de acordo com as diferentes visões que pode-se ter da base de dados, conforme descrito no capítulo 6.

Quando definindo ou modificando um procedimento, o usuário poderá buscar passos previamente criados, para que estes sejam incluídos no procedimento em edição/definição.

É possível agrupar-se vários passos em um bloco. Este bloco chama-se *cluster*, podendo ser movimentado da mesma maneira que os passos normais (através de operações de *Cut*, *Paste*, *Copy*, etc.)

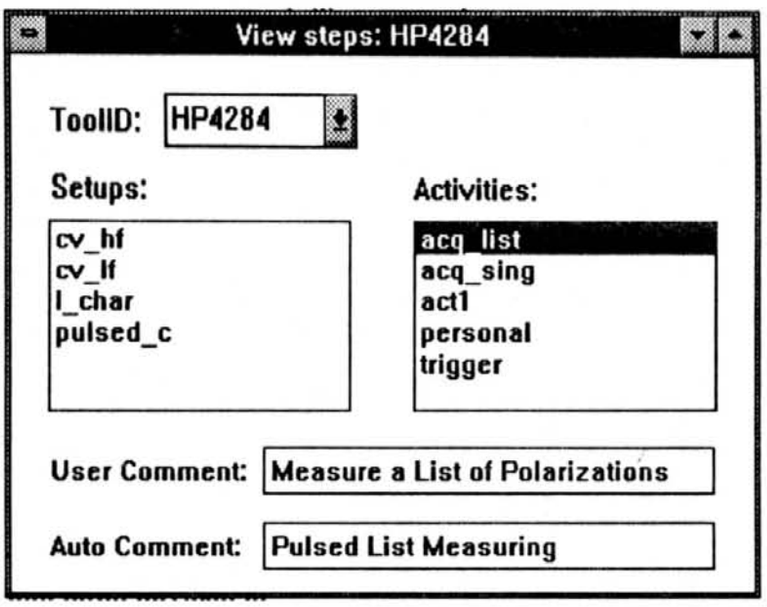


Fig. 4.7 Busca de passos (*setups* ou atividades) relacionados a uma ferramenta.

O Módulo de Gerenciamento dos procedimentos oferece duas alternativas para a inclusão de passos previamente definidos:

- Busca por Ferramenta: usado para a busca, na base de dados, de passos do tipo setup ou atividade já definidos para a ferramenta em questão. Esta busca é realizada a partir do menu de Ferramentas, conforme ilustrado pela fig. 4.7. Após a escolha da Ferramenta serão mostrados todos os *Passos* já definidos para a ferramenta em questão. Os passos estarão separados em atividades e *setups*. O usuário poderá identificar os procedimentos aos quais estes *Passos* pertencem. O usuário escolhe o procedimento e inclui os passos desejados. Se o usuário requisita a visualização do passo, a ferramenta responsável é invocada e o passo é mostrado através da Interface com Usuário da ferramenta.

- Busca por Procedimento: Permite a seleção de *Passos* existente em outros procedimentos. A partir do menu de procedimentos o usuário escolhe um procedimento, que será mostrado de acordo com a fig. 4.2. Se o usuário requisita a

visualização do passo, a ferramenta responsável é invocada e a visualização é através da Interface com Usuário da ferramenta. Esta opção permite a seleção de múltiplos *Passos*.

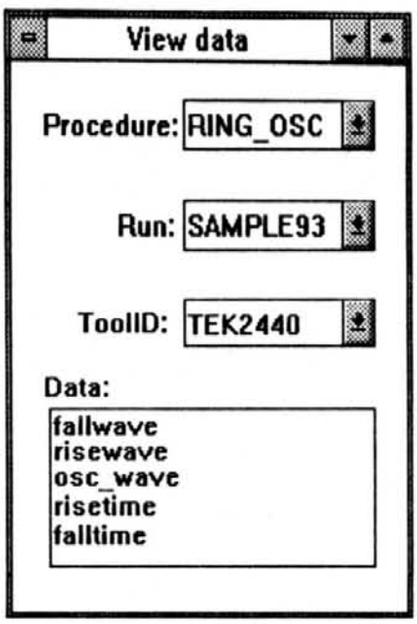


Fig. 4.8 Janela para busca de dados.

4.3.5 Busca de Dados.

Um dos componentes de um passo pode ser a definição dos dados que uma ferramenta deve receber, conforme descrito no item 4.3.2.1. Quando os dados de entrada de uma ferramenta são pertencentes a base de dados do ambiente SGC, o usuário poderá selecioná-los a partir da janela indicada na fig. 4.8. Esta janela permite:

i) Fazer uma pesquisa (browsing) a partir das execuções de procedimentos. Neste caso, os dados são identificados como os gerados pelas atividades dos procedimentos, em suas diferentes execuções.

ii) Fazer uma pesquisa (browsing) a partir das ferramentas. Neste caso, primeiramente o usuário identifica a ferramenta desejada.

Quando da seleção destes dados a serem transferidos a uma ferramenta, permite-se a escolha de múltiplas entidades, de acordo com as necessidades de cada ferramenta.

Esta implementação para a busca de dados evita que as ferramentas necessitem acessar a base de dados diretamente. O SGC permite que o usuário navegue

através da base de dados e selecione os dados desejados. Durante a execução do procedimento o SGC busca os dados e os envia à ferramenta. Antes da transferência dos dados, o SGC verifica se estes estão no formato adequado à ferramenta.

4.3.6 Execução de um Procedimento.

Depois que um procedimento é recuperado da base de dados, ou um novo procedimento é definido, ele pode ser executado.

A execução do procedimento é realizada a partir da janela mostrada na fig. 4.9. É possível iniciar a execução (*play*), resetá-la (*stop*), retroceder (*rew*) ou avançar (*ff*) um passo, suspender (*pause*) ou abortar a execução (fechando a janela).

O SGC verifica qual a ferramenta utilizada em cada passo, quando for o caso, e a executa, solicitando então que ela realize a atividade definida para o passo, realiza as transferências de dados solicitadas, ou as expressões condicionais ou aritméticas.

A medida que o procedimento é executado, a barra que indica a posição atual, na janela que mostra ao usuário o procedimento (fig. 4.9), é deslocada.

Ao final da execução do procedimento o usuário é solicitado a informar o nome da execução (*run*), e a digitar um comentário que auxilia na documentação deste. Este comentário é opcional. A documentação de cada conjunto de dados que resulta da execução de um procedimento de teste é de importância para o usuário do sistema de gerência e controle de teste.

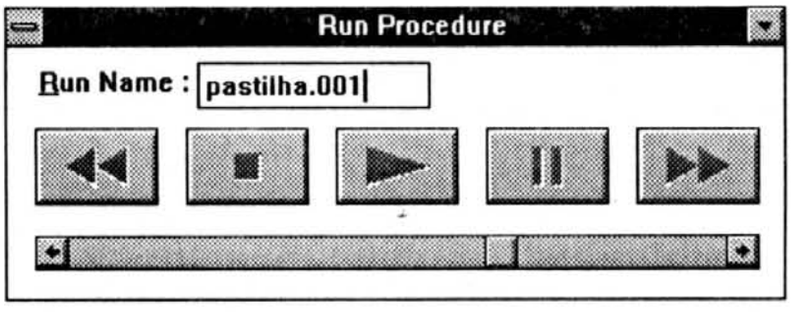


Fig. 4.9 Janela para controle da execução de procedimentos.

4.4 Conclusão.

O Módulo de Gerenciamento dos Procedimentos oferece uma interface simples e amigável para a edição, visualização e execução de procedimentos de

caracterização e teste. A linguagem implementada, fundamentalmente visual, com o mínimo necessário de componentes declarativas, e na qual cada objeto é programado com base no conhecimento que este representa, é de fácil aprendizado e utilização otimizada, impondo o mínimo possível de limitações no universo de aplicações. Esta filosofia de trabalho norteou a implementação do SGC, na expectativa de tornar o ambiente amigável, poderoso e condizente com uma produtividade adequada ao engenheiro de caraterização e teste.

5 INTERCOMUNICAÇÃO NO AMBIENTE SGC.

5.1 Introdução.

Neste capítulo são analisadas as características que o módulo do SGC que implementa a troca de dados e comandos (intercomunicação) entre os diversos módulos do SGC e as ferramentas deve possuir. A seguir a sua implementação é apresentada e discutida.

5.2 Os três níveis de intercomunicação.

Em um ambiente integrado de caracterização e teste de circuitos integrados como o SGC, existem necessidades de intercomunicação distintas. A intercomunicação é necessária, entre outros, para que: i) as ferramentas possam informar ao SGC as ações (passos) com elas definidos, durante a edição de procedimentos; ii) seja possível repetir ou editar estes procedimentos; iii) as ferramentas possam acessar as interfaces de *hardware* (como o barramento GPIB) através do SGC; e (iv) é necessária para que se possa realizar a transferência de dados, em tempo real ou através da base de dados. Também é necessário que se permita a troca de dados com ferramentas não desenvolvidas de acordo com os padrões definidos pelo SGC, e que internamente tratam os dados em formatos incompatíveis com aqueles utilizados pelo SGC.

Assim, três níveis de intercomunicação entre as ferramentas devem ser providos: i) Comunicação em tempo real; ii) Comunicação através da Base de Dados; e (iii) Comunicação com o auxílio de conversores.

Estes três mecanismos de comunicação permitirão a construção de ferramentas com um maior ou menor acoplamento às demais ferramentas. O projetista da ferramenta terá ampla liberdade de escolher o grau de acoplamento de cada ferramenta, de acordo com as necessidades a serem atendidas.

Na figura 5.1 estes três mecanismos de intercomunicação estão representados.

Havia duas alternativas possíveis para a implementação destas facilidades de intercomunicação no ambiente SGC.

Na primeira, poder-se-ia implem[illegible] biblioteca de funções que poderiam ser chamadas pelas ferramentas par[illegible] dados e comandos entre estas e

o SGC. O código objeto das ferramentas seria ligado (*linked*) ao código do SGC, formando-se então um único módulo executável.

Na segunda, implementar-se-ia estas funções através da troca de mensagens entre o SGC e as ferramentas. Estas mensagens seriam implementadas obedecendo-se o padrão estabelecido pelo Dinamic Data Exchange (DDE) do Microsoft Windows /MIC90/.

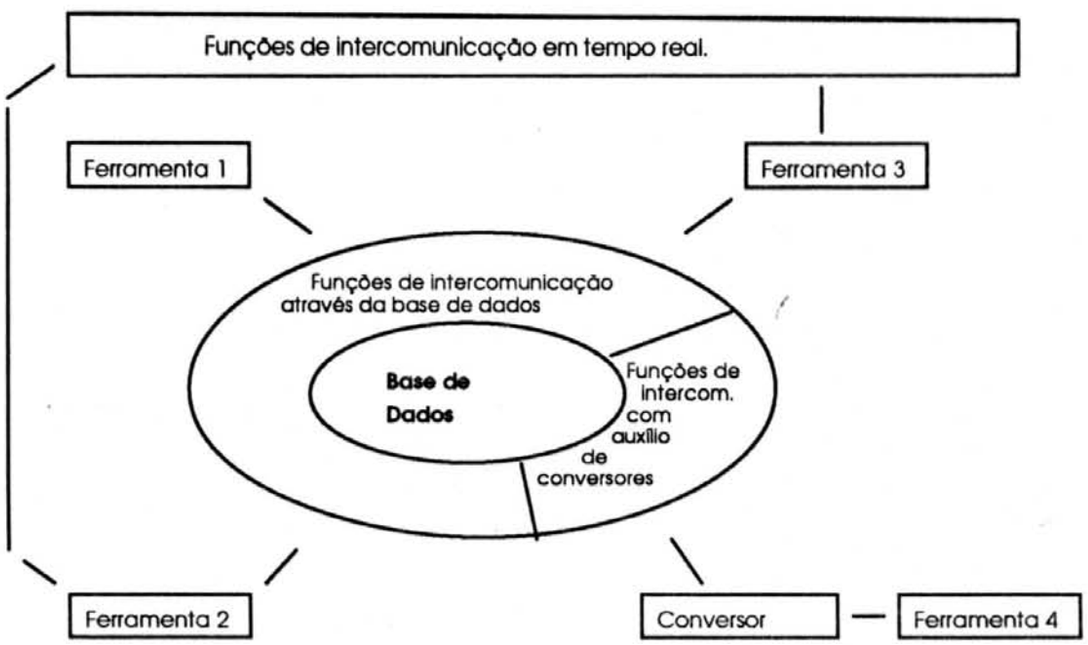


Fig. 5.1 Mecanismos de intercomunicação disponíveis no SGC.

Na implementação do SGC optou-se pela segunda alternativa. A principal vantagem desta alternativa é a sua flexibilidade. Quando, por exemplo, alguma modificação for realizada no modelamento da base de dados, as funções do SGC que manipulam a base de dados devem ser modificadas. No caso da primeira opção ter-se-ia que ligar (*to link*) os módulos objeto (.obj) de todas as ferramentas novamente, utilizando a nova biblioteca de funções para acesso a base de dados. Na segunda opção, com troca de mensagens, isto não é necessário (nenhuma modificação será necessária nas ferramentas). Apenas substitui-se o módulo do SGC que manipula as mensagens para acesso a base de dados. O mesmo vale para o caso das funções que implementam a troca de dados entre ferramentas em tempo real e acesso às interfaces de *hardware*, através do SGC.

5.3 Comunicação em Tempo Real.

O SGC implementa a comunicação em tempo real, que pode ser utilizada por ferramentas totalmente integradas (integração *white-box*). Toda comunicação que é realizada através da troca direta de objetos ou comandos entre ferramentas, sem que estes sejam armazenados em um meio intermediário, como por exemplo a base de dados, é dita em tempo real. Este nível de intercomunicação é essencial para a realização de procedimentos que dependem da sincronização dos equipamentos de caracterização e teste utilizados. Além de permitir a sincronização das ferramentas, estas funções também permitem que os dados gerados/adquiridos por uma ferramenta possam ser transferidos a outra ferramenta, a medida que esta aquisição/geração ocorre, permitindo, por exemplo, que uma curva tensão *versus* corrente (IxV) adquirida através de uma ferramenta que controla um multímetro e outra que controla uma fontes de tensão, possa ser mostrada na tela por uma ferramenta para exibição de curvas, a medida que cada ponto desta curva é gerado.

As informações trocadas entre o SGC e as ferramentas enquanto o usuário está definindo o procedimento também são mensagens trocadas em tempo real. Estas mensagens permitem que se armazene as informações necessárias a sua documentação e posterior repetição.

A intercomunicação em tempo real é também utilizada pelas ferramentas para o acesso às interfaces de *hardware*.

Estas protocolo de intercomunicação em tempo real é implementado obedecendo-se as recomendações feitas para a utilização do DDE (*Dinamic Data Exchange*) do *MS-Windows* /MIC90/.

Agora os mecanismos de intercomunicação utilizados em cada um dos casos citados são descritos.

5.3.1 Passos de Procedimento.

Conforme descrito no capítulo 4, o usuário pode proceder a definição dos procedimentos a partir da interface da própria ferramenta, bastando para isto invocá-la e realizar os ajustes (*setups*) e medidas desejados. Para que esta definição dos procedimentos possa ser implementada desta maneira transparente para o usuário, é necessário que, após cada passo realizado pelo usuário, a ferramenta com o qual este passo foi realizado informe o SGC das ações do usuário. Assim, o SGC poderá documentar este procedimento e repeti-lo no futuro, quando o usuário assim desejar.

Quando da edição de um procedimento, as ferramentas informam o SGC das ações do usuário através do envio de duas mensagens básicas:

A primeira mensagem básica, *SendActivity*, dizendo que o usuário acabou de definir uma atividade. Juntamente com esta mensagem, a ferramenta envia ao SGC o *autocomment* (comentário automaticamente gerado pela ferramenta), a informação de quais os objetos (dados) que são gerados por esta atividade, e uma estrutura de dados que só a ferramenta é capaz de interpretar, que, para a ferramenta, contém todos os dados necessários a identificação da atividade realizada.

A segunda mensagem básica, *SendSetup*, dizendo que o usuário acabou de definir um *setup*. Juntamente com esta mensagem, a ferramenta envia ao SGC o *autocomment* (comentário automaticamente gerado) e uma estrutura de dados que só a ferramenta é capaz de interpretar, que, para a ferramenta, contém todos os dados relativos ao setup definido.

Já no momento da execução de um procedimento, o SGC solicita às ferramentas que estas repitam os passos com elas definidos, a partir do envio a esta de outras duas mensagens básicas.

A primeira destas mensagens básicas, *PerformActivity*, solicita que a ferramenta execute uma atividade. Juntamente com esta mensagem, o SGC envia à ferramenta a estrutura de dados que só a ferramenta é capaz de interpretar e que, para a ferramenta, contém todos os dados necessários a identificação e execução da atividade.

A segunda mensagem básica, *PerformSetup*, solicita que a ferramenta realize um setup (ajuste). Juntamente com esta mensagem, a ferramenta recebe a estrutura de dados que só a ferramenta é capaz de interpretar e que, para a ferramenta, contém todos os dados relativos ao *setup* definido.

O código necessário ao envio destas mensagens ao SGC, e ao gerenciamento do recebimento das mensagens enviadas pelo SGC é fornecido juntamente com o esqueleto de ferramenta, descrito no capítulo 8.

5.3.2 Troca de Objetos

Vários objetos são passíveis de transferência entre as ferramentas utilizando-se a comunicação em tempo real. Esta troca de objetos (dados) é possível devido a uma definição de formatos padronizados para estes objetos. Existem vários objetos pré-definidos, como vetores, matrizes, e pontos de curvas, entre outros, descritos no capítulo 6. A qualquer momento novos objetos podem ser definidos. Para incorporar

ao SGC o suporte a estes novos objetos, basta cadastrar um novo tipo de objeto no menu de sistema do SGC.

A definição de padrões de armazenamento de dados faz com que seja possível a troca de dados entre ferramentas, mesmo que estas sejam desenvolvidas em instituições diferentes e em épocas distintas, além de evitar que o usuário envie a uma ferramenta dados que esta não pode tratar.

Para que esta troca de dados entre ferramentas possa ser implementada, existem quatro mensagens básicas.

A primeira, *SendInputObjects*, é utilizada pelo SGC para solicitar a uma ferramenta a lista de objetos que esta pode receber. Como resposta a esta mensagem, a ferramenta enviará ao SGC uma lista dos objetos que esta pode receber, contendo o nome dos objetos, textos (*strings*) explicativas (utilizadas para auxiliar o usuário na identificação destes objetos) e o tipo (padrão de armazenamento) destes. O nome que o objeto possue na lista identifica-o univocamente, e poderá ser utilizado posteriormente para enviar à ferramenta objetos identificados por este nome, quando a ferramenta saberá como tratá-los, graças a esta identificação unívoca.

A segunda, *SendOutputObjects*, é utilizada pelo SGC para solicitar a uma ferramenta a lista de objetos que esta pode enviar, ou seja, os objetos que esta gera ou adquire. Como resposta a esta mensagem, a ferramenta enviará ao SGC uma lista dos objetos que esta pode enviar, contendo o nome dos objetos, strings explicativas (utilizadas para auxiliar o usuário na identificação deste objetos) e o tipo (padrão de armazenamento) destes. O nome que o objeto possue na lista identifica-o univocamente, e poderá ser utilizado posteriormente para solicitar à ferramenta o envio do objeto identificado por este nome.

A mensagem *SendObject* é utilizada pelo SGC para solicitar a ferramenta que esta envie um objeto ao SGC. Juntamente com esta mensagem é enviado o nome do objeto que o SGC pretende receber. Como resposta a esta mensagem, a ferramenta envia ao SGC o objeto solicitado.

A mensagem *GetObject* é utilizada pelo SGC para enviar um objeto à ferramenta. Juntamente com esta mensagem o SGC envia o nome do objeto que está sendo enviado, além do próprio objeto. Através deste nome a ferramenta identifica a função que este objeto para ela tem, univocamente, e processa-o adequadamente.

5.3.3 Acesso às Interfaces de *Hardware*.

O acesso às interfaces de *hardware* também é realizado através da troca de mensagens em tempo real. As ferramentas nunca acessam estas interface diretamente. As ferramentas "não sabem" a que interface de *hardware* estas estão conectadas, ou seja, em seu código não existe nenhuma informação a respeito disto. As informações que são tratadas no código da ferramenta são apenas aquelas que dizem respeito a "o quê" estas ferramentas necessitam enviar a, por exemplo, um instrumento por elas remotamente controlado. Qual a interface a ser utilizada (GPIB ou RS-232, por exemplo), e como gerenciar esta interface (endereço dos instrumentos no barramento, terminador de *string* utilizado, recuperação de erros, etc.), são tarefas desempenhadas unicamente pelo SGC. Esta implementação traz muitas vantagens. Existem, por exemplo, equipamentos que podem ser controlados via GPIB ou RS-232, dependendo da opção escolhida no momento da compra, mas para os quais as mensagens (dados e comandos) utilizadas para o seu controle remoto são as mesmas para as duas opções. Outras vezes, podemos ter, em instalações diferentes, placas GPIB incompatíveis, ou seja, para as quais o *driver* de interface deve ser diferente. Em todos estes casos, a ferramenta implementada pode ser utilizada sem alterações, já que as funções de interface são inteiramente gerenciadas pelo SGC.

Existem várias mensagens que implementam este acesso às interfaces de *hardware*. As principais são:

- A mensagem *WriteStrData*, enviada ao SGC para que este faça chegar ao instrumento remoto ao qual a ferramenta se refere, através da interface de *hardware* adequada, a *string* enviada juntamente com esta mensagem. Esta *string* pode ser uma *ASCII null terminated string* (*string* alfanumérica terminada com o caracter zero) ou uma *binary string* (*string* binária), no caso das *strings* binárias, o número de bytes contidos na *string* deve ser informado.

- A mensagem *ReadStrData*, enviada ao SGC para que este adquira do instrumento remoto ao qual a ferramenta se refere, através da interface de *hardware* adequada, o número de bytes indicado. Estes dados serão enviados à ferramenta sob a forma de uma *string* que pode ser uma *ASCII null terminated string* ou uma *binary string*, no caso das *strings* binárias, o número de bytes contidos na *string* transferida será informado.

- A mensagem *ReadStatus*, que solicita ao SGC a aquisição do instrumento remoto ao qual a ferramenta se refere, através da interface de *hardware*

adequada, o estado atual deste (*status*). O estado do equipamento será enviado à ferramenta sob a forma de uma *string* binária. Este formato é utilizado para que obtenha-se a máxima flexibilidade (compatibilidade com todos os instrumentos). Uma *string* binária pode representar um inteiro, uma *string* alfanumérica, ou outros. A ferramenta, que conhece o formato adotado pelo instrumento remoto que esta modela, saberá interpretar este dado corretamente. Normalmente o *status* do instrumento remoto é representado por um inteiro.

- A mensagem *DeviceClear*, que solicita ao SGC que envie ao instrumento remoto ao qual a ferramenta se refere, através da interface de *hardware* adequada, a solicitação para que este retorne ao seu estado inicial (*power on state*).

- A mensagem *InterfaceClear*, solicitando que a interface utilizada seja levada ao seu estado inicial. Os projetistas de ferramentas devem evitar ao máximo a utilização desta mensagem, tomado-se muito cuidado, já que todos os instrumentos conectados à interface de *hardware* em questão serão afetados. Esta mensagem foi incluída apenas para fins de completude do conjunto de mensagens, mas seu uso é desencorajado, assim como o uso do botão de *reset* dos computadores é desencorajado.

- A mensagem *ProcessStatus*. Nas mensagens anteriores, eram sempre as ferramentas que iniciavam o processo de troca de mensagens. Esta é a única mensagem de gerenciamento de interface na qual o processo é iniciado pelo SGC. Esta mensagem é enviada pelo SGC à ferramenta quando algum instrumento remoto atua no barramento (interface de *hardware*) ao qual este está conectado solicitando serviços. O SGC identifica a ferramenta que gerencia o instrumento solicitante e repassa esta mensagem a esta ferramenta. Juntamente com esta mensagem, é enviado o estado (*status*) do instrumento. Esta mensagem é enviada a uma ferramenta, por exemplo, quando um instrumento conectado ao barramento GPIB ativa a linha de *service request* (SRQ).

5.4 Comunicação Através da Base de Dados.

Estas facilidades são utilizadas para fazer a transferência de dados entre as ferramentas, e entre as ferramentas e o SGC, com o auxílio da Base de Dados. Este é o modo de intercomunicação normalmente utilizado pelas ferramentas totalmente integradas (integração white-box) que não necessitam realizar a troca de dados em tempo real, para armazenar na base de dados os dados gerados durante a execução de um procedimento. Esta troca de dados é realizada através de objetos em formatos padronizados, que são armazenados na base de dados, ficando então disponíveis a todas

as ferramentas. Estes formatos de armazenamento de dados estão descritos no capítulo 6.

Quando, durante a execução de uma atividade (passo específico) de um procedimento, a ferramenta gerar ou adquirir dados, e desejar que estes sejam armazenados na base de dados, ela deve enviar ao SGC a mensagem *SaveActData* (salve dados de atividade). Juntamente com estes dados, a ferramenta deve indicar o tipo (padrão seguido) e o tamanho da estrutura de armazenamento destes. O SGC os armazenará na base de dados, criando todas as relações necessárias a identificação da ferramenta a partir dos quais foram obtidos, a particular execução do procedimento, e o passo do procedimento do qual derivam, entre outros, conforme descrito no capítulo 6.

Este modo de intercomunicação é normalmente utilizado por ferramentas que geram dados que podem ser utilizados posteriormente por outras ferramentas, como por exemplo o HP4145, com o qual curvas tensão *versus* corrente (IxV) são adquiridas. Curvas estas posteriormente utilizadas pelo extrator de parâmetros SEPE /BAM90/, para extrair os parâmetros Spice de uma tecnologia que está sendo caracterizada.

Já no momento de definir-se os dados de entrada de uma ferramenta, podemos percorrer os dados armazenados na base de dados a partir das facilidades oferecidas pelo próprio SGC, de maneira semelhante à descrita no capítulo anterior, item 4.3.5. Após a seleção do objeto desejado, este pode ser transferido à ferramenta, conforme descrito no item 4.3.2.2. O SGC realizará esta transferência utilizando as funções de intercomunicação em tempo real, descritas no item 5.3.2. Assim, evita-se que as facilidades para navegação (*browsing*) na base de dados sejam incluídas em todas as ferramentas. Esta metodologia também pode ser utilizada para a definição de dados de entrada quando uma ferramenta é utilizada isoladamente (*stand alone*).

5.5 Comunicação com o Auxílio de Conversores.

A condição ideal em um ambiente automatizado de caracterização e teste como o SGC, é que este seja formado por ferramentas bem comportadas, isto é, que realizem todas as suas operações (armazenamento de dados, comunicação GPIB, etc.) de uma maneira padronizada. Nem sempre isto é possível, pois existem muitas ferramentas previamente existentes, ou que ainda serão desenvolvidas em outras instituições, sem que se obedeça a filosofia proposta pelo SGC.

Para que seja possível a troca de dados entre ferramentas que armazenam seus dados em formatos incompatíveis e a utilização destas ferramentas em conjunto com

as demais integradas ao SGC, é necessária a utilização conversores de arquivos. Este mecanismo de comunicação orientado a arquivos é utilizado pelas ferramentas encapsuladas (integração *black-box*, como descrito no capítulo 8). Para cada uma destas ferramentas é necessária a escrita de um conversor de seu formato para o formato padrão do SGC, quando esta ferramenta é uma ferramenta que gera dados, ou um conversor do formato padrão SGC para o formato da própria ferramenta, quando a ferramenta recebe dados da base de dados. Obviamente, quando a ferramenta gera e recebe dados, a conversão nos dois sentidos é necessária.

Mesmo quando da utilização de conversores, a definição de um formato padrão SGC é conveniente. Quando não há um formato intermediário padrão, é necessária a escrita, no pior caso, de $2(n-1)n$ conversores, onde n é o número de ferramentas, porque para cada ferramenta é necessária a escrita de um conversor de/para o formato de cada uma das demais ferramentas. Quando há a definição de um formato padrão, no pior caso necessitamos de $2n$ conversores, um para cada ferramenta, de/para o formato padrão. Além disto, quando não há padronização, com a evolução das ferramentas de teste/caracterização, os formatos nos quais estas representam seus dados se modificam, fazendo-se necessária a manutenção constante de todos os conversores.

Para o usuário e o programador de ferramentas que recebem/geram dados que são comunicados com ferramentas que trabalham com dados que não estão no padrão SGC, a utilização de conversores é transparente.

Isto é possível porque o encapsulamento das ferramentas é realizado com o auxílio do *ToolPreProcessor* e do *ToolPostProcessor*, que são as interfaces oferecidas ao usuário para a interação com as ferramentas, conforme descrito no capítulo 8.

5.6 Conclusão.

As necessidades de intercomunicação no ambiente SGC foram analisadas. As conclusões desta análise foram utilizadas na definição de uma metodologia de intercomunicação, que foi implementada a partir de uma subdivisão das facilidades a serem oferecidas em três grandes grupos: i) intercomunicação em tempo real; ii) intercomunicação utilizando-se como meio intermediário a base de dados; iii) intercomunicação com o auxílio de conversores. Esta subdivisão oferece ao engenheiro de teste e caracterização e ao projetista de ferramentas uma grande flexibilidade, podendo-se assim acomodar as diversas situações que ocorrem durante a realização de procedimentos de teste e caracterização e projeto e desenvolvimento de ferramentas para teste e caracterização automáticos.

O uso destas facilidades de intercomunicação oferece a vantagem de que modificações no sistema tem efeito local, não exigindo troca ou recompilação dos demais módulos do sistema.

6 A Base de Dados.

6.1 Introdução.

Para que se possa compartilhar os dados entre diversas ferramentas, é necessário o estabelecimento de um padrão (modelamento) para estes. No presente capítulo, os pré-requisitos que o modelo de dados suportado pelo SGC deve possuir são estudados, definindo-se então um modelamento. A seguir, a implementação da base de dados é apresentada.

6.2 As características.

A base de dados do ambiente SGC deve suportar adequadamente dados de natureza diversa. É necessário, por exemplo, que se armazene dados como os passos dos procedimentos, que por sua vez devem estar relacionados a informações de estado de ferramentas (*setups*), transferências de dados e outras informações que permitam a repetição de atividades. É necessário que sejam armazenadas as informações fornecidas pelo usuário para a documentação dos procedimentos e das execuções destes. É necessário que os dados gerados durante a execução de um procedimento sejam armazenados adequadamente, relacionando-os com os passos de procedimentos e ferramentas a partir das quais estes foram gerados. Também é necessário que sejam armazenados os dados relativos às ferramentas, como quais interfaces de *hardware* estas utilizam, e a sua configuração em relação a estas interfaces de *hardware*, entre outros.

Outra característica importante que deve ser oferecida, é a navegação pela base de dados, de maneira que as informações desejadas possam ser recuperadas facilmente.

O modelo adotado deve permitir que os dados sejam vistos, e que se possa navegar através da base de dados, das seguintes maneiras:

i) Como um conjunto de Procedimentos: Desta forma, vê-se um conjunto de definições de rotinas de teste/caracterização. Os dados gerados [illegible]idas, saída de extratores de parâmetros, entre muitos outros) e os passos dos pro[illegible]imentos poderão ser vistos como filhos dos procedimentos.

ii) Como um conjunto de dados que foram gerados a partir de uma mesma execução de um procedimento (*run*). Estes dados tem em comum, geralmente, o fato de terem sido gerados a partir de uma mesma amostra ou pastilha, ou corresponderem ao mesmo modelamento de uma determinada realidade.

iii) Como um conjunto de atividades ou *setups*, referentes a uma determinada ferramenta. Esta maneira de percorrer os dados é útil durante a busca de passos pré-definidos, para inclusão em procedimentos que estão sendo definidos/editados.

Para que o usuário possa proceder a navegação através da base de dados e recuperar os dados propostos, transferindo-os a uma ferramenta donde pretende tratá-los, ou realizar a manutenção da base de dados, é necessário que funções como como busca, seleção, edição, e remoção de dados, entre outras, sejam oferecidas.

Quando deseja-se recuperar da base de dados passos de procedimentos, pode-se navegar através da base de dados partindo das ferramentas ou dos procedimentos. Neste caso, não faz sentido navegar-se pela base de dados partindo das execuções de procedimentos.

Quando deseja-se recuperar da base de dados os dados gerados pelas ferramentas nas diversas execuções de procedimentos, poder-se navegar através da base de dados partindo das execuções de procedimentos ou das ferramentas.

Além disto, os mecanismos de acesso a esta base de dados, como os que permitem a remoção de objetos, devem ser estruturados de forma a não permitir que o usuário leve-a a um estado inconsistente.

6.3 Modelamento dos dados:

Considerando-se as observações feitas no ítem anterior, modelou-se os dados de acordo com a fig. 6.1.

Nesta figura, os três pontos de partida (visões) distintas para o acesso aos dados, propostos no ítem anterior, são facilmente identificáveis:

i) Pode-se partir do conjunto de procedimentos armazenados na base de dados. Utilizando-se esta visão, navega-se através da base de dados a partir das

entidades *Procedure*, definidas abaixo. Os dados presentes na base de dados são vistos como dados gerados a partir destes procedimentos.

ii) Pode-se partir do conjunto de diferentes execuções de procedimentos. Utilizando-se esta visão navega-se através da base de dados tendo-se como ponto de partida as entidades *Run*, definidas abaixo, identificando-se os dados como dados a partir delas gerados.

iii) Pode-se partir do conjunto de ferramentas integradas. Utilizando-se esta visão tem-se como de partida as entidades *Tool*, também abaixo definidas, identificando-se os dados como relacionados aos passos (*steps*) para elas definidos.

A possibilidade de enxergar-se os dados da base de dados como um conjunto de procedimentos, juntamente com as facilidades de definir, modificar e repetir procedimentos, fornecem ao usuário um meio de adicionar "conhecimento" (novas informações) ao sistema, ou seja, integrar ao ambiente SGC uma metodologia de caracterização ou teste. Quando o usuário define um procedimento, ele está definindo a maneira pela qual uma rotina de teste/caracterização deve ser executada. Este conhecimento é automaticamente agregado ao sistema, quando armazena-se o conjunto de passos que formam o procedimento de caracterização ou teste na base de dados. Esta informação é adicionada ao sistema sem que sejam necessárias alterações em qualquer código fonte, através da base de dados.

Além disto, também estarão armazenadas e disponíveis na base de dados informações que auxiliam no gerenciamento dos recursos de *hardware* e *software* disponíveis (como dados sobre manutenção de equipamentos ou ferramentas, fornecedores de material de consumo para equipamentos, entre outros). Estes dados estão relacionados a uma ferramenta em particular (entidade *Tool*).

Agora passa-se a descrever cada uma das entidades que aparecem no modelo apresentado na fig. 6.1.

6.3.1 Entidade *Procedure*:

A entidade *Procedure* modela um procedimento de caracterização ou teste. Esta entidade é a portadora de todas as informações necessárias a correta execução do procedimento por ela modelado. A partir dela também é possível recuperar-se a documentação referente a dados obtidos durante uma execução deste procedimento modelado. A entidade *Procedure* possue os seguintes atributos:

Name: Nome do procedimento, que o identifica univocamente.

Comment: Texto ASCII definido pelo usuário, utilizado para auxiliar na posterior identificação do procedimento. Não possue limitação quanto ao número de caracteres que o formam.

CDate: Data de criação do procedimento. Automaticamente gerada pelo SGC, com base no relógio do sistema. Parte integrante da documentação do procedimento.

Creator: Nome do usuário que criou o procedimento. Automaticamente gerado pelo SGC. Como está descrito no ítem 8.2, ao iniciar uma seção SGC, o usuário deve identificar-se (*login*). Esta informação é então utilizada para documentação do procedimento.

MDate: Data da última modificação realizada no procedimento. Automaticamente gerada pelo SGC, com base no relógio do sistema. Parte integrante da documentação do procedimento.

Modifier: Nome do usuário que fez a ultima modificação no procedimento. Automaticamente gerado pelo SGC, a partir do *login* do usuário. Esta informação é então utilizada para documentação do procedimento.

Icon: Path para um arquivo que contém um ícone para localização visual do procedimento, a partir do menu de procedimentos.

Conforme descrito no capítulo 1, um procedimento é formado por passos, passos estes que podem ser classificados em *setups*, atividades, fluxos de dados, passos aritméticos e passos condicionais.

Para cada uma destas categorias de passo existe uma entidade que o modela.

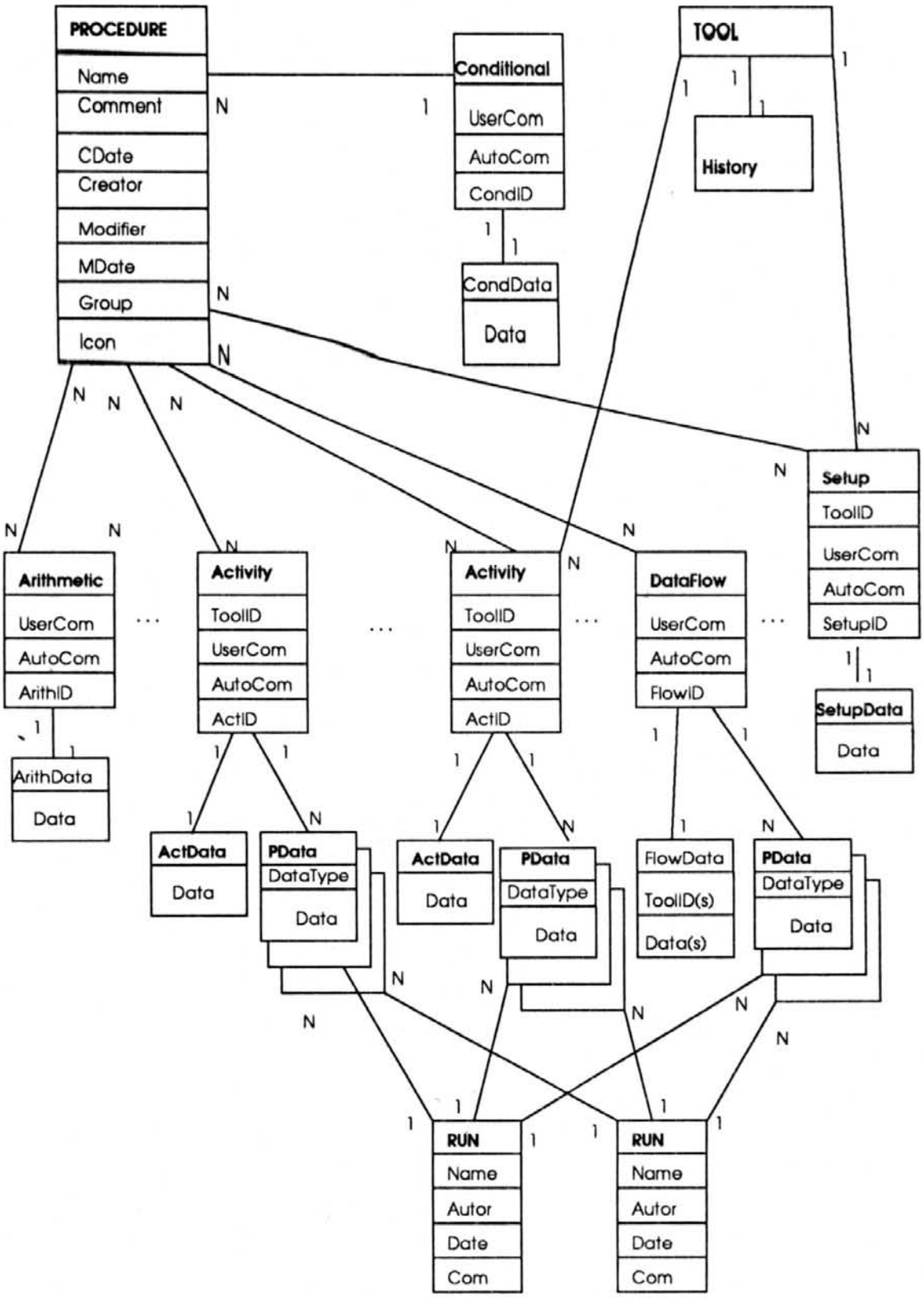


Fig. 6.1 Modelamento da base de dados.

6.3.2 Entidade *Setup*:

A entidade *Setup* modela um passo *setup*, ou seja, modela o estado de uma ferramenta em um determinado momento de um procedimento. A partir das informações nela armazenadas pode-se restaurar os ajustes (*setup*) de uma determinada ferramenta à condição em que estes encontravam-se quando da execução do passo de procedimento por ela modelado.

A entidade *Setup* possue os seguintes atributos:

- *ToolID*: Identifica univocamente a ferramenta à qual o setup se refere.

- *AutoCom*: Um comentário gerado automaticamente pela ferramenta para permitir a identificação do passo realizado sem a necessidade de analisar-se todas as informações de *setup*. Este comentário é mostrado na janela de visualização e edição do procedimento.

- *UserCom*: Comentário definido pelo usuário, explicando as característica e utilidade do *setup*. Parte integrante da documentação do procedimento. Não possue limitação quanto ao número de caracteres que o formam e é mostrado na janela de visualização e edição do procedimento.

- *SetupID*: Nome dado ao *setup*, definido pelo usuário. É através deste atributo que o usuário poderá identificar um *setup* em particular, para incluí-lo em procedimentos futuros ou em passos seguintes de um mesmo procedimento.

A entidade *Setup* possue apenas um filho, que é uma entidade do tipo *SetupData*.

6.3.3 Entidade *SetupData*:

A Entidade *SetupData* tem apenas um atributo, não estruturado, que apenas a ferramenta à qual ele se refere é capaz de interpretar, a fim de que se mantenha a orientação a objetos, porque o SGC não necessita conhecer os detalhes de como os ajustes de uma ferramenta são realizados. Esta é uma tarefa da ferramenta que modela uma determinada realidade. O SGC deve ater-se às tarefas necessárias a gerenciar os procedimentos e as ferramentas.

Este atributo contém todos os dados necessários para que posteriormente se possa restaurar o estado da ferramenta

6.3.4 Entidade *Conditional*:

A entidade *Conditional* modela um determinado passo condicional de um procedimento. A partir das informações nela armazenadas pode-se repetir a execução do passo condicional por ela modelado. Os passos condicionais encontram-se descritos em detalhe no ítem 4.2.1. A entidade *Contitional* possue os seguintes atributos:

- *AutoCom*: Um comentário gerado automaticamente pelo SGC para permitir a identificação do passo realizado sem a necessidade de analisar-se todos os dados a ele relacionados. Este comentário é mostrado na janela de visualização e edição do procedimento.

- *UserCom*: Comentário definido pelo usuário, explicando as características do passo e o motivo pelo qual este foi incluído no procedimento. Parte integrante da documentação do procedimento. Não possue limitação quanto ao número de caracteres que o formam e é mostrado na janela de visualização e edição do procedimento.

- *CondID*: Nome dado ao passo condicional. É definido pelo usuário. É através deste atributo que o usuário poderá identificar este passo condicional, para incluí-lo em procedimentos futuros ou repetí-lo em passos seguintes de um mesmo procedimento.

A entidade *Conditional* possue apenas um filho, *CondData*.

6.3.5 Entidade *CondData*:

A Entidade *CondData* tem apenas um atributo, gerado e interpretado pelo SGC. Este atributo contém todos os dados necessários à posterior execução do procedimento.

6.3.6 Entidade *Arithmetic*:

A entidade *Arithmetic* modela um passo aritmético em particular. Os passos artiméticos definidos no contexto do SGC estão descritos no ítem 4.2.2. A partir das informações nesta entidade armazenadas pode-se repetir este passo aritmético. Os atributos que esta entidade possui são:

- *AutoCom*: Um comentário gerado automaticamente pelo SGC para permitir a identificação do passo realizado sem a necessidade de analisar-se todos os dados a ele relacionados. Este comentário é mostrado na jnaela de visualização/edição do procedimento.

- *UserCom*: Comentário definido pelo usuário, explicando as característica e finalidade deste passo aritmético. Não possue limitação quanto ao número de caracteres que o formam e é mostrado na janela de visualização e edição do procedimento.

- *AritID*: Nome dado ao passo aritmético, definido pelo usuário. É através deste atributo que o usuário poderá identificar este passo artimético, para incluí-lo em procedimentos futuros ou em passos seguintes de um mesmo procedimento.

A entidade *Arithmetic* possue apenas um filho, *AritData*.

6.3.7 entidade *AritData*:

A Entidade *AritData* tem apenas um atributo, gerado e interpretado pelo SGC. Este atributo contém todos os dados necessários para posterior execução ou repetição do passo artimético a que se refere.

6.3.8 Entidade *Activity*:

A entidade *Activity* define uma atividade a ser realizada por uma ferramenta, e possue os seguintes atributos:

- *ToolID*: Identifica a ferramenta à qual o setup se refere.

- *AutoCom*: Um comentário gerado automaticamente pela ferramenta para permitir a rápida identificação do passo realizado. Extremamente útil quando estiver-se visualizando ou editando procedimentos, já que neste momento as atividades não são executadas. As atividades são executadas apenas durante a execução dos procedimentos. Este comentário é mostrado na janela de visualização e edição do procedimento.

- *UserCom*: Comentário definido pelo usuário, explicando a atividade programada. Não possue limitação quanto ao número de caracteres que o formam e é mostrado na janela de visualização e edição do procedimento.

- *ActID*: Nome dado à atividade. É definido pelo usuário. É através deste nome que o usuário poderá identificar este passo atividade posteriormente, para incluí-lo em procedimentos futuros ou em passos seguintes de um mesmo procedimento.

A Entidade *Activity* possui sempre um filho do tipo *ActData*. Esta entidade contém a informação sobre a atividade a ser executada.

A Entidade *Activity* pode ou não possuir filhos do tipo entidade *PData*. As Entidades *PData* podem ter sido geradas em uma mesma execução ou em execuções diferentes do procedimento. Por exemplo, a execução de uma atividade pode gerar mais

de um tipo de dado (como uma curva e uma tabela), assim como pode-se repetir o mesmo procedimento para várias amostras. Pode-se identificar a execução em que um determinado dado foi gerado através da entidade *Run* com que este se relaciona.

6.3.9 Entidade *PData*:

A entidade *PData* tem dois atributos. O primeiro identifica o tipo de dado armazenado, que pode indicar um dado estruturado ou não-estruturado. Este atributo é importante para permitir ao SGC a gerência da transferência de dados entre ferramentas e, em ferramentas que tratam dados estruturados, evita que sejam designados como dados de entrada da ferramenta, dados que esta não possa tratar.

Os dados estruturados são aqueles que são armazenados de acordo com um padrão que é conhecido pelo SGC e por outras ferramentas integradas, além é claro, da ferramenta que os gerou. Neste caso, este atributo identifica a natureza do dado, com por exemplo, vetores de teste, curvas para visualização, conjuntos de parâmetros SPICE, entre outros.

Os dados não estruturados são aqueles que possuem um formato de armazenamento que não é definido no contexto do SGC. Somente a ferramenta que gerou estes dados, e talvez outra ferramenta construída em conjunto com esta, podem interpretar estes dados. Assim, o SGC não poderá garantir que uma ferramenta que receba dados não estruturados sempre receberá dados no formato em que ela os espera.

É importante salientar que sempre poderão ser definidos novos padrões para armazenamento de dados. Ou seja, ao serem introduzidas ferramentas que manipulem dados que não se encaixem em nenhum dos padrões já definidos no SGC, poderá se definir um novo padrão de armazenamento de dados.

Para introduzir novos tipos de dados no ambiente SGC, o usuário é assistido pela janela de diálogo específica. O usuário deverá informar o nome do dado (uma *string* alfanumérica), e um inteiro, que o identificará univocamente. O nome será usado em diálogos futuros com o usuário, sempre que se fizer necessário referenciar-se um dado que segue este padrão de armazenamento pelo tipo, facilitando-se assim ao usuário a identificação do dado em questão. Este número que identifica o dado univocamente é utilizado quando da troca de mensagens entre o SGC e as ferramentas (descritas no ítem 5.3.2).

Para que se obtenha a maior flexibilidade possível, e também porque no momento não conta-se com uma equipe de desenvolvimento grande o suficiente, a verificação da compatibilidade de tipos de dados, no momento, resume-se à comparação

do tipo do dado armazenado com o tipo de dado que a ferramenta informa aceitar. O tipo do dado armazenado é definido pela ferramenta que os gerou. O SGC não verifica se os dados realmente seguem a padronização à qual a ferramenta diz obedecer. Ou seja, no momento cadastra-se dados aos quais dá-se um identificador de tipo que é usado na troca de dados entre ferramentas, e um nome para ser utilizado na interface com o usuário. A formatação dos dados de acordo com o padrão definido será responsabilidade de cada ferramenta. Futuramente pretende-se realizar melhorias na base de dados a fim de se verificar se os dados realmente obedecem a padronização informada pela ferramenta que os gera.

Existem vários formatos de dados estruturados pré-definidos. Entre eles pode-se citar:

Vetores: Um vetor é um objeto que pode possuir ou não um *header* (cabeçalho), possui obrigatoriamente um corpo, e que pode possuir ou não um *tail* (complemento), conforme descrito a seguir:

Em seu cabeçalho ele pode possuir um número indeterminado de linhas comentário. Estas linhas comentário são um conjunto de *strings* ASCII, que podem ser iniciadas por qualquer caracter, exceto os números de 0 a 9, um ponto ("."), ou os sinais de mais ("+") ou menos ("-"). As linhas comentário são separadas por CR/LF. O cabeçalho termina quando encontra-se a primeira linha iniciada por um dos caracteres citados acima.

Quando o cabeçalho termina, inicia o corpo. O corpo é formado por um conjunto de caracteres ASCII que representam números em ponto flutuante, separados por CR/LF. Um linha componente do corpo de um vetor deve possuir apenas um elemento. Estes elementos (números) podem estar em notação científica (mantissa e expoente) ou não. O número de caracteres da mantissa e do expoente não é limitado. O "e" indicativo do expoente pode ser maiúsculo ("E") ou minúsculo ("e"). Cada linha do corpo de um vetor pode iniciar por um número ilimitado de espaços em branco, os números de 0 a 9, um ponto ("."), ou os sinais de mais ("+") ou menos ("-"). A primeira linha que não inicia por um destes caracteres indica o final do corpo. O número de elemetos (linhas) de um vetor não é limitado.

As demais linhas do vetor são consideradas pertencentes ao complemento.

O complemento poderá ser formado por um número indeterminado de linhas de comentário.

O inteiro identificador do tipo vetor é 01 decimal.

Matriz *nxm*: É um objeto que pode possuir ou não um *header* (cabeçalho), possui obrigatoriamente um corpo, e que pode possuir ou não um *tail*.

Em seu cabeçalho ele pode possuir um número indeterminado de linhas comentário. Estas linhas comentário são um conjunto de strings ASCII, que podem ser iniciadas por qualquer caracter, exceto os números de 0 a 9, um ponto ("."), ou os sinais de mais ("+") ou menos ("-"), separadas por CR/LF. O cabeçalho termina quando encontra-se a primeira linha iniciada por um dos caracteres citados acima.

Quando o cabeçalho termina, inicia o corpo. O corpo é formado por um conjunto de caracteres ASCII que representam números em ponto flutuante, separados por CR/LF. Cada linha componente do corpo de um vetor deve possuir *n* elementos, onde *n* é o número de colunas da matriz. Estes elementos (números) podem estar em notação científica (mantissa e expoente) ou não. O número de caracteres da mantissa e do expoente não é limitado. O "e" indicativos do expoente pode ser maiúsculo ("E") ou minúsculo ("e"). Cada linha do corpo de uma matriz pode iniciar por um número ilimitado de espaços em branco, os números de 0 a 9, um ponto ("."), ou os sinais de mais ("+") ou menos ("-").

Os *n* elementos de uma linha podem ser separados por espaços em branco, vírgula, ponto e vírgula ou *tab's*. O número de linhas (*m*) e colunas (*n*) de uma matriz não é limitado.

A primeira linha que não inicia por um destes caracteres indica o final do corpo e o início do complemento. O complemento poderá ser formado por um número indeterminado de linhas de comentário.

O inteiro identificador do tipo matriz *nxm* é o 02 decimal.

Ponto x,y: É um objeto formado por dois números, iniciados pelos sinais de mais ("+") ou menos ("-"), ou espaços em branco. Seus dois elementos (números) podem estar em notação científica (mantissa e expoente) ou não. O número de caracteres da mantissa e do expoente não é limitado. O "e" indicativo do expoente pode ser maiúsculo ("E") ou minúsculo ("e"). Os dois elementos podem estar separados por espaço(s) em branco, vírgula, ponto e vírgula ou *tab*.

O inteiro identificador do tipo ponto x,y é o 03 decimal.

Inteiro: O objeto inteiro consiste apenas de um número inteiro, formado pelos algarismos de "0" a "9", podendo ser precedido dos sinais "+" ou "-", ou de espaços em branco.

O inteiro identificador do tipo inteiro é o 04 decimal.

Ponto Flutuante: É um objeto que representa um número em ponto flutuante. Pode ser iniciado pelos sinais de mais ("+") ou menos ("-"), ou espaços em branco. Seu único elemento (número) pode estar em notação científica (mantissa e expoente) ou não. O número de caracteres da mantissa e do expoente não é limitado. O "e" indicativo do expoente pode ser maiúsculo ("E") ou minúsculo ("e").

O inteiro identificador do tipo ponto flutuante é o 05 decimal.

Os objetos que não são armazenados de acordo com um padrão pré-determinado deverão ter como identificador de tipo o número decimal 00 (zero). Estes objeto serão ditos não estruturados.

O outro atributo (*Data*) da entidade *PData* contém o objeto propriamente dito, que deve estar armazenado de acordo com o padrão indicado.

6.3.10 Entidade *ActData*:

A entidade *ActData* possui apenas um atributo, não estruturado, que contém a informação sobre a atividade a ser executada. Esta informação permite a repetição da atividade programada, quando de execução do procedimento. O objeto aqui armazenado é interpretado apenas pela ferramenta a que este se refere. Para este objeto, o SGC é apenas o gerente, que o armazena na base de dados e o envia de volta a ferramenta no momento adequado. Assim mantém-se a orientação a objetos, porque o SGC não necessita conhecer os detalhes de como uma atividade deve ser executada. Ele deve ater-se aos dados necessários a gerenciar os procedimentos e as ferramentas.

6.3.11 Entidade *DataFlow*:

A entidade *DataFlow* define uma transferência de dados (passo fluxo de dados) a ser realizada pelo SGC. Esta transferência pode ser o recebimento de dados de

uma ferramenta, o envio de dados a uma ferramenta ou o envio de dados à base de dados, conforme descrito no ítem 4.2.2.

A entidade *DataFlow* possue os seguintes atributos:

- *AutoCom*: Um comentário gerado automaticamente pelo SGC para permitir a identificação do passo realizado sem a necessidade de analisar-se todos os dados de setup deste passo. Extremamente útil quando estiver-se visualizando ou editando procedimentos, já que neste momento as transferências de dados não são executadas. Este comentário é mostrado na janela de visualização e edição do procedimento.

- *UserCom*: Comentário definido pelo usuário, explicando o passo programado. Não possue limitação quanto ao número de caracteres que o formam e é mostrado na janela de visualização e edição do procedimento.

- *FlowID*: Nome dado ao passo. É definido pelo usuário. É através deste atributo que o usuário poderá identificar este fluxo de dados, para incluí-lo em procedimentos futuros ou em passos seguintes de um mesmo procedimento.

A entidade *DataFlow* sempre possuirá um filho entidade *FlowData*. Esta entidade armazena a informação referente a quais os dados a serem transferidos, e para ou de onde esta transferência deve ocorrer.

A entidade *DataFlow* pode ou não possuir um filho do tipo *PData*. Este filho existirá apenas quando o fluxo de dados programado for a transferência de dados para a base de dados, porque somente nestes caso há o armazenamento de dados por parte do SGC, que posteriormente poderão ser da base de dados recuperados. Quando tratar-se da transeferência de dados entre ferramentas, não existirá entidade *PData* relacionada a entidade *DataFlow*. Isto ocorre porque, quando de transferência de dados entre ferramentas, estes são transferidos para processamento. Se a ferramenta solicitar o armazenamento dos dados após o seu processamento, eles serão armazenados como filhos de uma entidade *Activity*. Quando trata-se do recebimento de dados de uma ferramenta, não há a criação de uma entidade *PData* porque os dados recebidos são armazenados na memória temporário do SGC, ficando disponíveis para transferência a outras ferramentas, ou a base de dados (através de um novo passo de fluxo de dados, que por sua vez gerará uma entidade *PData*).

As Entidades *PData* podem ter sido gerados em uma mesma execução ou em execuções diferentes do procedimento. Por exemplo, pode-se solicitar a transferência a disco de mais de um tipo de dado (como uma curva e uma tabela), assim como pode-se

repetir o mesmo procedimento para várias amostras. Pode-se identificar a execução em que um determinado dado foi gerado através da entidade *Run* com que este se relaciona. A entidade *PData* a qual uma entidade *DataFlow* se relaciona é idêntica aquela definida no ítem 6.3.9, para a entidade *Activity*.

6.3.12 Entidade *Run*:

Cada entidade *PData* relaciona-se com uma e apenas uma entidade *Run*, numa relação de N entidades *PData* para uma entidade *Run*. Conforme comentado anteriormente, um mesmo procedimento pode ser executado diversas vezes. Por exemplo, pode-se executar o mesmo procedimento de caracterização de osciladores em anel sobre *n* amostras diferentes. É a entidade *Run* que identifica em que execução de um determinado procedimento as entidade *PData* a ele relacionados foram obtidas.

Através da identificação de uma entidade *Run* pode-se recuperar todos os dados obtidos durante a execução de um procedimento.

A entidade *Run* possue os seguintes atributos:

Name: Nome da execução, fornecido pelo usuário, que identifica uma execução univocamente.

Comment: Texto ASCII, definido pelo usuário, utilizado para auxiliar na posterior identificação da execução. Aqui documenta-se, por exemplo, sob qual amostra de um circuito a presente execução de um procedimento foi realizada.

Date: Data da execução. Gerada automaticamente pelo SGC.

Autor: Nome do usuário que realizou a execução. Gerado automaticamente pelo SGC através da informação de *login*.

6.3.13 Entidade *Tool*:

A entidade *Tool* modela uma ferramenta integrada ao ambiente SGC. É a ela que estão relacionadas as informações que permitem ao SGC e ao usuário seu gerenciamento.

A uma entidade *Tool* relacionam-se entidades do tipo *Setup*, *Activity* e *DataFlow* acima descritas, que representam os passos definidos com a ferramenta modelada pela entidade *Tool* em questão.

A entidade *Tool* possui os seguintes atributos:

Name: É uma string com o nome da ferramenta. Este é nome que será mostrado ao usuário no menu de ferramentas do SGC. Com um "double click" de mouse sobre este nome, o usuário ativa a ferramenta.

BusConnection: Inteiro que indica se a ferramenta se comunica com algum equipamento através de alguma interface de Hardware, como por exemplo o barramento GPIB ou VXI. No código fonte das ferramentas não há qualquer referência a interface de *hardware* a ser utilizada para comunicação com eventuais equipamentos remotos. Esta configuração é realizada durante a instalação da ferramenta no ambiente SGC, quando esta informação é armazenada neste atributo. Durante a execução de procedimentos, ou utilização da ferramenta em modo *stand alone* (indepedentemente), o SGC identificará, através deste atributo, qual a interface de *hardware* a utilizar como meio para o envio ou recebimento de mensagens e comandos trocados com eventuais equipamentos remotos.

Este atributo é um inteiro que possue um dos seguintes valores possíveis:

NULL: Quando a ferramenta não está conectada a nenhuma interface de *hardware*.

GPIB: Quando a ferramenta utiliza o barramento GPIB para comunicar-se com algum equipamento remoto.

VXI: Quando a ferramenta realiza comunicação através do barramento VXI.

RS-232: Quando a ferramenta comunica-se com equipamentos remotos utilizando-se da interface serial (padrão RS-232).

Na medida que novos *drivers* para interfaces de *hardware* forem construídos, novos valores podem ser agregados a tabela acima.

Além disto, cada entidade *Tool* também se relaciona com uma entidade *History*, descrita abaixo.

6.3.14 Entidade *History*:

A entidade *History* possui atributos que armazenam informações genéricas a respeito da ferramenta. Estas informações auxiliarão nas atividades de gerenciamento dos recursos de *hardware* e *software* disponíveis, por parte do usuário. Pretende-se que estas informações representem o histórico da ferramenta, onde todas as

informações importantes a ela relacionadas estão armazenadas e podem ser rapidamente recuperadas.

Todos estes dados podem ser editados com o auxílio da janela gráfica oferecida pelo SGC.

Os atributos da entidade *History* são os seguintes:

Vendor: *String* alfanumérica que armazena o endereço, telefone, fax e nome da pessoa para contatos, entre outros, do fornecedor da ferramenta. Muito útil quando for necessário realizar-se contatos com a empresa que forneceu a ferramenta.

Manutenção: *String* alfanumérica que armazena o endereço, telefone, fax e nome da pessoa para contatos, da empresa responsável pela manutenção da ferramenta. Muito útil quando for necessário realizar-se contatos com a empresa responsável pela manutenção da ferramenta.

Insumos: *String* alfanumérica que armazena o endereço, telefone, fax e nome da pessoa para contatos, entre outros, de empresas que fornecem material de consumo para a ferramenta (como microponteiras, no caso de microprovadores, por exemplo).

Events: *String* alfanumérica que armazena a história dos eventos que já ocorreram com a ferramenta (problemas apresentados, calibração/manutenção realizada, etc.).

Peculiaridades: *String* alfanumérica que armazena peculiaridades da ferramenta e cuidados especiais a serem tomados durante sua operação ou manutenção.

O Diagrama Entidade-Relacionamento (ER) que representa o modelo aqui apresentado para a base de dados encontra-se na Fig. 6.2.

As funções que implementam o acesso a base de dados, através de mensagens (comandos e dados) trocadadas entre o SGC e as ferramentas, estão descritas no capítulo 5.

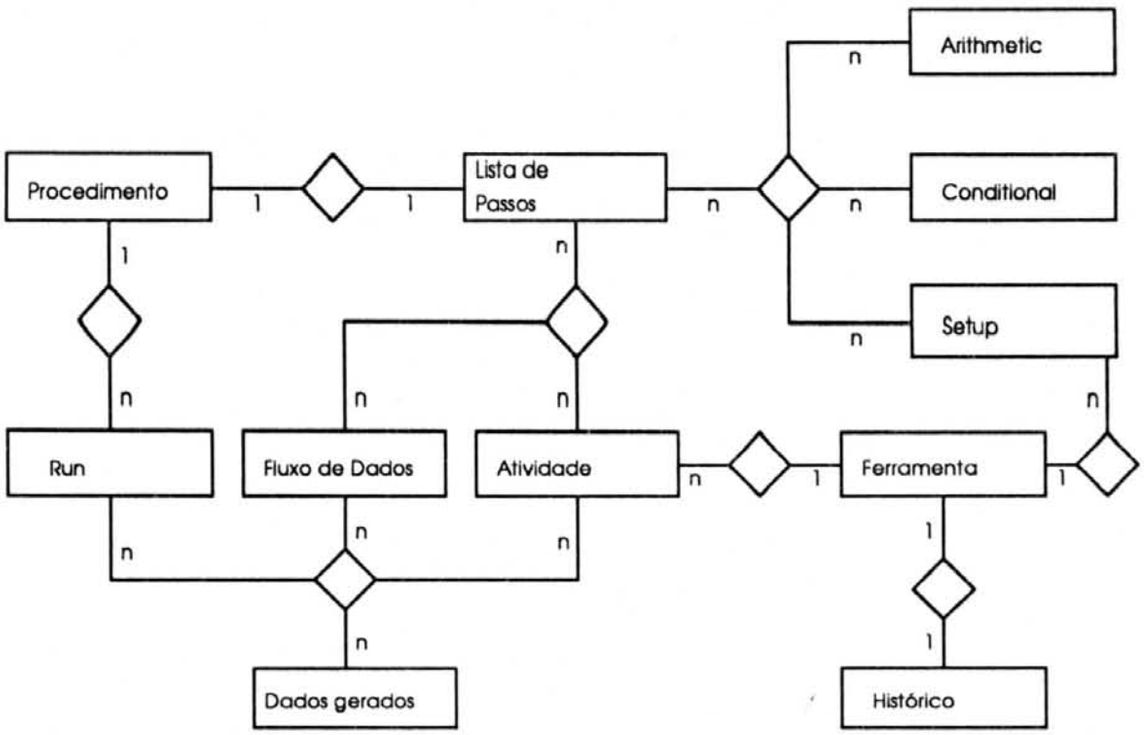


Fig. 6.2 Diagrama Entidade-Relacionamento para a Base de Dados.

6.4 Consistência da Base de Dados.

A consistência da base de dados é feita de forma a garantir que todas as entidades e seus relacionamentos contenham dados compatíveis, e que todas as entidades com as quais uma determinada entidade se ralaciona existam.

É importante salientar que nenhuma ferramenta acessa a base de dados diretamente. Sempre que uma ferramenta deseja acessar a base de dados, ela o faz enviando mensagens ao SGC, conforme descrito no capítulo 5. Esta característica facilita em muito a implementação dos mecanismos para garantir a consistência da base de dados.

Quando a ferramenta solicita ao SGC o armazenamento de um determinado dado ela deve informar qual o tipo do dado, deixando ao SGC a tarefa de armazená-lo corretamente. Assim, a responsabilidade de garantir a consistência quanto ao conteúdo da entidade é dividido entre o SGC e as Ferramentas. Por exemplo, quando uma ferramenta solicita ao SGC o armazenamento de um *setup*, ele garante que os atributos *ToolId*, *AutoCom*, *SetupId*, bem como os atributos da entidade *SetupData* existirão e serão armazenados corretamente, e que as relações necessárias (com

ferramentas e procedimentos) serão criadas. Mas é responsabilidade da ferramenta, neste exemplo, a correção dos dados armazendos (pelo SGC) no atributo *Data* da entidade *SetupData* gerada, porque, de acordo com o paradigma orientado a objetos adotado, apenas a ferramenta sabe interpretar estes dados de *setup*.

Para que se possa garantir a consistência quanto a existência das entidades com as quais uma determinada entidade se relaciona, as seguintes limitações às ações do usuário sobre a base de dados são adotadas:

- Não será permitida a exclusão de uma entidade *Activity* ou *DataFlow* de um procedimento, enquanto existirem entidades *PData* delas derivadas. Se o usuário desejar excluir de um procedimento uma entidade *Activity* ou *DataFlow* que possua entidades *PData* a ela relacionadas, o usuário deverá confirmar a exclusão simultânea da *Activity* ou *DataFlow* e dos dados a partir delas gerados.

- Sempre que ocorrer a exclusão de um procedimento, todas as entidades *Run* a ele relacionados também são excluídos (o usuário será informado da existência das entidades *Run*, que só serão excluídos, caso este confirme a operação). Ao excluir-se as entidades *Run* e *Procedure*, os dados (*PData*) a eles relacionados também são excluidos. Assim não permite-se que ocorra a proliferação na base de dados de objetos cujas condições em que estes foram gerados não são conhecidas.

- Quando edita-se um procedimento que possui entidades *Run* a ele relacionados, este deverá ser salvo na base de dados com um novo nome, para que evite-se confusões com relações as condições em que os dados a ele relacionados foram gerados.

- Sempre que ocorrer a exclusão de uma entidade *Run*, todos as entidades *PData* a ela relacionadas também serão excluídas. O usuário será informado da existência das entidades *PData* que só serão excluídas caso este confirme a operação. Esta é uma maneira simples de, por exemplo, remover-se da base de dados todos os dados referentes a uma amostra não mais desejada.

6.5 Implementação da Base de Dados.

Para a implementação da base de dados acima modelada não utilizou-se nenhum sistema de banco de dados previamente existente. Esta base de dados foi localmente implementada em C++, assim como os demais módulos do ambiente SGC. Após o modelamento da base de dados analisou-se vários sistemas de banco de dados

comerciais, como DBase (Clipper) /VDA92/, Oracle /ORA90/ e Access /MIC92b/. Mas nenhum destes mostrou um bom compromisso entre as facilidades oferecidas, a complexidade de implementação e modelamento, e as necessidades impostas pelo ambiente SGC.

As facilidades implementadas para o acesso à base de dados permitem a navegação nesta a partir de 3 pontos distintos, conforme descrito anteriormente.

Para visualizar e navegar através da base de dados tendo como ponto de partida os procedimentos, o usuário deve utilizar o *menu* "*File*" (Arquivo) ou o menu "*DataBase*" (Base de Dados) *sub-menu* "*View Procedure*" (visualizar procedimento), disponíveis no SGC. A partir deste *menus* o usuário poderá visualizar e/ou editar os procedimentos.

Para navegar através da base de dados tendo como ponto de partida as ferramentas (entidades *Tool*) ou as execuções de procedimentos (entidades *Run*), o usuário é auxiliado através de uma caixa de dialogo semelhante a da fig. 4.7, do capítulo 4.

Estas facilidades de navegação também são utilizadas quando o usuário deseja, durante a definição de procedimentos ou utilização de uma ferramenta isoladamente (modo *stand-alone*), indicar como dados de entrada para a ferramenta dados que fazem parte da base de dados. Para indicar à ferramenta que esta utilize dados presentes na base de dados, o usuário utiliza-se da primitiva de definiçao do fluxo de dados SEND TO DATABASE, apresentada no ítem 4.3.2.2. Esta situação ocorre, por exemplo, quando necessita-se recuperar da base de dados as curvas IxV e/ou CxV de transistoress e transferí-las a um programa extrator de parâmetros SPICE. Estas curvas serão então os dados de entrada para o algoritmo de extração de parâmetros. A ferramente, ao atender a solicitação do SGC para permitir a indicação dos dados de entrada, não acessará a base de dados. A ferramenta enviará uma mensagem ao SGC (de acordo com ítem 5.4), indicando os tipo de dados que esta pode receber. Ao receber esta mensagem, o SGC permitirá ao usuário a navegação através da base de dados e escolha dos dados de entrada adequados para a ferramenta, garantindo que os dados à ferramenta designados possuam o tipo (padrão de armazenamento) por ela esperado. Esta metodologia evita que sejam incluídas em todas as ferramentas as mesmas facilidades de navegação através da base de dados, facilitando a manutenção do ambiente e simplificando a implementação dos mecanismos responsáveis pela consistência da base de dados.

6.6 Conclusão.

Definiu-se e implementou-se uma base de dados que suporta a troca de dados entre ferramentas e entre estas e o SGC de acordo com os mecanismos descritos no capítulo 5. Esta base de dados também respeita o paradigma orientado a objetos seguido em todos os demais módulos do ambiente SGC, e não permite que o usuário a leve a um estado inconsistente. Além disso, a base de dados foi planejada e implementada de forma a obter-se a máxima flexibilidade em relação ao universo de objetos que podem ser nela armazenados e por ela gerenciados, e a ser um código de fácil manuntenção.

7 Ferramentas e *Drivers*.

7.1 Introdução.

Nos capítulos anteriores foi possível observar-se que o SGC oferece um poderoso conjunto de facilidades ao engenheiro de caracterização e teste, para que se possa automatizar todos os procedimentos de rotina realizados no laboratório. Mas, para que isto possa ser efetivamente realizado, é necessário que estejam integrados ao ambiente SGC objetos de *software* que modelem equipamentos remotos, algoritmos de extração de parâmetros, algoritmos de teste e facilidades de visualização, objetos estes chamados de ferramentas. Também é necessário que existam objetos de *software* que modelem as interfaces de *hardware* existentes e permitam a sua utilização, objetos estes chamados de *drivers*.

Neste capítulo as ferramentas e *drivers* já implementados e integrados ao ambiente SGC são apresentados e suas principais características são discutidas. As linhas que devem guiar os projetistas de ferramentas e *drivers* quando da definição de novas ferramentas e *drivers* a serem desenvolvidos também são discutidas.

7.2 Ferramentas Integradas.

No momento já existem diversas ferramentas integradas ao ambiente SGC. As ferramentas até o momento integradas pertencem ao grupos das ferramentas de controle de instrumentos e aquisição de dados, conforme a subdivisão em grupos, adotada e discutida no capítulo 3. No item 7.4 discute-se as razões pelas quais optou-se por implementar primeiramente as ferramentas pertencentes a este grupo.

Agora passa-se a apresentação das ferramentas integradas. Esta apresentação não pretende desempenhar o papel de manual do usuário. Estes manuais são individualizados para cada ferramenta. Aqui deseja-se apenas apresentar as características principais de cada uma, informando ao potencial usuário as facilidades por elas implementadas. Todas as ferramentas foram desenvolvidas utilizando-se o compilador C++ BorlandC /BOR91/.

7.2.1 Ferramenta para controle remoto do analisador de parâmetros HP4145B.

O HP4145B *Semiconductor Parameter Analyzer* (analisador de parâmetros de semicondutores), fabricado pela Hewllet-Packard, permite que medidas tensão *versus* corrente (IxV) sejam realizadas em dispositivos ou circuitos, encapsulados ou não.

Através da ferramenta localmente implementada temos acesso remoto a todas as facilidades por ele oferecidas, além de existir a possibilidade de acessar-se as facilidades oferecidas no modo que este chama de *User Mode* (modo usuário) /HP89a/, facilidades estas que não estão disponíveis quando o equipamento é operado pelo seu painel frontal (localmente). Na fig. 7.1 temos uma das caixas de diálogo utilizadas para o controle remoto destas facilidades.

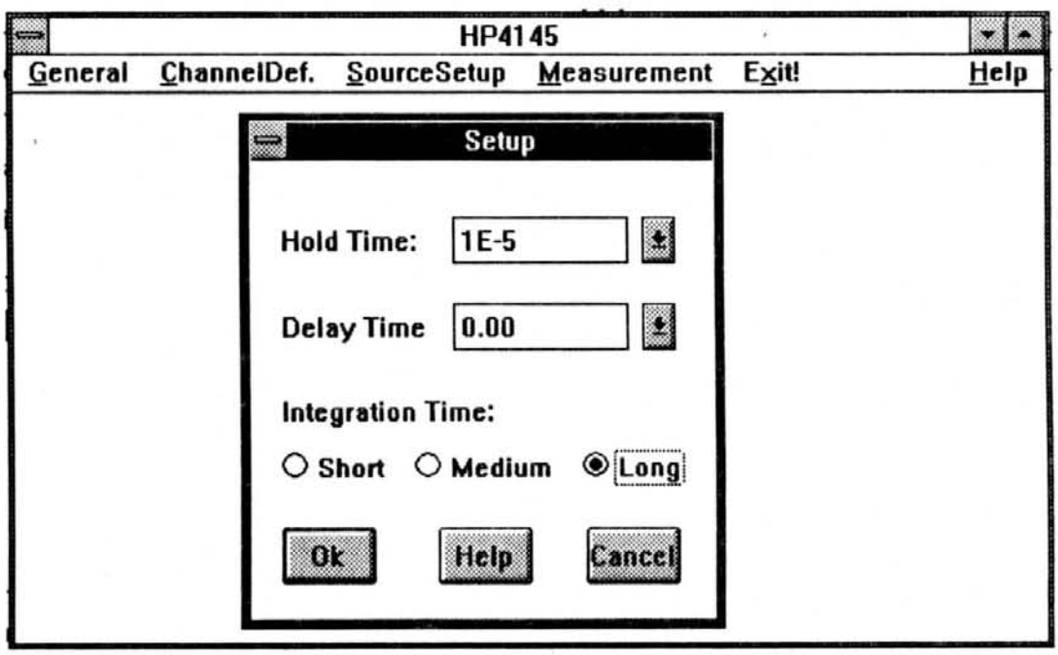


Fig. 7.1 Caixa de diálogo da ferramenta para controle remoto do HP4145.

7.2.2 Ferramenta para controle remoto do medidor de R, L e C HP4284A.

O medidor de resistência (R), indutância (L) e capacitância (C) de alta precisão, modelo HP4284A, de fabricação da Hewllet-Packard, é muito utilizado na microeletrônica, principalmente para a caracterização de capacitores não encapsulados, que são muito úteis para a obtenção de parâmetros tecnológicos referentes a processos

de fabricação de circuitos integrados. O HP4284 pode também ser utilizado para realizar medidas em componentes discretos. Maiores detalhes sobre este equipamento podem ser encontrados no seu manual do usuário /HP89b/.

A ferramenta localmente implementada, além de permitir que todas as suas funções possam ser operadas remotamente e seus dados transferidos ao computador, estende as suas potencialidades, ao permitir que listas de valores de excitação sejam definidas (como listas de freqüências ou amplitudes para os sinais AC, ou valores para as tensões DC) e aplicadas ao dispositivo sob teste em intervalos de tempo regulares e pré-definidos pelo usuário. Estas listas não possuem limitação quanto ao número de componentes. Também é possível estabelecer-se critérios para estabilização dos valores medidos, que são verificados automaticamente pela ferramenta que o controla remotamente. Maiores detalhes encontram-se no manual do usuário da ferramenta para o controle remoto do HP4284.

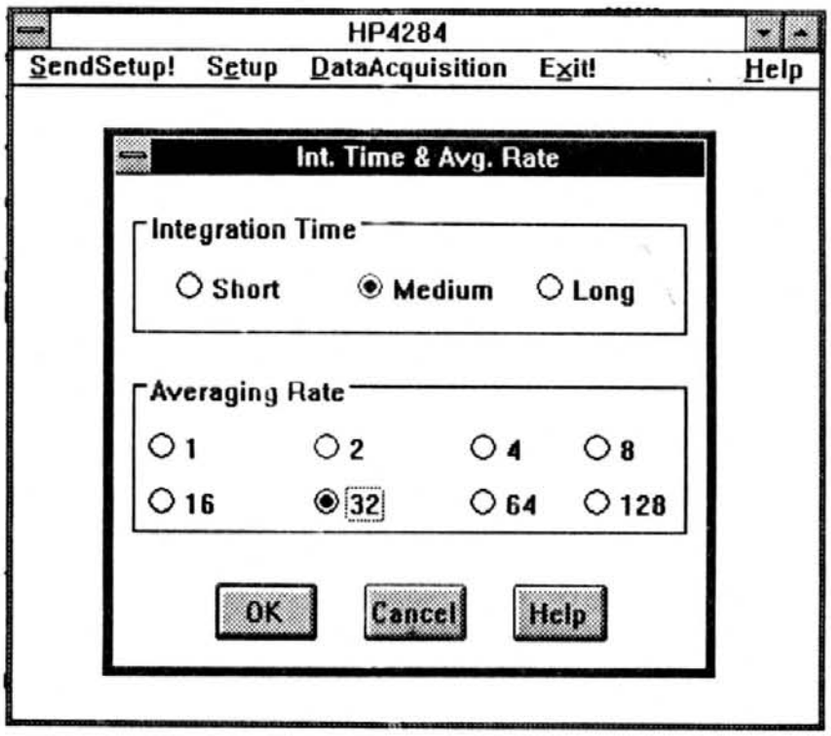


Fig. 7.2 Caixa de diálogo da ferramenta para controle do HP4284.

7.2.3 Ferramenta para controle remoto do osciloscópio Tek2440.

O osciloscópio digital Tektronix modelo 2440 possue dois canais com freqüência de amostragem de 500 MS/s (milhões de amostragens por segundo). Maiores detalhes sobre as suas características podem ser obtidas no manual fornecido pela Tektronix /TEK89/, no manual da ferramenta integrada que implementa seu controle remoto e aquisição de dados, ou através do auxílio ao usuário (*help on-line*) oferecido pela ferramenta. Esta ferramenta permite a aquisição de formas de onda, medida de tensões *rms*, mínima, máxima, pico-a-pico, média, tempos de subida e descida, período e freqüência de um sinal alternado, entre outros.

Uma das caixas de diálogo utilizadas para comunicação com o usuário pode ser vista na fig. 7.3.

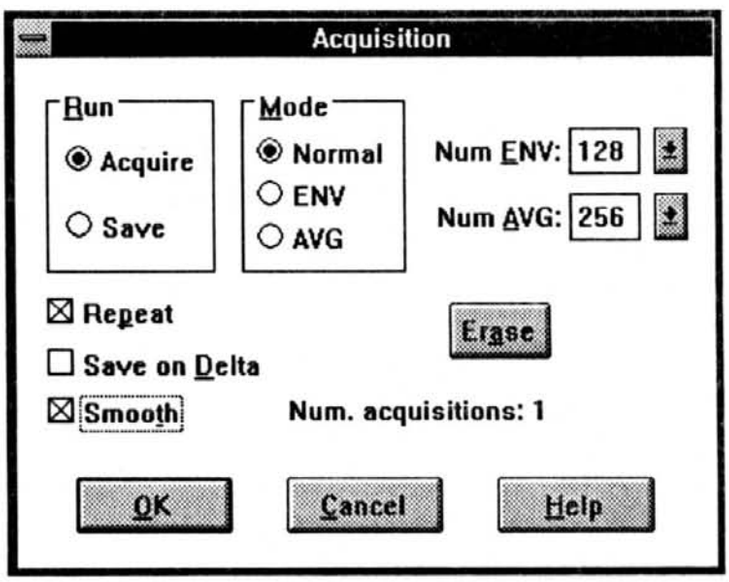


Fig. 7.3 Caixa de diálogo da ferramenta para controle do TEK2240.

7.2.4 Ferramenta para controle remoto do gerador de funções PFG5010.

Integrada no ambiente SGC também encontra-se uma ferramenta para o controle remoto do gerador de funções programável modelo PFG5105, produzido pela Tektronix. Esta ferramenta implementa o controle remoto de todas as facilidades oferecidas pelo PFG5105, permitindo assim a utilização nos procedimentos de teste e caracterização da geração das mais diversas formas de onda, pulsos e trens de pulsos. Uma descrição detalhada das facilidades oferecidas encontram-se no manual da

ferramenta, no manual fornecido pelo fabricante do equipamento /TEK88/, ou podem ser obtidas a partir do auxílio ao usuário (*help on-line*) oferecido pela ferramenta.

Na fig. 7.4 uma das caixas de diálogo oferecidas pela ferramenta pode ser visualizada.

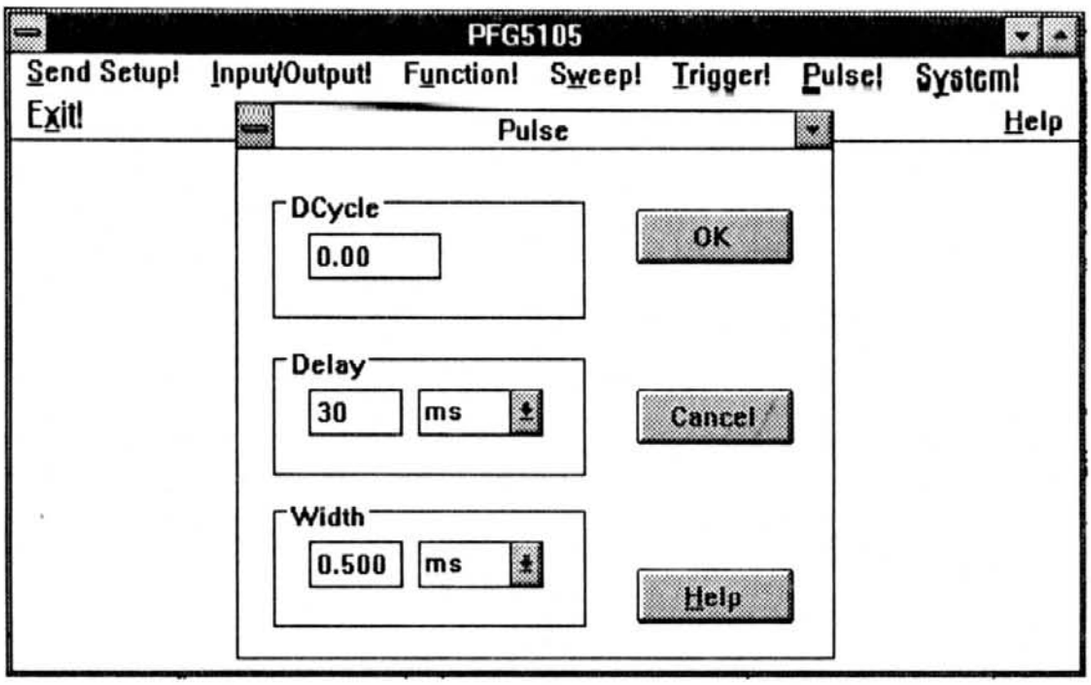


Fig. 7.4 Caixa de diálogo da ferramenta para controle remoto do PFG5105.

7.2.5 Ferramenta para controle remoto da fonte de tensão PS5010.

A ferramenta para o controle remoto da fonte de tensão programável Tektronix PS5010 foi desenvolvida localmente e permite a utilização automatizada desta fonte de tensão em procedimentos de caracterização e teste. A fonte de tensão PS5010 é uma fonte de tensão dual com 3 saídas disponíveis: i) saída positiva, de 0 a 32 Volts, com resolução de 0.01 V para tensões de até 10 V e 0.10 V acima de 10 V, com limite de corrente de 1.6 Ampéres para tensões abaixo de 15 V e 0.75 A na faixa acima de 15 V; ii) saída negativa, de 0 a -32 V, com resolução de 0.01 V até -10 V e 0.10 V abaixo de -10 V, com limite de corrente de 1.6 A para tensões na faixa de 0 a -15 V e 0.75 A na faixa acima de -15 V; iii) saída lógica, que fornece tensões de 4.5 a 5.5 V, com resolução de 0.010 V e limite de corrente de 3 A. Na fig. 7.5 podemos observar uma das caixas de diálogo utilizadas para o controle das facilidades deste equipamento.

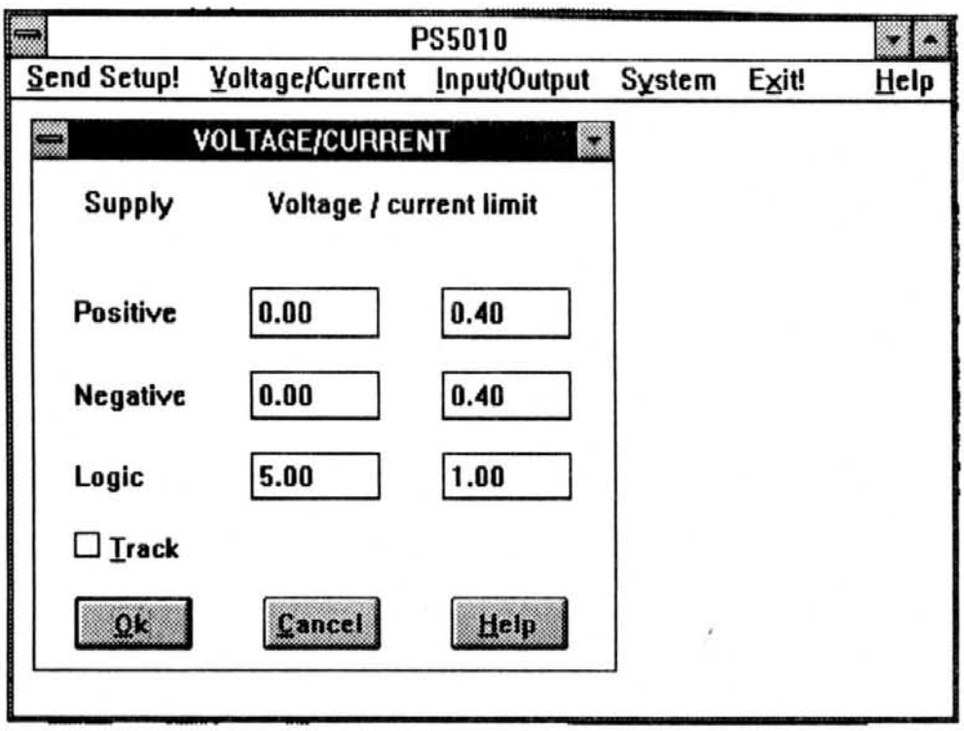


Fig. 7.5 Caixa de diálogo da ferramenta para controle remoto do PS5010.

7.2.6 Ferramenta para controle remoto do gerador de funções HP3325.

A ferramenta para controle remoto do gerador de funções HP3325 foi desenvolvida no Laboratório Central de Eletrotécnica e Eletrônica (LAC), do Departamento de Eletrônica da UFPR (convênio entre UFPR e COPEL-Companhia Paranaense de Energia), durante a instalação nesta instituição do ambiente SGC, com treinamento de usuários e projetistas de ferramentas. Esta ferramenta permite a utilização do HP3325 em diferentes procedimentos de teste e caracterização, sempre que for necessária a utilização de ondas senoidais, quadradas, triangulares, dente-de-serra positiva ou dente-de-serra negativa, em freqüências fixas ou que podem variar no tempo (*sweep*). Também é possível a geração de diversos tipos de pulsos e trens de pulsos. Uma das janelas disponíveis para a programação das facilidades oferecidas pela ferramenta pode ser vista na fig. 7.6.

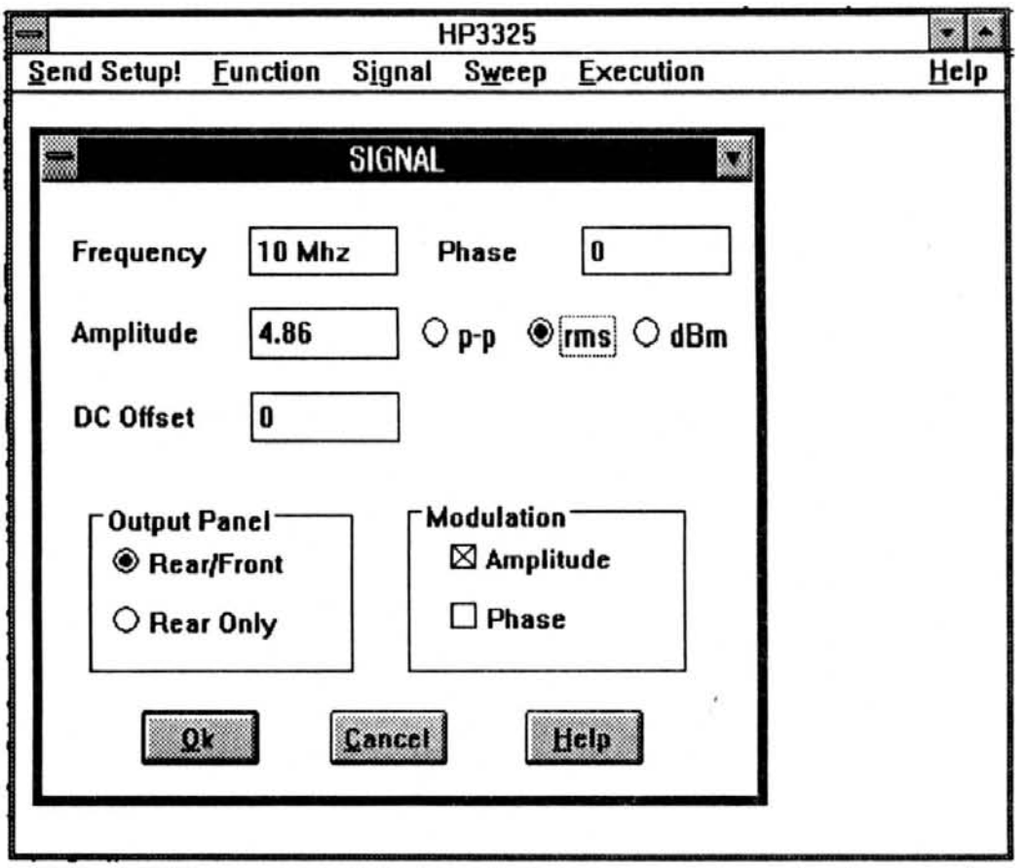


Fig. 7.6 Janela da Ferramenta Implementada (controle remoto do HP3325).

7.3 *Drivers* Integrados.

Ao ambiente SGC podem ser integrados programas que permitem o gerenciamento de dispositivos de entrada e saída, como o barramento GPIB ou a interface serial. No momento existem integrados ao SGC dois *drivers* para o barramento GPIB. Novos *drivers* para gerenciamento de outras interfaces de *hardware* podem ser integrados a qualquer momento.

7.3.1 *Drivers* para gerenciamento do barramento GPIB.

Encontram-se integrados no ambiente SGC *drivers* para o barramento GPIB. O primeiro compatível com o *hardware* produzido pela empresa STD - Sistema Técnicos e Digitais /STD88/, e o segundo compatível com o *hardware* para interfaceamento GPIB produzido pela empresa National Instruments Inc. /NAT90/. Estes *drivers* foram desenvolvidos em linguagem C++ utilizando-se o compilador BorlandC. O barramento GPIB encontra-se descrito em /NAT90/.

Estes *drivers* implementados podem ser utilizados por qualquer ferramenta que necessite comunicar-se com instrumentos remotos através do barramento GPIB, utilizando-se de uma das duas placas disponíveis. Para as ferramentas não há nenhuma diferença em relação a qual dos *drivers* está sendo utilizado. Para a ferramenta também não faz diferença o tipo de interface que está sendo utilizada. Poderia por exemplo, tratar-se de uma interface serial. Esta informação não está contida no código da ferramenta. Ela é configurada no momento da instalação no ambiente SGC.

Para que as ferramentas destinadas a controlar equipamentos remotos o façam através de um dos *drivers* GPIB disponíveis, basta que no momento da instalação da ferramenta, esta indique que deseja utilizar-se deste *driver*. A partir deste momento, o SGC repassará todas as mensagens destinadas a instrumentos remotos, bem como as solicitações de aquisições de dados, ao *driver* GPIB, que procederá o gerenciamento destas mensagens e do barramento.

7.4 Critérios para definição de ferramentas.

Conforme discutido no capítulo 3, as ferramentas integradas e a serem integradas no SGC podem ser divididas em 4 grupos. Esta divisão se faz de acordo com o conhecimento que estas representam. Cada ferramenta deve preocupar-se em atender uma necessidade específica da maneira mais flexível (genérica e completa) possível.

Assim, uma ferramenta para o controle de um equipamento remoto deve modelar este objeto e possibilitar que através dela possa-se dispor de todas as potencialidades que este oferece, sem impor-se nenhuma limitação no universo de aplicações, exceto aquelas inerentes a construção do objeto físico (equipamento remoto) que esta ferramenta representa. Ao implementar-se esta ferramenta, não deve-se acrescentar a ela nenhuma característica que não derive diretamente do objeto físico que ela representa, como por exemplo acrescentar-se a esta funções para visualização dos dados adquiridos. Esta visualização será realizada com o auxílio de uma ferramenta construída especialmente para tal fim. Esta ferramenta de visualização será também utilizada para visualizar-se dados gerados ou adquiridos por outras ferramentas. Quanto mais genéricas e completas forem as facilidades de visualização, mais úteis elas serão ao usuário, atendendo um número maior de necessidades.

Da mesma maneira, as ferramentas que implementam algoritmos de teste ou algoritmos de extração de parâmetros, devem preocupar-se apenas em modelar e

integrar ao SGC estas metodologias que elas representam, da melhor maneira possível, sem preocuparem-se com a aquisição ou exibição de dados.

Quando um conjunto de ferramentas assim implementadas é integrada em um ambiente que possui facilidades de intercomunicação entre ferramentas, então oferece ao usuário uma maneira simples e fácil de definir o fluxo de dados, e possue uma base de dados para o correto e seguro armazenamento destes, estas ferramentas podem ser utilizadas em conjunto, interativamente, atendendo então a todas as necessidades do usuário.

Estas ferramentas podem ser desenvolvidas em diferentes instituições de ensino, pesquisa, ou em indústrias, e integradas ao ambiente SGC. Ao instalar-se o SGC em um nova instituição, esta herdará todas as ferramentas nele já integradas, necessitando assim desenvolver apenas ferramentas para atender necessidades a ela particulares, evitando-se a duplicação de esforços. Esta filosofia já pode ser verificada na prática quando do desenvolvimento junto ao Laboratório Central de Eletrotécnica e Eletrônica (LAC), do Departamento de Eletrônica da UFPR (convênio entre UFPR e COPEL), do programa para o controle remoto de um gerador de funções (modelo HP3325). Esta ferramenta foi desenvolvida e integrada ao ambiente SGC com sucesso, estando agora disponível a todas aqueles que possuírem um HP3325 e desejarem utlilizá-lo em procedimentos automatizados de caracterização e teste. No decorrer do tempo, com uma maior difusão do ambiente SGC em outras intituições, o número de ferramentas integradas aumentará consideravelmente, tornando-se assim as facilidades oferecidas pelo SGC cada vez mais completas.

Como pode ser observado, o conjunto de ferramentas até o momento implementadas são ferramentas para o controle de instrumentos remotos e aquisição de dados. Optou-se por iniciar-se com a integração destas ferramentas, por ser a aquisição dos dados medidos por equipamentos remotos o ponto de partida para a maioria dos procedimentos de caracterização e também de teste. Após esta aquisição de dados estes podem ser processados e ou visualizados. Por exemplo, após realizadas medidas sobre transistores MOSFET, podemos utilizar um extrator de parâmetros SPICE para realizar tal extração. Em diferentes instituições de pesquisa pode-se divergir em relação aos diferentes métodos utilizados para extrair tais parâmetros, mas apesar disso, elas poderão utilizar as mesmas ferramentas para realizar medidas sobre os transistores, como é o caso da ferramenta para controle e aquisição de dados com o HP4145, equipamento disponível na UFRGS, UFPR, UFSC e CTI-Campinas, entre outros.

Para visualizar e realizar algum processamento mais rotineiro sobre os dados pode-se utilizar ferramentas comercialmente disponíveis, como a planilha de cálculo EXCEL /MIC92a/. A maioria dos programas atualmente desenvolvidos para o ambiente *MS-Windows* permite a troca de dados utilizando-se o padrão definido pelo DDE (*dinamic data exchange*) /MIC90/. Para que esta troca de dados seja possível, é necessário que o SGC implemente o protocolo particular utilizado por cada um destes programas para a troca de dados, como por exemplo o EXCEL.

O desenvolvimento de programas com as facilidades de visualização e processamento, com uma poderosa interface gráfica, como é o caso das planilhas de cálculo como EXCEL /MIC92a/, LOTUS /SOU89/, QuattroPro /OTT90/, e programas para tratamento matemático de dados como MATHEMATICA /CRA91/ e MATLAB /MOL86/, por exemplo, é uma tarefa que exige um esforço muito grande, além das capacidades de um pequeno grupo de trabalho. Muitos esforços já foram despendidos por empresas como Microsoft, Borland, Lotus, e MathWorks, entre outras, na criação de tais ferramentas de *software*. Deve-se procurar desenvolver ferramentas apenas para atender facilidades específicas, ainda não atendidas por ferramentas facilmente disponíveis. As demais facilidades poderão ser atendidas a partir de ferramentas integradas ao SGC através das facilidades descritas no capítulo 8.

7.5 Conclusão

As ferramentas e os *drivers* já integradas ao ambiente SGC foram apresentados, de maneira a prover uma descrição das facilidades que estas já integram ao ambiente SGC. Os critérios a serem utilizados no processo de definição de novas ferramentas e *drivers* a serem integradas foram discutidos, de maneira a obter-se um melhor aproveitamento dos esforços despendidos no desenvolvimento destes. Ao seguir-se estes critérios, um ambiente automatizado para teste e caracterização, poderoso e cooperativo, é obtido.

8 Gerenciamento, Integração e Encapsulamento de Ferramentas.

8.1 Introdução:

Em um laboratório de testes e/ou caracterização automatizados, freqüentemente novas ferramentas fazem-se necessárias. Isto ocorre, por exemplo, quando novos equipamentos de medida são adquiridos, quando novos algoritmos de teste/caracterização tornam-se necessários, ou quando deseja-se que novas facilidades de visualização sejam incorporadas ao ambiente. Muitas vezes também é necessário proceder-se a substituição de ferramentas já acopladas ao sistema por uma nova versão da mesma, na qual novas características são acrescentadas ou eventuais problemas são solucionados.

Por outro lado, também é necessário permitir-se o acesso rápido e fácil às ferramentas integradas por parte do engenheiro de caracterização e teste. Estas ferramentas integradas também necessitam ser gerenciadas, isto é, as informações relevantes a elas relacionadas devem estar disponíveis para consulta e atualização.

Neste capítulo se discute os mecanismos oferecidos pelo SGC para que se possa realizar a substituição ou acoplamento de novas ferramentas fácil e rapidamente, sem que problemas de compatibilidade com os demais módulos do sistema ocorram. O projetista de ferramentas de *software* também é auxiliado pelo SGC na tarefa de desenvolver tais ferramentas, através dos esqueletos de ferramenta oferecidos. Estes esqueletos são apresentados e discutidos neste capítulo. O engenheiro de caracterização e teste é auxiliado na utilização e no gerenciamento das ferramentas através dos mecanismos que aqui serão descritos.

8.2 O módulo de gerenciamento das ferramentas.

Quando não é fornecido ao usuário um meio adequado de se proceder o gerenciamento das ferramentas, até mesmo a tarefa de executar uma ferramenta com sucesso pode vir a ser um processo complexo, se o usuário necessitar conhecer muitos detalhes a respeito da ferramenta, como a sua localização no sistema de arquivos, os dados de ambiente que ela necessita (como endereço GPIB e localização de arquivos de configuração), e a necessidade de executar conversores de formatos de dados, entre outros.

No ambiente SGC todas estas informações são controladas pelo módulo de gerenciamento das ferramentas, no qual permite-se o acoplamento de ferramentas em dois níveis: Integração e Encapsulamento. Estas duas modalidades de acoplamento são discutidas nos ítens 8.4 e 8.5 abaixo, respectivamente.

Para integrar ou encapsular uma ferramenta corretamente, os seguintes dados são necessários /BAR92/ e devem ser gerenciados adequadamente:

i) Nome da ferramenta. O usuário pode definir o nome a ser utilizado para identificar a ferramenta dentro do ambiente SGC. Este nome não necessita coincidir, por exemplo, com o nome do módulo executável principal da ferramenta.

ii) Versão da ferramenta. A versão da ferramenta poderá ser consultada a partir do menu "*about*" existente em todas as ferramentas, já que a estrutura deste é fornecida com o esqueleto de ferramenta (conforme descrito no item 8.3).

iii) Localização física da ferramenta. O usuário não necessita conhecer a localização física da ferramenta para executá-la rotineiramente. Ele apenas necessita selecioná-la a partir do menu de ferramentas do SGC. Esta informação é necessária apenas no momento do acoplamento da ferramenta (conforme fig. 8.2).

iv) Ícone a ser utilizado na interface gráfica do SGC. As ferramentas podem ser identificadas a partir de ícones.

v) Argumentos a serem passados quando da execução da ferramenta. Para ferramentas acopladas não é necessário a passagem de nenhum tipo de argumento quando da execução da ferramenta. Os argumentos eventualmente necessários são definidos quando do acoplamento da ferramenta e por estas recuperados no momento da execução, através das funções específicas fornecidas no esqueleto de ferramenta descrito no item 8.3. A definição da transferência de objetos (fluxo de dados) é realizada a partir de interfaces gráficas, conforme descrito no capítulo 4 e no item 8.4 abaixo.

vi) Informações de auxílio (*Help*). Todas as ferramentas possuem auxílio sensível a contexto (*help on-line*), que auxiliam o usuário na correta utilização da ferramenta, conforme exemplificado na fig. 8.1. As facilidades necessárias a construção deste auxílio ao usuário são fornecidas junto ao esqueleto de ferramenta, conforme descrito no item 8.4.

vii) *Hardware* necessário para a execução da ferramenta. As informações a respeito do *hardware* necessário para a execução de uma ferramenta também podem

ser obtidas a partir do menu "*about*" ou do histórico da própria ferramenta. O histórico da ferramenta está descrito no item 6.3.14.

viii) Dados de entrada aceitos pela ferramenta. Conforme descrito nos capítulos 5 e 6, o SGC gerencia o fluxo de dados entre ferramentas, permitindo que sejam transferidos a uma ferramenta apenas objetos que esta pode tratar. Além disso, o SGC permite a navegação através da base de dados, a fim de selecionar-se os objetos que se deseja transferir, conforme descrito no item 4.3.2.2.

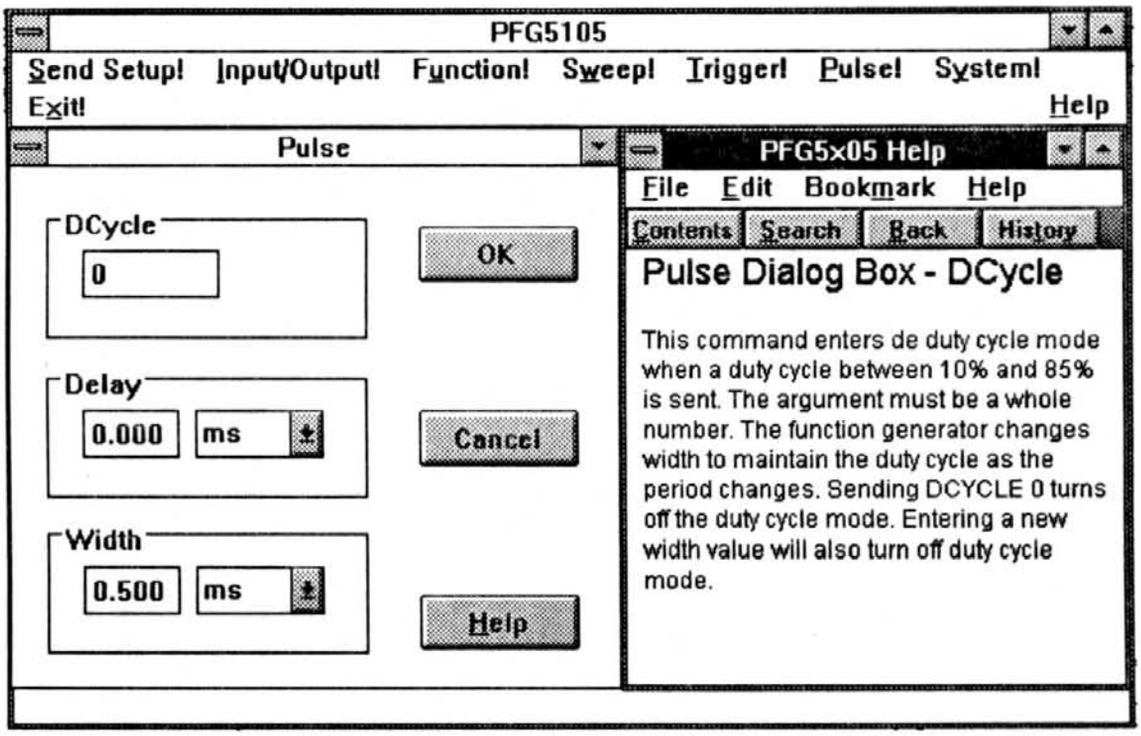


Fig. 8.1 Exemplo de janela de auxílio ao usuário.

ix) Dados de saída gerados pela ferramenta. O SGC encarrega-se de armazenar os dados gerados pelas ferramentas a ele acoplados na base de dados automaticamente, documentando-os, conforme descrito nos capítulos 4 e 6.

x) Condições a serem satisfeitas antes da execução da ferramenta. Antes da execução de uma ferramenta verifica-se se todos os dados que a ferramenta irá utilizar durante a execução estão presentes. Isto é particularmente importante quando da execução de uma ferramenta dentro de um procedimento, quando os dados que a ferramenta irá utilizar podem ser os gerados em passos anteriores do procedimento, por

outras ferramentas. Isto também é realizado no caso da utilização de arquivos de configuração (geralmente para ferramentas encapsuladas).

xi) Condição gerada após a execução, de forma a identificar se a execução foi bem sucedida ou não.

Além do gerenciamento destas informações que caracterizam a ferramenta, o SGC também realiza as seguintes tarefas:

xii) Executa os conversores de formato de dados necessários, para o caso de ferramentas encapsuladas. Estes conversores são executados antes da execução da ferramenta propriamente dita, para extrair os dados da base de dados e repassá-los a ferramenta, e após a execução da ferramenta, para possibilitar o armazenamento dos dados gerados na base de dados. O mecanismo implementado encontra-se descrito no item 8.4 abaixo.

xiii) Verifica a eventual necessidade de utilização de *drivers* para a comunicação através das interfaces de *hardware* oferecidas, gerenciando as informações necessárias e repassando-as aos *drivers* necessários. Graças a este mecanismo é possível, por exemplo: i) evitar que o mesmo endereço de barramento seja utilizado por mais de um equipamento; ii) fazer com que um procedimento continue válido quando o endereço no barramento de algum equipamento mudar; iii) em caso de laboratórios que possuem o mesmo equipamento, mas configurados com endereço de barramento diferente, permite-se que o mesmo procedimento possa ser executado nos dois laboratórios, sem alterações; iv) quando da troca do endereço no barramento de algum equipamento, não serão necessárias modificações no código fonte da ferramenta (apenas é necessário informar ao SGC o novo endereço do equipamento); e (vi) evita-se que as mesmas particularidades de uma determinada interface sejam tratadas em todas as ferramentas.

xiv) Realiza o gerenciamento das informações referentes ao histórico da ferramenta. A fim de auxiliar o usuário nas tarefas rotineiras ao gerenciamento das informações de caráter genérico referentes a ferramenta, como manutenção e informações relativas ao licenciamento da ferramenta, entre outras, o SGC mantém um histórico das informações e eventos mais significativos a esta relacionados, que pode ser visualizado e editado através de uma interface gráfica..

Ao realizar este gerenciamento, o SGC alivia o usuário da necessidade de conhecer um número excessivo de detalhes para a correta utilização das ferramentas. Para ter acesso às ferramentas, tanto para a utilização destas em procedimentos como para executá-las isoladamente, basta que o usuário as selecione com o *mouse*, a partir do

menu de ferramentas do SGC. O SGC automaticamente verifica todas as condições acima apontadas e procede a chamada da ferramenta de maneira adequeda, utilizando-se das funções de intercomunicação descritas no capítulo 5, das facilidades oferecidas pela base de dados descrita no capítulo 6, ou das facilidades implementadas para encapsulamento de ferramentas, como descrito no item 8.4.

Para integrar novas ferramentas ao sistema, o usuário conta com o auxílio de facilidades como a ilustrada pela caixa de diálogo mostrada na fig. 8.2, que permite a definição de alguns dos parâmetros acima citados. Outros parâmetros, como a configuração de endereço de equipamentos no barramento GPIB, por exemplo, são definidos pelo *driver* que fará o gerenciamento desta interface de *hardware*.

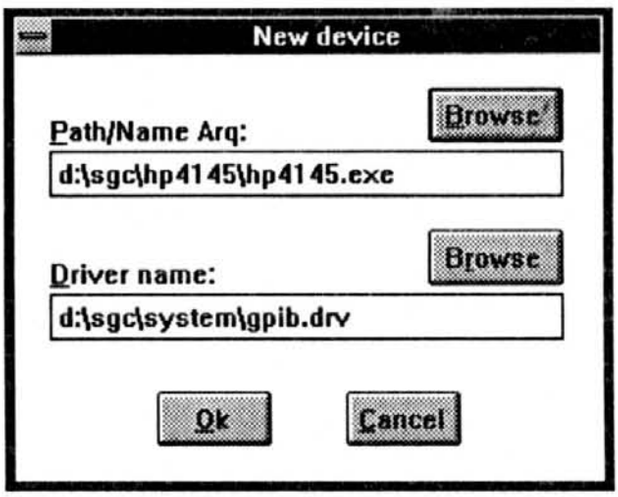


Fig. 8.2 Caixa de diálogo para instalação de nova ferramenta.

Depois da conexão de uma ferramenta ao ambiente SGC, a sua configuração poderá ser modificada a qualquer momento (*driver* utilizado e endereço do equipamento no barramento, localização física da ferramenta, ícone a ser utilizado, nome a ser mostrado no menu de ferramentas, entre outros). Para evitar que usuários menos experimentados ou não autorizados modifiquem estes dados relativos às ferramentas, o SGC implementa um cadastro de usuários com 3 diferentes níveis de acesso: i) usuários comuns; ii) gerentes de grupos e (iii) super-usuários. Os usuários comuns apenas poderão utilizar as ferramentas, editar e executar procedimentos, e navegar através da base de dados, editando ou excluindo os dados por eles gerados. Os gerentes de grupo, além das atividades previstas para os usuários comuns, poderão editar ou excluir os dados gerados por qualquer membro do grupo do qual são gerentes, além de poderem

cadastrar novos membros (usuários comuns) para o grupo do qual são gerentes. Os super-usuários, além de poderem realizar todas as atividades previstas para os gerentes de grupo, poderão criar novos grupos, cadastrar ou modificar as senhas de novos gerentes de qualquer grupo ou outros super-usuários, terão pleno acesso a todos os dados e operações da base de dados e poderão modificar as configurações do ambiente e das ferramentas, poderão instalar novas ferramentas ou *drivers*, bem como desconectar ferramentas ou *drivers* previamente conectadas. A janela de *login* do SGC está mostrada na fig. 8.3.

Não existe limitação para o número de gerentes de um mesmo grupo, como também não existe limitação para o número de super-usuários que podem estar cadastrados no sistema.

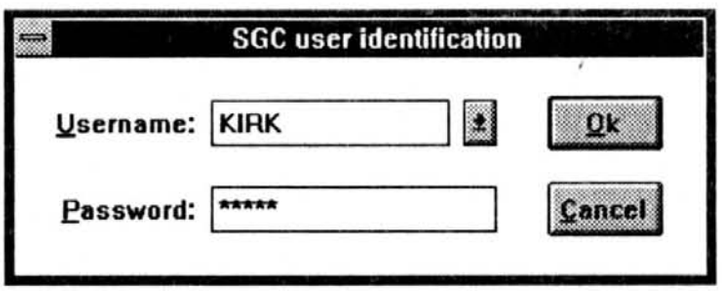


Fig. 8.3 Janela para a identificação de usuário (*login*).

Este sistema de cadastramento dos usuários também é importante para que se possa identificar os usuários que editaram ou executaram procedimentos, documentando-os, conforme descrito no capítulo 6.

Quando da execução de ferramentas encapsuladas (*black box*), o usuário será auxiliado na definição dos parâmetros necessários através de caixas de diálogo gerenciadas pelo SGC. Estas caixas de diálogo realizarão a interface entre a ferramenta e o usuário, conforme descrito no item 8.4, evitando que seja necessária a busca manual de dados da base de dados e a posterior execução também manual de conversores.

Quando o usuário necessita utilizar ferramentas ainda não acopladas ao SGC (nem integradas, nem encapsuladas), ele poderá utilizar-se da caixa de diálogo mostrada na fig. 8.4. Nesta figura podemos ver uma caixa de diálogo que pode ser utilizada para o interfaceamento com qualquer tipo de ferramenta ainda não acoplada, já que esta caixa de diálogo não foi desenvolvida tendo uma ferramenta em especial como

alvo. Ao realizar-se o acoplamento (integração *Black Box*), estas caixas de diálogo incluirão outras opções, como a possibilidade de indicar-se com o *mouse* o tratamento que a ferramenta deve destinar aos dados passados, sendo a linha de comando automaticamente construída pelo SGC. A caixa de diálogo da fig. 8.4 apenas auxilia na localização da ferramenta e dos dados a serem a ela transferidos, além é claro, de documentar e permitir a repetição do passo com ela definido, na medida do possível.

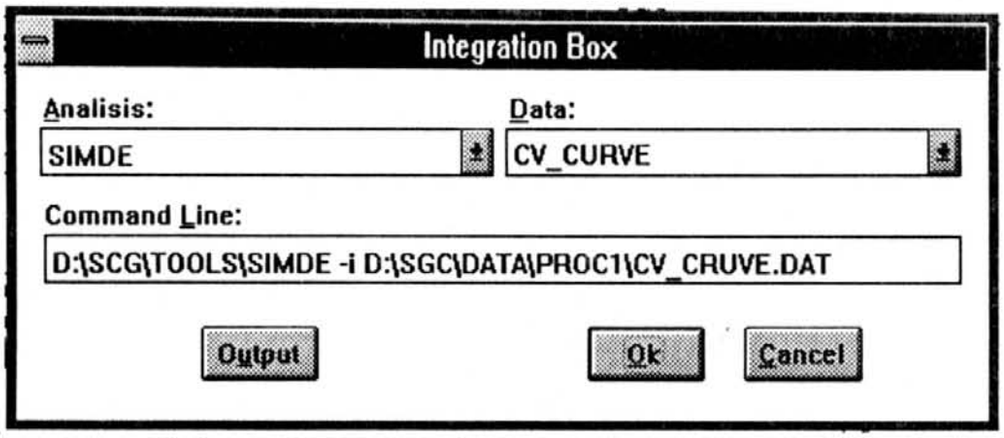


Fig. 8.4 Caixa de diálogo para interfaceamento com ferramentas não acopladas.

8.3 Integração de ferramentas (*White Box*).

A integração *white box* é o método mais aconselhado para o acoplamento de ferramentas ao ambiente SGC. As ferramentas desta maneira acopladas utilizam as facilidades oferecidas pelo ambiente SGC para o acesso à base de dados, às interface de *hardware*, e para comunicação com outras ferramentas, entre outros. Este tipo de integração é possível de ser realizado com ferramentas desenvolvidas de acordo com os padrões adotados pelo ambiente SGC, ou com ferramentas das quais esteja disponível o código fonte, que posteriormente é modificado, a fim de obedecer o padrão SGC.

O acoplamento destas ferramentas é realizado conforme descrito no item anterior. Já para o desenvolvimento das mesmas, um esqueleto de ferramenta é oferecido. Este esqueleto ajuda no desenvolvimento destas ferramentas, ao fornecer as funções básicas para o acesso às facilidades oferecidas pelo ambiente SGC, além da estrutura básica de um programa *Windows*. O esqueleto de ferramenta oferece as classes básicas para:

i) As estruturas de janela básicas para um bom programa *Windows*. Estas estruturas de janelas representam os módulos fundamentais, a partir do qual a nova ferramenta será construída /BOR91/.

ii) A implementação das funções de rechamada (*callback functions*) /BOR91/ e demais funções para o protocolo de comunicação com o SGC via DDE /MIC90/. Estas são as funções utilizadas para implementar a troca de dados e comandos entre os diversos módulos do sistema SGC.

iii) Funções para armazenamento das configurações realizadas pelo usuário em arquivo (.INI), da sua leitura quando da inicialização da ferramenta, e funções para o ajuste automático da ferramenta de acordo com estas configurações.

iv) Funções para manipulação de erros de maneira padrão.

v) Exemplos de caixas de diálogo para a construção da interface com o usuário. A utilização destes exemplos ajuda a construir-se a interface de usuário padronizada.

vi) Estrutura básica para a implementação do auxílio ao usuário sensível a contexto (*help on-line*).

A utilização deste esqueleto também facilitará uma eventual manutenção posterior de ferramentas, caso alguma modificação nos mecanismos de conexão entre o SGC e as ferramentas torne-se necessária. Este esqueleto, que consiste em um módulo a ser ligado (*to be linked*) à ferramenta, isola estas funções de conexão do restante do código da ferramenta.

Estes esqueletos estão disponíveis como código fonte para todos aqueles que desejarem implementar ferramentas utilizando o compilador *Borland C++*. Aqueles que desejarem implementar ferramentas utilizando esta linguagem de programação poderão ligar (*to link*) os módulos objeto gerados (.OBJ) ao código objeto da ferramenta. Aqueles que desejarem utilizar outros compiladores deverão implementar as funções acima descritas na linguagem de programação escolhida, com orientação a objetos ou não (em forma de classes, funções ou procedimentos).

8.4 Encapsulamento de ferramentas (*Black Box*).

Este tipo de acoplamento é realizado com ferramentas que foram desenvolvidas sem o conhecimento dos padrões adotados pelo SGC, e das quais não se

dispõem do código fonte, ou, mesmo tendo-se o código fonte, a integração *white box* não traria uma boa relação custo/benefício.

O encapsulamento das ferramentas é realizado com o auxílio de dois programas. O primeiro, que chamaremos de *ToolPreProcessor*, é executado antes da ferramenta, e o segundo, chamado *ToolPostProcessor*, que é executado após a ferramenta. Assim, toda vez que o usuário ativa uma ferramenta encapsulada a partir do menu de ferramentas, três programas são executados, na seguinte ordem: o *ToolPreProcessor*, a Ferramenta, e finalmente o *ToolPostProcessor*.

Para cada Ferramenta encapsulada um *ToolPreProcessor* e um *ToolPostProcessor* devem ser definidos.

O *ToolPreProcessor* permite ao usuário a definição e/ou busca na base de dados dos dados de entrada da Ferramenta, de forma a verificar se os dados de entrada correspondem aos tipos aceitos pela ferramenta. Como estes dados podem ser dados que ainda não estão disponíveis na base de dados (como dados gerados por atividades do mesmo procedimento), o *ToolPreProcessor* monta a seqüência de eventos necessários à extração destes dados da base de dados e sua armazenagem em arquivos que serão acessados pela ferramenta (indicando a necessidade ou não da execução de conversores). Esta seqüência de atividades só é executada quando da execução do procedimento, ou quando a ferramenta é executada isoladamente (*stand alone*). Além disto, o *ToolPreProcessor* constrói a linha de comando, que será usada para executar a ferramenta em questão.

Caso a Ferramenta interaja necessariamente com o usuário durante a execução de um procedimento (como para entrada de parâmetros que não podem ser passados pela linha de comando), esta interação será feita durante a execução do procedimento, no momento em que a ferramenta é executada. Esta interação direta entre ferramenta encapsulada e usuário é extremamente negativa, porque faz necessária a suspensão da execução do procedimento até que o usuário informe os dados necessários, e faz com que esta execução não seja totalmente documentada (e automaticamente repetível), porque estes dados informados durante a interação direta do usuário com a ferramenta encapsulada não estarão armazenados na base de dados.

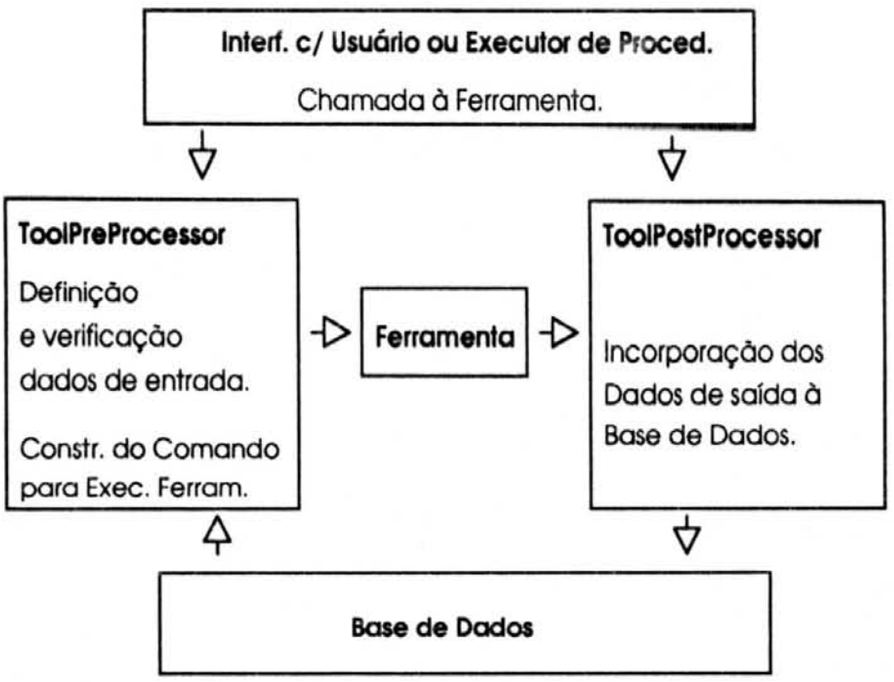


Fig. 8.5 Modelo para integração Black Box (encapsulamento).

O *ToolPostProcessor* verifica se a execução foi bem sucedida e incorpora os dados gerados pela ferramenta encapsulada à base de dados, da mesma maneira como acontece com as ferramentas totalmente integradas. As tarefas necessárias (como eventual conversão de arquivos) são automaticamente por ele programadas quando da definição do passo, e realizadas quando da execução do procedimento.

Na Fig. 8.5 os passos que ocorrem durante a execução de uma ferramenta encapsulada estão esquematizados.

8.5 Integração de *drivers*.

O SGC também permite a integração de novos *drivers* que realizarão a interface com novos dispositivos de *hardware* instalados.

Para facilitar o desenvolvimento de *drivers*, um esqueleto de *driver* é fornecido. O esqueleto de *driver* deve ser utilizado quando do desenvolvimento de *drivers* para interfaces de *hardware* ainda não suportadas pelo SGC. Ele consiste da estrutura básica (classes básicas) de um *driver* para o *Windows 3.1* /MIC90/ e das funções necessárias à sua conexão aos demais módulos do ambiente SGC, como por exemplo:

i) Implementação das funções de rechamada especificas aos *drivers*. Estas são as funções que implementam a troca de dados e comandos entre o *driver* e o SGC. Não existe comunicação direta entre as ferramentas e os *drivers*. Ela é realizada sempre através das facilidades oferecidas pelo SGC (como descrito no capítulo 4).

ii) Implementação do armazenamento dos dados necessários a inicialização em arquivo (.INI), obedecendo os padrões *Windows* /MIC90/. Todos os *drivers* integrados no ambiente SGC devem salvar as informações de configuração particulares em arquivo, de forma padronizada, e ajustarem automaticamente a sua configuração quando da sua inicialização.

iii) Implementação das funções necessárias durante o procedimento de configuração. Estas funções implementam a configuração e incialização do *driver* de maneira padronizada.

iv) Exemplos de resposta a mensagens de maneira padronizada e tratamento de erros. As mensagens trocadas entre o SGC e os *drivers* devem obedecer a formatos padronizados, definidos no âmbito do SGC, bem como o tratamento de eventuais erros. Os exemplos fornecidos auxiliam na implementação destas facilidades.

8.6 Conclusão.

O SGC permite que o usuário tenha acesso as ferramentas de forma padronizada, escondendo deste grande parte dos detalhes necessários a este acesso, facilitando assim a sua utilização em procedimentos, em conjunto com outras ferramentas, ou de maneira isolada (*stand alone*).

Ao projetista de ferramentas ou *drivers* para teste e caracterização automatizados, o SGC oferece esqueletos que auxiliam no desenvolvimento, economizando esforços, além de garantir compatibilidade e facilidade de manutenção.

Também é possível acomodar ferramentas não desenvolvidas de acordo com os padrões estabelecidos pelo ambiente SGC (*foreign tools*), de maneira padronizada e a fim de minimizar o impacto causado pelo seu comportamento não padronizado sobre a repetividade e documentação de procedimentos, da interface com o usuário, e sobre o padrão de armazenamento de dados.

O usuário é auxiliado nas tarefas rotineiras necessárias ao gerenciamento das ferramentas, como o acesso e a atualização dos dados referentes a manutenção e utilização das ferramentas.

9 TRABALHOS FUTUROS.

9.1 Introdução.

Devido a sua natureza, um sistema de *software* jamais poderá ser considerado acabado, pronto, em seu estado definitivo. Com o passar do tempo, novas características se tornam necessárias e algumas das já implementadas necessitam ser reformuladas. Isto ocorre porque tanto um pequeno programa computacional como um complexo sistema de *software* e/ou *hardware* modelam, quando bem sucedidos, o conhecimento no momento aceito como o mais adequado para atender a uma determinada necessidade, conjugado com a quantidade de esforço e tempo à disposição para atendê-la. Tanto o conhecimento evolui, como esforços são somados, a fim de que melhor se possa atingir os objetivos propostos, como também os objetivos a serem atingidos não são imutáveis, podendo também estes sofrerem alterações.

O SGC não é um sistema acabado. O seu estado atual é o resultado do esforço até agora realizado, limitado por condições intelectuais (conhecimento de seus autores e número de pesquisadores envolvidos) e materiais (recursos e tempo disponível). Mas, para que se possa validar este esforço e obter-se um retorno do significado do SGC para outros pesquisadores, retorno este indispensável para que possa-se obter um melhor resultado final, é preciso que se tome o produto dos esforços até este momento despendidos e o transforme em uma versão que possa ser utilizada por terceiros. É necessário que se reporte as experiências até aquele momento adquiridas. A versão do SGC e as experiências aqui relatadas são uma fotografia do atual estágio do SGC, que espera-se, continue sempre evoluindo.

É neste espírito que neste capítulo são propostas novas atividades a serem realizadas no âmbito do SGC.

9.2 Definição de classes genéricas.

A definição de classes genéricas para ferramentas visa atingir a dois objetivos básicos:

i) facilitar o desenvolvimento de *software* e maximizar os esforços despendidos neste desenvolvimento.

ii) permitir que procedimentos sejam repetidos em diferentes instituições, mesmo quando os equipamentos e/ou ferramentas nelas existentes não são exatamente os

mesmos, mas possuam equivalência funcional, do ponto de vista do procedimento em questão.

Para que melhor se possa compreender o que se entende por classes genéricas, é mais adequado que estas sejam discutidas com o auxílio de um exemplo concreto. Um equipamento normalmente disponível e muito utilizado nas mais diversas instituições que desenvolvem atividades relacionadas ao teste e a caracterização é um osciloscópio. Existem osciloscópios fornecidos pelos mais diversos fabricantes, cada um destes oferecendo uma ampla gama de modelos diferentes. Todos eles implementam a mesma função básica: a aquisição de curvas de tensão *versus* tempo. Mas as especificações destes equipamento podem variar muito, eles podem diferir quanto a: i) o número de canais disponíveis para a aquisição destas curvas; ii) a taxa de amostragem do sinal (amostras por segundo), no caso dos osciloscópios digitais; iii) pode-se oferecer ou não facilidades como medida de tensão pico-a-pico, *rms* e média, período de oscilação e tempo de subida; iv) a impedância das ponteiras utilizadas para medida; v) a precisão com que a tensão do sinal amostrado pode ser medida, entre outros.

Apesar destas diferenças, eles possuem muitas coisas em comum, como definição de escalas de tensão para a aquisição da forma de onda, definição do período de varredura, escolha do canal a partir do qual a forma de onda deve ser adquirida, definição da fonte e nível de *trigger*, e solicitação para que uma forma de onda seja adquirida.

O mesmo ocorre no caso de geradores de função, fontes de tensão, multímetros e outros.

Quando procedimentos definidos para um dado modelo de um determinado fabricante necessitam ser executados em locais diferentes, se não existir uma uniformidade na estruturação dos dados que representam o passo para este equipamento utilizado, não será possível repetí-lo sem que este procedimento seja novamente editado, mesmo que o equipamento existente nesta outra instituição seja funcionalmente equivalente ao utilizado na instituição onde o procedimento foi originalmente definido.

No exemplo do osciloscópio, se definirmos um procedimento a partir de uma ferramenta que controla remotamente um osciloscópio Tektronix, e desejarmos repetí-lo agora utilizando um osciloscópio HP, isto será possível se e somente se as funções executadas com o osciloscópio Tektronix também puderem ser executadas com o osciloscópio HP e a ferramenta que controla o osciloscópio HP for capaz de

interpretar os passos gerados a partir da ferramenta que controla o osciloscópio Tektronix. Com a definição de uma classe osciloscópio genérica que modela as facilidades comuns a todos os osciloscópios, e armazena os dados que identificam o passo de uma maneira padronizada, será possível atingir tal objetivo. As funções básicas necessárias, como freqüência de operação (varredura), nível de *trigger* e escala de tensão requeridas serão verificadas por esta classe osciloscópio genérica, para garantir-se que as tarefas solicitadas serão adequadamente realizadas. Quando os passos programados exigirem facilidades além das capacidades da ferramenta que irá executar o passo, como a amostragem do sinal em uma taxa maior que a oferecida pelo instrumento remoto, o usuário será alertado e o procedimento não é executado.

Esta utilização de classes genéricas também facilitará em muito o desenvolvimento das ferramentas. As características comuns destes equipamentos serão modeladas nestas classes genéricas. A partir do princípio de herança /CHA92/, novas classes poderão ser derivadas a partir das classes genéricas, para tratar das características específicas a um equipamento (ou ferramenta) em particular. Como exemplo poder-se-ia criar uma classe genérica osciloscópio, que englobaria facilidades como interface com o usuário, armazenamento e recuperação dos dados que caracterizam o passo, facilidades para definição das escalas de tensão e tempo a utilizar, escolha do canal a ser utilizado para aquisição, definição da fonte e nível de *trigger* e solicitação de aquisição da forma de onda, entre outras. Assim, estas ferramentas não necessitarão mais ser programadas a partir do princípio, mas partir-se-á de uma classe que já oferece grande parte das funções necessárias. Uma filosofia semelhante já é hoje utilizada com sucesso, implementado as classes que oferecem os mecanismo de intercomunicação e acesso a base de dados, por exemplo.

9.3 Desenvolvimento de novas ferramentas.

Um dos pontos fundamentais para o sucesso do SGC como ambiente de teste e caracterização automatizado é a necessidade de se oferecer as ferramentas necessárias a execução dos procedimentos. Conforme descrito no capítulo 7, o conjunto de ferramentas necessárias é variado e complexo. A situação ideal seria aquela em que para cada equipamento remoto utilizado houvesse uma ferramenta de *software* para controlá-lo e a partir dele gerar ou adquirir dados, para cada algoritmo de análise de dados ou extração de parâmetros houvesse uma ferramenta que o implementasse, para cada necessidade de visualização de dados houvesse uma ferramenta apropriada, para cada metodologia de geração de padrões de teste houvesse uma ferramenta que a

implementasse. Infelizmente esta é uma situação difícil de ser atingida. Existe uma grande diversidade de equipamentos remotos utilizados em procedimentos de caracterização e teste, e novos equipamentos são produzidos continuamente. As metodologias de extração de parâmetros são continuamente objeto de estudos, o mesmo ocorrendo com a problemática de geração de padrões de teste para dispositivos, fazendo com que a implementação de novos algoritmos seja necessária. Assim, pode-se afirmar que uma tarefa que sempre merecerá atenção é o desenvolvimento destas ferramentas.

Trabalhos estão sendo e continuarão a ser realizados no sentido de prover-se um conjunto adequado de ferramentas para atender as necessidades comuns a maioria dos procedimentos de caracterização e teste realizados no âmbito da microeletrônica. Quanto maior o número de ferramentas integradas, maior será a potencialidade do ambiente SGC.

Brevemente este curso de pós-graduação receberá um novo analisador lógico. Então será necessário o desenvolvimento de uma ferramenta de *software* para controlar remotamente todas as suas facilidades, a fim de que se possa utilizá-lo adequadamente no teste dos circuitos integrados localmente projetados.

Graças a metodologia de desenvolvimento e integração de ferramentas adotada, novas ferramentas podem ser continuamente integradas ao ambiente SGC. Outras instituições também estão auxiliando no desenvolvimento e integração de ferramentas ao ambiente SGC, como é o caso do LAC - Laboratório Central de Eletrotécnica e Eletrônica, Convênio entre Universidade Federal do Paraná e Companhia Paranaense de Energia (COPEL), de Curitiba-PR.

9.4 Instalação do ambiente SGC em diversos laboratórios.

No momento o ambiente SGC está sendo utilizado rotineiramente no laboratório de microeletrônica deste curso de pós-graduação, bem como no Laboratório Central de Eletrotécnica e Eletrônica, LAC, onde foi instalado no final do mês de agosto do corrente.

Contatos já foram realizados para a instalação do SGC também junto à Universidade Federal de Santa Catarina (UFSC) (no laboratório de instrumentação eletrônica - LINSE, e no laboratório de máquinas elétricas e eletrônica de potência - LAMEP).

Durante o IX Congresso da Sociedade Brasileira de Microeletrônica (SBMicro), realizado de 08 a 12 de agosto do corrente, no Rio de Janeiro-RJ, com apresentação de artigo científico /WIR94/ e demonstração do SGC, também fez-se contato com pesquisadores da universidade estadual de Campinas (UNICAMP) e do Centro de Tecnologia para Informática (CTI), situado em Campinas, para uma eventual instalação do ambiente SGC junto a estas instituições. Os detalhes ainda pendentes para viabilizar esta instalação devem ser solucionados em breve, através de novos contatos.

A instalação do ambiente SGC em outras instituições é muito importante para validar o trabalho até o momento realizado e para que se possa somar experiências adquiridas com a sua utilização, o que possibilitará um aperfeiçoamento do sistema, além de permitir que outros pesquisadores possam usufruir do trabalho já realizado.

9.5 Conexão com ferramentas para ambiente *Windows* "estrangeiras".

Existem muitas ferramentas desenvolvidas para o ambiente *Windows* e que suportam algum tipo de conexão com outras ferramentas. Entre elas podemos citar planilhas de cálculo, ferramentas para processamento matemático e estatístico de dados e poderosas ferramentas para visualização de dados.

Esta conexão geralmente é implementada via DDE (*dinamic data exchange*) ou OLE (*object linking and embedding*) /MIC90/.

Através destas facilidades de interconexão é possível permitir que ferramentas não desenvolvidas para o ambiente SGC sejam rotineiramente utilizadas em procedimentos de caracterização e teste, em conjunto com as demais ferramentas oferecidas pelo SGC. Se estas ferramentas forem adequadamente conectadas ao ambiente SGC, é possível utilizá-las para a definição de passos da mesma maneira que as demais ferramentas desenvolvidas para o ambiente SGC e a ele integradas são utilizadas.

Para que esta conexão possa ocorrer é necessário que se implemente e integre ao SGC uma "ferramenta virtual". Esta ferramenta virtual representa a conexão entre o SGC e a "ferramenta real" disponível. Através desta ferramenta virtual o usuário poderá incluir nos procedimentos definidos no ambiente SGC a utilização de facilidades oferecidas por estas ferramentas. O usuário utiliza estas facilidades ao indicar na interface com o usuário da ferramenta virtual a execução dos serviços implementados pela ferramenta real. A eventual transferência de dados (fluxo de dados) poder ser indicada como para as demais ferramentas integradas, conforme descrito nos capítulos 4

e 6. Depois que estes dados estão disponíveis na ferramenta real, a interface do usuário desta poderá ser utilizada pelo usuário para solicitar que operações sejam realizadas sobre estes, quando necessário.

A metodologia que mais beneficia o SGC para a implementação desta conexão é o DDE. As ferramentas que suportam o DDE registram os serviços os quais estas implementam. No momento em que estes serviços estão registrados, todas as ferramentas que estão cadastradas e ativas para operações DDE são avisadas dos novos serviços registrados. Então estes serviços estarão disponíveis às ferramentas que desejarem utilizá-los. As ferramentas poderão utilizá-los ao enviar ao servidor (prestador de serviços) mensagens requisitando o acesso aos serviços prestados.

Para que se possa utilizar estes serviços a partir do SGC, é necessário que a ferramenta virtual possa identificar os serviços prestados e conheça o protocolo exigido para a transferência dos dados que necessitam ser processados durante a execução do serviço (como por exemplo quando desejamos enviar dados a um determinado conjunto de células de uma planilha de cálculo, ou quando se deseja que uma curva x,y seja visualizada a partir de uma ferramenta de visualização), ou para a obtenção dos dados gerados a partir da execução de um serviço (como por exemplo receber o resultado da solicitação da realização da convolução de uma forma de onda enviada a uma ferramenta matemática).

A implementação destas ferramentas virtuais representa a integração no ambiente SGC de uma ferramenta real já disponível, evitando-se que as facilidades por esta oferecidas sejam novamente implementadas em uma ferramenta específica.

Esta metodologia já foi validada ao conectar-se ao SGC a planilha de cálculo EXCEL [MIC92a]. Através desta conexão é possível traçar-se gráficos em tempo real, ou seja, gráficos em que os dados são mostrados a medida que estes são gerados ou adquiridos.

9.6 Conexão com *CAD Frameworks.*

Para a automação do processo de projeto de circuitos integrados, *CAD Frameworks* [BAR92] são amplamente utilizados. Estes *Frameworks* suportam objetos como *chips*, pinos e vetores de teste, objetos estes também suportados nos ambientes de teste automatizado. Mas, apesar disso, a tarefa de transferir este dados entre estes ambientes, definindo o teste a ser realizado, atualmente ainda é bastante complexa e trabalhosa.

O teste dos circuitos integrados pode ser realizado durante ou ao final do processo de produção, e a conexão entre o ambiente de teste e o ambiente de projeto em muito incrementaria a produtividade deste processo. É possível de desejável, por exemplo, trocar dados de projeto e vetores de teste entre estes dois ambientes, como mostrado na fig. 9.1. Estes dados seriam utilizados, no âmbito do ambiente de teste, para a definição das excitações e realização das medidas, através das ferramentas adequadas. Os resultados seriam enviados ao ambiente de projeto, onde poderiam ser visualizados e analisados. Esta ferramenta de visualização poderia, por exemplo, ser a mesma utilizada para visualizar os resultados das simulações realizadas durante a etapa de projeto, e poderia ser comum a ambos os ambientes.

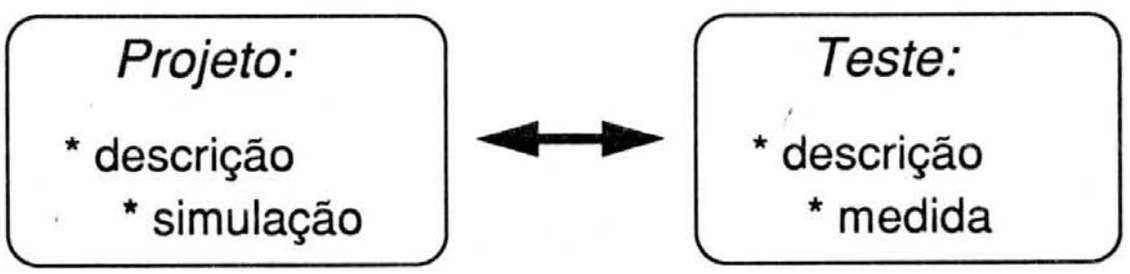


Fig. 9.1 Comunicação entre os Ambientes de Projeto e Teste.

10 CONCLUSÃO.

Neste trabalho foi analisada a utilização de ambientes de *software* tanto para a automação dos procedimentos de caracterização e teste quanto para o desenvolvimento de *software* para automação destes procedimentos. Um ambiente para automação destes procedimentos e desenvolvimento de *software*, chamado SGC, foi implementado. Após a realização deste trabalho, as seguintes observações podem ser feitas:

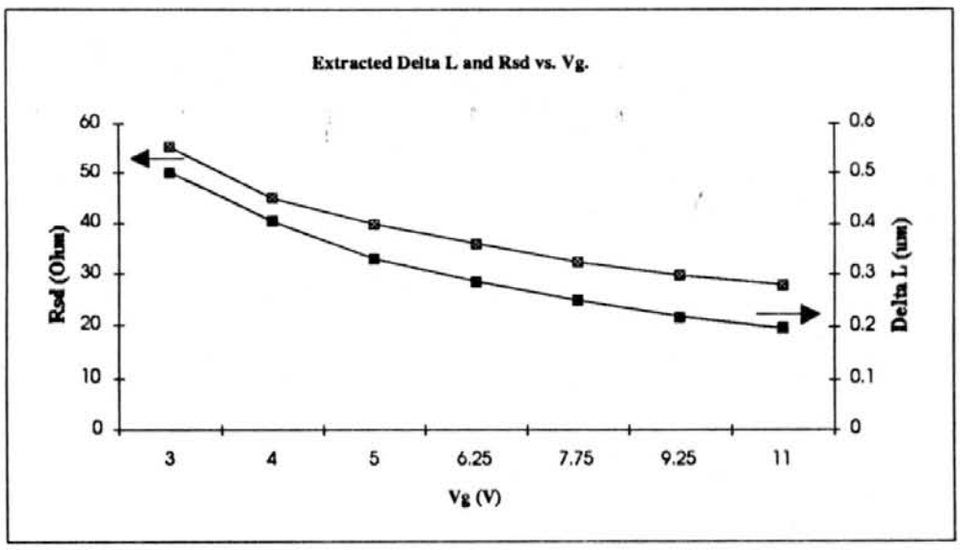


Fig. 10.1 Resultados de um procedimento de caracterização de transistores LDD MOSFET realizado com o auxílio do SGC.

i) O resultado mais importante e gratificante alcançado foi sem dúvida a implementação de um sistema que incrementa em muito a produtividade alcançada na realização dos procedimentos de caracterização e teste. Hoje o ambiente SGC é uma realidade que em muito auxilia o engenheiro de caracterização e teste em suas tarefas de rotina. Esta realidade pode ser comprovada quando da caracterização de transistores LDD MOSFET /WIR93/. Durante este processo de caracterização, as medidas elétricas necessárias foram realizadas utilizando a ferramenta para controle remoto e aquisição de dados com o HP4145, descrita no capítulo 7. Após a realização de medidas de tensão *versus* corrente com os transistores, e a conseqüente incorporação dos dados medidos à base de dados do SGC, estes ficaram disponíveis para tratamento através da planilha de cálculo EXCEL /MIC92a/ ou através do programa de tratamento de curvas SIMDE

/FER88/. Estes dados medidos foram então diretamente carregados para células desta planilha ou transferidos para o programa de tratamento de curvas, onde foram processados. Alguns dos resultados obtidos com a utilização do SGC quando da caracterização destes transistores LDD MOSFET estão mostrados na figura 10.1.

O SGC foi também utilizado em de outros procedimentos de caracterização e teste. Durante a sua utilização, a módulo de gerenciamento dos procedimentos, com sua linguagem visual, mostrou-se bastante versátil e de simples utilização, o mesmo ocorrendo com o módulo de gerenciamento das ferramentas. A documentação dos procedimentos, através do armazenamento de todas as condições sob as quais estes são realizados e de comentários automaticamente gerados pelas ferramentas e outros fornecidos pelos usuários, mostrou-se adequada para evitar que ocorresse a proliferação de dados desconexos no ambiente de caracterização e teste. os mecanismos adotados fazem com que informações relevantes relacionadas aos dados estejam sempre adequadamente documentadas.

A interface com o usuário das ferramentas integradas e dos diversos módulo do SGC mostrou-se bastante homogênea, facilitando o rápido reconhecimento da funcionalidade de cada módulo ou ferramenta utilizada.

Repetir procedimentos préviamente definidos mostrou-se uma tarefa bastante simples. Os mecanismos providos para esta repetição mostraram-se adequados e completamente funcionais.

ii) Outro resultado importante obtido com o SGC é a possibilidade de realizar-se a transferência e compartilhamento de metodologias de caracterização e teste entre diversos grupos de trabalho. A seqüência de passos realizados através das diversas ferramentas acopladas ao SGC durante um procedimento implementa uma determinada metodologia de caracterização e teste. O SGC permite que esta metodologia seja compartilhada, trabalhada e estudada por grupos de trabalho diferentes, que podem estar em locais de trabalho distintos. Isto é possível através da transferência entre grupos de trabalho dos procedimentos de caracterização e teste definidos com o auxílio do SGC. Estes procedimentos são transferidos em forma de arquivos, que podem ser lidos, visualizados, modificados e repetidos em qualquer instituição que disponha do SGC com as devidas ferramentas a ele acopladas.

iii) Como mencionado na capítulo anterior, um sistema de *software* jamais poderá ser considerado acabado, em seu estado definitivo. Uma importante contribuição do ambiente SGC é sem dúvida a definição de uma organização para um Sistema de Informação e Medida (SIM) que possa absorver mudanças facilmente. Esta flexibilidade do SGC é devida ao paradigma orientado a objetos adotado e a sua organização em módulos bem definidos, divididos em 3 níveis hierárquicos, que podem ser dinamicamente alterados, ou novos módulos acrescentados (como novas ferramentas ou *drivers*) sem que com isto perca-se a compatibilidade entre os mais diversos módulos integrantes do sistema.

A validade deste modelo pode ser comprovada quando da substituição da placa GPIB de fabricação da empresa Sistema Técnicos e Digitais S.A. /STD88/, inicialmente utilizada no laboratório de microeletrônica deste instituto, por uma nova placa adquirida da empresa National Instruments Corporation /NAT90/. Quando esta substituição foi realizada, foi necessário apenas o desenvolvimento de um novo objeto de *software* para modelar esta nova placa de interface, ou seja, integrou-se ao SGC um novo *driver* GPIB. Os procedimentos anteriormente definidos continuaram válidos, sem que nenhuma alteração nestes fosse necessária. Da mesma forma, nenhuma alteração foi necessária ao nível das ferramentas para que estas passassem a utilizar o novo *driver* GPIB para acesso aos instrumentos remotos.

iv) O ambiente SGC transforma a tarefa de desenvolver programas para a automação dos procedimentos de caracterização e teste em um trabalho cooperativo, permitindo que os seus mais diversos componentes sejam desenvolvidos por pessoas diferentes, em locais distintos, e acoplados ao sistema de maneira fácil e rápida, tornando-se então parte integrante do ambiente. Isto pode ser comprovado durante o desenvolvimento e integração de ferramentas e *drivers* ao SGC. Naquela oportunidade, não foi necessário que cada um dos projetistas de ferramentas conhecesse os detalhes da implementação dos demais módulos componentes do SGC. Foi necessário apenas que se conhecesse a metodologia de interfaceamento entre módulos adotada, ou seja, as facilidades oferecidas pelo módulo de intercomunicação do SGC.

Esta afirmação também pode ser comprovada quando da instalação do SGC junto ao LAC - Laboratório Central de Eletrotécnica e Eletrônica, Convênio entre Universidade Federal do Paraná e Companhia Paranaense de Energia (COPEL). Ao instalar-se o SGC no LAC, todas as ferramentas anteriormente implementadas no âmbito

deste curso de pós-graduação ficaram disponíveis para utilização. Ao desenvolver-se uma nova ferramenta (HP3325, descrita no item 7.6), esta ficou disponível para utilização em todas as instituições que utilizarem o SGC.

v) A metodologia oferecida pelo SGC para a implementação e acoplamento de novas ferramentas mostrou-se adequada. Quando da instalação do SGC junto ao LAC - Laboratório Central de Eletrotécnica e Eletrônica, Convênio entre Universidade Federal do Paraná e Companhia Paranaense de Energia (COPEL), em Curitiba-PR, no final do mes de agosto do corrente, realizou-se a análise, implementação e integração de uma nova ferramentas de *software* ao SGC, partindo-se do esqueleto de ferramenta descrito no item 8.3. Os resultados alcançados foram altamente satisfatórios, já que graças a esta metodologia foi possível a implementação e integração no ambiente SGC de uma ferramenta para o controle remoto do gerador de funções HP3325 (descrito no item 7.6) com um trabalho despendido menor que 30 horas/homem, com uma equipe que não possuía conhecimentos prévios do ambiente SGC, e também com pouquíssimo experiência de programação em ambiente *MS-Windows*. Estima-se que sem as facilidades oferecidas pelo SGC esta implementação custaria no mínimo 100 horas/homem de trabalho, sem que ao final se obtivesse as vantagens oferecidas pela integração da ferramenta em um ambiente integrado como o SGC.

vi) Através da implementação e posterior utilização das diversas ferramentas descritas no capítulo 7 foi possível validar as funções de intercomunicação oferecidas pelo SGC. Através desta implementação e utilização das ferramentas também foi possível validar o modelamento e a implementação da base de dados. A base de dados e os mecanismos de intercomunicação mostraram-se adequados e completamente operacionais.

vii) A utilização do ambiente SGC permite que as potencialidades dos equipamentos utilizados para caracterização e teste sejam ampliadas. Esta afirmação é exemplificada pelas diversas facilidades implementadas pelo HP4145 /HP89a/ e não disponíveis quando da sua operação pelo seu painel frontal (local). Funções estas somente disponíveis através do seu controle remoto com a ferramenta descrita no item 7.1. Esta afirmação também é exemplificada através da linguagem para definição de procedimentos implementada que, em conjunto com as ferramentas para controle remoto

de equipamentos, permite que equipamentos mais simples como uma fonte de tensão e um multímetro sejam utilizados para programar-se a aquisição e exibição de curvas de corrente *versus* tensão (IxV) em tempo real, como mostrado no exemplo do item 4.3.2.2. Sem as facilidades oferecidas pelo ambiente SGC isto seria possível somente com a utilização de equipamentos bem mais complexos, como o HP4145, por exemplo.

Resumindo, pode-se concluir que os objetivos no capítulo 1 propostos foram atingidos e que ao longo do tempo muito aprendeu-se sobre o tema que é objeto deste trabalho. Esta experiência adquirida poderá ser utilizada para melhorar o sistema implementado. O ambiente implementado está disponível para auxiliar àqueles que realizam procedimentos de teste e caracterização bem como àqueles empenhados na tarefa de desenvolver programas de computador (ferramentas de *software*) para tal fim.

BIBLIOGRAFIA

[AKS 92] AKSIT, Mehmet; BERGMANS, Lodewijk. Obstacles in Object-Oriented Software Development. **ACM Sigplan Notices**, New York, v. 27, n. 10, p. 341-358, Oct. 1992.

[AUD 89] AUDIO PRECISION INC. **Audio Precision System One**: User Manual. Beaverton, USA: Audio Precision Inc., 1989. 298 p.

[AUT 90] AUTOMATED METROLOGY. In: FLUKE AND PHILIPS. **Fluke and Philips Catalogue**. Eindhoven, The Netherlands: Philips, 1990, p. 220-238.

[BAM 90] BAMPI, Sergio. An Experiment on PC-Based Automated Extraction of Electrical Parameters for VLSI MOSFETs: Methods, Algorithms, and Implementation. **Microprocessing and Microprogramming - Euromicro Journal**, Paris, v. 28, p. 277-282, Mar. 1990.

[BAR 92] BARNES, Timothy J. et al. **Electronic CAD Frameworks**. Boston: Kluwer Academic, 1992. 198 p.

[BOR 91] BORLAND INTERNATIONAL. **Windows API Guide**: Reference: Version 3.0. Scotts Valley, CA: Borland International, 1991-1992. 3 v.

[BOR 92] BORLAND INTERNATIONAL. **BorlandC++**: Version 3.1: Programmer's Guide. Scotts Valley, CA: Borland International, 1992. 467 p.

[CAD 92] CADDEMI, A.; MARTINES, G.; SANNINO, M. Automatic Characterization and Modeling of Microwave Low-Noise HEMT's. **IEEE Trans. on Instrum. and Meas.**, New York, v. 41, n. 6, p. 946-950, Dec. 1992.

[CHA 92] CHAMPEAUX, Dennis de; LEA, Doug; FAURE, Penelope. The Process of Object Oriented Design. **ACM Sigplan Notices**, New York, v. 27, n. 10, p. 45-63, Oct. 1992.

[CHA 92a] CHAMPEAUX, Dennis de; ANDERSON, Al; FELDHOUSEN, Ed. Case Study of Object-Orinted Software Development. **ACM Sigplan Notices**, New York, v. 27, n. 10, p. 377-391, Oct. 1992.

[CLA 87] CLAESEN, L. et al. Open Framework of Interactive and Communicating CAD Tools. In: IFIP WG 10.2 WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, Nov. 1987, Paderborn, Germany **Proceedings...** Amsterdam: North-Holland, 1988. 256 p. p. 133-157.

[COR 94] CORRIGAN, Peter. **Oracle Performance Tunning.** Sebastapol: O'Reilly, 1994. 603 p.

[CRA 91] CRANDALL, Richard E. **Mathematica for the Sciences.** California: Addison-Wesley, 1991. 300 p.

[DAP 92] DAPONTE, P.; NIGRO, L.; TISATO, F. Object-Oriented Deseign of Measurement Systems. **IEEE Trans. on Instrum. and Meas.**, New York, v. 41, n. 6, p. 874-880, Dec. 1992.

[DOE 84] DOERNBERG, J. Full-Speed Testing of A/D Converters. **IEEE Journal of Solid-State Circuits**, New York, v. SC-19, p. 820-827, Dec. 1984.

[FEE 92] FEELEY, Michael J.; LEVY, Henry M. Distributed Shared Memory with Versioned Objects. **ACM Sigplan Notices**, New York, v. 27, n. 10, p. 247-262, Oct. 1992.

[FER 88] FERREIRA, L. F. et al. SIMDE: Sistema de Apoio à Simulação de Dispositivos Eletrônicos. In:

CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA (SBMicro), 3., São Paulo, 12 a 14 de julho de 1988. **Anais...** São Paulo: Sociedade Brasileira de Microeletrônica, 1988. 622 p. p. 536-538.

[FER 90] FERRERO, A. Software for Personal Instruments. **IEEE Trans. on Instrum. and Meas.**, New York, v. 39, n. 6, p. 860-863, Dec. 1990.

[GLI 90] GLINERT, E. P. **Visual Programming Environments:** Paradims and Systems. Los Alamitos: IEEE Computer Society Press, 1990. 661 p.

[GOS 90] GOSSAIN, Sanjiv; ANDERSON, Bruce. An Iterative-Design Model for Reuseble Object-Oriented Software. **ACM Sigplan Notices**, New York, v. 25, n. 10, p. 12-27, Oct. 1990.

[GRO 90] GROENING, B. et al. From Tool Encapsulation to Tool Integration. In: IFIP WG 10.2 WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, 2., Nov. 1990, Charlottesvile, USA, **Proceedings...** Amsterdam: North-Holland, 1991. 326 p. p. 21-36.

[GUY 90] GUY, Emmanuel T. et al. Tool Communication in CFI Frameworks. In: IFIP WG 10.2 WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, 2., Nov. 1990, Charlottesvile, USA, **Proceedings...** Amsterdam: North-Holland, 1991. 326 p. p. 213-228.

[HAN 91] HANSON, Eric N.; HARVEY, Tina M.; ROTH, Mark A. Experiences in DBMS Implementation Using an Object Oriented Persistent Programming Language and a Database Toolkit. **ACM Sigplan Notices**, New York, v. 26, n. 11, p. 314-328, Nov. 1991.

[HAY 91] HYES, Fiona; COLEMAN, Derek. Coherent Models for Object-Oriented Analysis. **ACM Sigplan Notices**, New York, v. 26, n. 11, p. 171-183, Nov. 1991.

[HEL 90] HELM, Richard; HOLLAND, Ian M.; GANGOPAHYAY, Dipayan. Contracts: Specifying Behavioral Compositions in Object-Oriented Systems. **ACM Sigplan Notices**, New York, v. 25, n. 10, p. 12-27, Oct. 1990.

[HP 89] HEWLLET-PACKARD CO. **HP4145B Semiconductor Parameter Analizer**: Operation Manual. California: Hewllet-Packard Co., April 1989. 206 p.

[HP 89a] HEWLLET-PACKARD CO. **HP4284A Precision LCR Meter**: Operation Manual. California: Hewllet-Packard Co., Mar. 1989. 421p.

[ISH 92] ISHIKAWA, Yukata. Communication Mechanism on Autonomous Objects. **ACM Sigplan Notices**, New York, v. 27, n. 10, p. 303-314, Oct. 1992.

[KAC 90] KACHEL, G.; RADEKE, E.; HEIJENGA, W. Support of CAX-Applications by IDM, a Data Model of a Non-Standard Data Base System. In: IFIP WG 10.2 WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, 2., Nov. 1990, Charlottesvile, USA, **Proceedings...** Amsterdam: North-Holland, 1991. 326 p. p. 365-380.

[KIL 91] KILOV, Haim. Object concepts and bibliography. **ACM Sigplan Notices**, New York, v. 26, n. 10, p. 11-13, Oct. 1991.

[KIM 89] KIM, Won; LOCHOVSKY, Frederick H. **Object-Oriented Concepts, Databases and Applications.** New York: ACM Press, 1989. 602 p.

[KRA 90] KRAFT, Niederlagenmanshaut A. Embedded Tool Encapsulation. In: IFIP WG 10.2 WORKSHOP ON

TOOL INTEGRATION AND DESIGN ENVIRONMENTS, 2., Nov. 1990, Charlottesvile, USA. **Proceedings...** Amsterdam: North-Holland, 1991. 326 p. p. 9-20.

[KRO 90] KRONMILLER, Theodor CAD Framework Integration: A Case Study. In: IFIP WG 10.2 WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, 2., Nov. 1990, Charlottesvile, USA. **Proceedings...** Amsterdam: North-Holland, 1991. 326 p. p. 1-8.

[LEW 91] LEWIS, John A. et al. An Empirical Study of the Object Oriented Paradigm and Software Reuse. **ACM Sigplan Notices**, New York, v. 26, n. 11, p. 184-196, Nov. 1991.

[MAE 89] MAEDER, Roman E. **Programming in Mathematica.** California: Addison-Wesley, 1989. 264 p.

[MAR 87] MARCHESI, M. et al. Object Oriented Programming for VLSI CAD Tool Integration. In: IFIP WG 10.2 WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, Paderborn, Germany, Nov. 1987. **Proceedings...** Amsterdam: North-Holland, 1988. 256 p. p. 85-104.

[MEY 88] MEYER, Bertrand. **Object-Oriented Software Construction.** United Kingdom: Prentice Hall International, 1988. 441 p.

[MIC 90] MICROSOFT CORPORATION. **Microsoft Windows Software Development Kit**: Reference: Version 3.0. Redmond: Microsoft Corporation, 1990. 2 v.

[MIC 92] MICROSOFT CORPORATION. **Microsoft Excel para Windows**: Guia do Usuário: Planilha eletrônica completa com gráficos e banco de dados. Redmond: Microsoft Corporation, 1992. 2 v.

[MIC 92a] MICROSOFT CORPORATION. **Microsoft Access**: Language Reference: Functions, statments, methods, properties and actions: Relational Databse Management System for Windows. Redmond: Microsoft Corporation, 1992. 543 p.

[MIL 87] MILLER, D. M. **Developments in Integrated Circuit Testing**. California: Academic Press, 1987. 441 p.

[MOL 86] MOLER, Cleve et al. **PC-MatLab for MS-DOS Personal Computers**. Massachusets: The MathWorks, 1986. 328 p.

[MUR 93] MURAVYOV, S. V.; SAVOLAINEN, Vesa. Development Particularities for Programming Systems of Measurement Procedures. **IEEE Trans. on Instrum. and Meas.**, New York, v. 42, n. 5, Oct. 1993, p. 906-912.

[NAT 90] NATIONAL INSTRUMENTS CORPORATION. **Using Your NI-488.2 Software with Microsoft Windows**. Austin, Texas: National Instrum. Corp.,1993. 54 p.

[NEL 91] NELSON, Michael L. Concurrency and Object-Oriented Programming. **ACM Sigplan Notices**, New York, v. 26, n. 10, p. 63-72, Oct. 1991.

[ORA 90] ORACLE CORPORATION. **Oracle RDBMS Database**: Administrator's Guide: Version 6.0. Redwood Shores: Oracle Corp., 1990. 1 v.

[OTT 90] OTTENSMANN, John R. **Quattro**: Guia do Usuário. São Paulo: McGraw-Hill, 1990. 389 p.

[PIL 87] PILOTY, R.; WEBER, B. IREEN - A Datamodel for Tool Integration in Open Microelectronic CAD-Systems. In: IFIP WG 10.2 WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, Nov. 1987, Paderborn, Germany. **Proceedings**... Amsterdam: North-Holland, 1988. 256 p. p. 105-132.

[RIN 88] RINGEARD, C.; MALOEUVRE, M. CALIBRAT: A Software for Control and Calibration of Electronic Measurement Devices. **IEEE Trans. on Instrum. and Meas.**, New York, v. 37, n. 6, p. 497-500, Dec. 1988.

[RUS 90] RUSSO, F.; BROILI, S. A User-Friendly Environment for the Generation of High Portable Software in Computer-Based Instrumentation. **IEEE Trans. on Instrum. and Meas.**, New York, v. 39, n. 6, p. 864-866, Dec. 1990.

[SOE 90] SOENEN, E. G. et al., A Framework for Design and Testing of Analog Integrated Circuits. **IEEE Trans. on Instrum. and Meas.**, New York, v. 39, n. 6, p. 890-893, Dec. 1990.

[SOU 89] SOUZA, Helena V. de. **Lotus 123**: Básico. Porto Alegre: Insituto de Informática da UFRGS, 1989. 82 p.

[TEK 89] TEKTRONIX, INC. **TEK2440 Digital Oscilloscope User Reference Guide.** Oregon: Tektronix, Inc., 1989. 45 p.

[VDA 92] VIDAL, Antônio G. da Rocha. **Clipper**: Versão Summer 87. Rio de Janeiro: LTC. 1992. 925 p.

[VID 92] VIDEIRA, I.; SARMENTO, H. Tool Integration Made Easier. In: IFIP WG 10.2/WG10.5 WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, 3., Mar. 1992, Bad Lippspring, Germany. **Proceedings...** Amsterdam: North-Holland, 1992. 309 p. p. 207-224.

[WIR 93] WIRTH, Gilson I.; WARTCHOW, Ricardo; BAMPI, Sergio. Measurement instrument control in the SGC system for testing CMOS VLSI microstructures. In: SEMINÁRIO INTERNO DE MICROELETRÔNICA, 8., abr. 1993, Porto Alegre.

Anais... Porto Alegre: Curso de Pós-Graduação em Ciência da Computação da UFGRS, 1993. 283 p. p. 69-71.

[WIR 93a] WIRTH, Gilson I.; BAMPI, Sergio; SIMÕES, José O. SGC: Measurement and Control System for Test Devices Characterization. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA (SBMicro), 8., set. 1994, Campinas. **Anais**... Campinas: Sociedade Brasileira de Microeletrônica, 1993. 1 v., p. XXI.13-XXI.19.

[WIR 93b] WIRTH, Gilson I. et al. Channel Length and Series Resistance Extraction in LDD MOS Transistors. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA (SBMicro), 8., set. 1994, Campinas. **Anais**... Campinas: Sociedade Brasileira de Microeletrônica, 1993. 1 v., p. VI.13-VI.16.

[WIR 94] WIRTH, Gilson I.; WARTCHOW, Ricardo; BAMPI, Sergio. SGC - Measurement Management and Control System for the Automation of Test Procedures. (Submetido a: **IEEE Trans. on Instrum. and Meas.**).

[WIR 94a] WIRTH, Gilson I. et al. Ambientes para Automação de Procedimentos de Caracterização e Teste. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA (SBMicro), 9., ago. 1994, Rio de Janeiro. **Anais**... Rio de Janeiro: Sociedade Brasileira de Microeletrônica, 1994. 583 p. p. 158-67.

[WIR 94b] WIRTH, Gilson I. et al. A Framework for Electrical Test Engineering Automation. In: IFIP WG10.5 INTERNATIONAL WORKING CONFERENCE ON ELECTRONIC DESIGN AUTOMATION FRAMEWORKS, 4., Nov. 1994, Gramado, Brasil. **Proceedings**...

Amsterdam: North-Holland, 1995. 243 p.
p. 165-174.

Informática  UFRGS

SGC - Um Ambiente para a Automação de Procedimentos de Caracterização e Teste.

Dissertação apresentada aos Senhores:

Prof. Dr. Altamiro Amadeu Suzim

Prof. Dr. Carlos Inácio Mammana (CTI/Campinas)

Prof. Dr. Ricardo Pezzuol Jacobi

Prof. Tiaraju Vasconcellos Wagner

Vista e permitida a impressão.
Porto Alegre, 21/12/94

Prof. Dr. Sérgio Bampi,
Orientador.

Prof. Dr. José Palazzo Moreira de Oliveira,
Coordenador do Curso de Pós-Graduação
em Ciência da Computação.