**AUTOMAÇÃO LABORATORIAL:
UM DOMÍNIO DE APLICAÇÃO E UM
SOFTWARE DE APOIO**

LUIZ MANOEL SILVA CUNHA

ORIENTADOR:

PROF. DR. PAULO CESAR MASIERO

**Dissertação apresentada ao Instituto de Ciências Matemáticas de
São Carlos - USP, como parte dos requisitos para obtenção do
título de Mestre em Ciências - Área: Ciências de Computação e
Matemática Computacional.**

USP - São Carlos

1993

Aos meus pais Hilton e Glória que
sempre me apoiaram, mesmo de longe.

À Maria Julia pelo incentivo e amor
que sempre me ofereceu no transcurso
de nossa vida juntos.

Aos meus filhos Leonardo e Juliana
pela felicidade de sua existência.

Ao meu irmão Hilton Silva Cunha.

AGRADECIMENTOS

A Deus pela força espiritual que me deu no transcorrer de todo curso.

Ao professor Dr. Paulo Cesar Masiero pela orientação deste trabalho.

À Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA) pela oportunidade de realizar este trabalho.

A todos os professores, funcionários e alunos do Instituto de Ciências Matemáticas de São Carlos (ICMSC) pela receptividade e atenção que me dispensaram.

À chefia e aos técnicos do Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura (CNPTIA) pelo constante apoio e incentivo durante a elaboração deste trabalho.

A todas as pessoas que de uma forma ou de outra contribuíram para realização deste trabalho.

CONTEÚDO

LISTA DE TABELAS.....	v
LISTA DE FIGURAS.....	vi
RESUMO.....	viii
ABSTRACT.....	ix
1 INTRODUÇÃO.....	1
1.1 Considerações Iniciais.....	1
1.2 Situação e Caracterização do Problema.....	2
1.3 Objetivos do Trabalho.....	4
1.4 Importância e Justificativa do Trabalho.....	5
1.5 Organização da Dissertação.....	7
2 REVISÃO DA LITERATURA.....	8
2.1 Considerações Iniciais.....	8
2.2 Fundamentos.....	9
2.2.1 Interfaces de Comunicação.....	9
2.2.2 Interface Homem-Máquina.....	15
2.2.3 Ambientes de Desenvolvimento de Software..	26
2.2.4 Geradores de Aplicação.....	36
2.3 Sistemas para Automação Laboratorial.....	42
2.4 Análise Comparativa dos Sistemas para Automação Laboratorial.....	53

2.5 Considerações Finais.....	57
3 ESPECIFICAÇÃO E DESENVOLVIMENTO DO SISTEMA DE AUTOMAÇÃO	
LABORATORIAL - SAL.....	59
3.1 Considerações Iniciais.....	60
3.2 Requisitos do Sistema de Automação Laboratorial.....	60
3.3 Aspectos Seleccionados do Projeto do SAL.....	66
3.4 Arquitetura do Protótipo.....	78
3.5 Configuração do Protótipo	80
3.6 Considerações Finais.....	86
4 DEMONSTRAÇÃO E AVALIAÇÃO DO PROTÓTIPO IMLEMENTADO.....	87
4.1 Considerações Iniciais.....	87
4.2 Um Exemplo de Uso do Sistema.....	87
4.3 Avaliação Preliminar do Protótipo Implementado.....	104
4.4 Considerações Finais.....	107
5 CONCLUSÃO.....	109
5.1 Considerações Finais.....	109
5.2 Sugestões para Futuras Pesquisas.....	110
BIBLIOGRAFIA.....	112
APÊNDICE 1.....	119

LISTA DE TABELAS

3.1 Relacionamento entre Equipamento e Interfaces de Comunicação.....	73
3.2 Interface de Comunicação e seus Parâmetros.....	74
4.1 Ações, Parâmetros e seus Respektivos Valores para Especificação do Programa.....	91
4.2 Faixa de Validade dos Parâmetros da Interface AD5012.....	95

LISTA DE FIGURAS

2.1	Símbolos Utilizados para Construção de Bloco.....	22
2.2	Símbolos e seus Subgrafos Correspondentes.....	23
2.3	Estrutura do Editor Gráfico.....	25
2.4	Tela Principal do Editor Gráfico.....	26
2.5	Desenvolvimento de Programa Utilizando Procedimentos Pré-Definidos.....	30
2.6	Esquema de Desenvolvimento de Aplicação Usando VU.....	36
2.7	Arquiteturas de Geradores de Aplicação. a) SDSA b) MDMA..	41
2.8	Menus do ASSYSTANT +.....	47
2.9	Opções de Controle do Hardware.....	48
2.10	Painel de Comandos e seu Respectivo Diagrama de Blocos..	50
2.11	Principais Características do Sistemas Abordados na seção 2.3.....	55
3.1	Uma Visão Geral do Domínio de Aplicação para o SAL.....	64
3.2	Arquitetura do Protótipo.....	79
4.1	Uma Visão Global do Experimento de Secagem de Feijão.....	88
4.2	Especificação do Programa.....	89
4.3	Janela Principal do SAL.....	92
4.4	Janela Básica do EDSAL.....	93
4.5	Janela de Edição do EDSAL.....	93
4.6	Janela de Composição de Programa.....	94
4.7	Menus de Definição.....	96
4.8	Janela de Entrada de Dados para os Parâmetros da Interface AD5012.....	97

4.9	Painel do Secador.....	98
4.10	Janela de Tipos de Estatísticas.....	99
4.11	Janela de Tipos de Gráficos.....	99
4.12	Janela para Entrada do Nome do Arquivo.....	100
4.13	Janela do Coletor de Dados.....	101
4.14	Estatísticas Básica.....	102
4.15	Gráfico de Barras.....	102
4.16	Listagem do Dados Coletados.....	103
4.17	Listagem da Especificação do Programa.....	103

RESUMO

Sistemas para automação laboratorial permitem automatizar de forma significativa tarefas rotineiras envolvidas em processos de investigação de um produto qualquer (oleos essenciais, biscoitos, soja, etc). Esses sistemas, além de permitirem a obtenção rápida e precisa de dados oriundos de um equipamento de medição, permitem também analisá-los rapidamente, proporcionando ao usuário condições para tomar decisões em um curto espaço de tempo.

Neste trabalho foram realizados estudos referentes a interfaces de comunicação, interface homem-máquina, ambientes de desenvolvimento de software, geradores de aplicação, e sistemas voltados para automação laboratorial, buscando formar uma base teórica para o desenvolvimento de um protótipo de sistema para automação laboratorial. Esse protótipo foi denominado de Sistema para Automação Laboratorial - SAL, permitindo que técnicos de laboratório com pouco conhecimento de programação construam programas para realizar coleta e análise de dados de um determinado experimento de forma bastante simples e rápida.

O sistema é adequado para coleta e análise de dados. A criação e execução de programas usando o SAL também é muito simples.

ABSTRACT

Laboratory Automation Systems can automate a significant portion of tasks involved in the process of investigating a product. Such systems, in addition to providing for quick and precise capture of data from input devices, also allows fast analysis of those data, thus helping users to take faster decisions.

This dissertation reviews concepts related to communication interfaces, man-machine interfaces , software development environments, application generators and systems for laboratory automation with the objective of laying the foundations for the development of a prototype system for laboratory automation (SAL). This system allows that laboratory technicians with small knowledge of computer programming be able to develop programs to collect and analyze data for an experiment in a simple and quick way. The system is specially adequate for on-line data collection and analysis. It's also very simple to create and run the programs developed using SAL.

CAPÍTULO 1

INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

O presente documento trata de sistemas destinados a automatizar tarefas desenvolvidas em laboratórios de pesquisa (por exemplo, tarefas relacionadas ao controle de qualidade de alimentos, biotecnologia, cromatografia, etc), que possuem equipamentos ligados a um computador digital. São abordados também alguns tipos de interfaces de comunicação e suas características básicas. Aspectos julgados importantes para a construção de interfaces homem-máquina e trabalhos na área de ambientes de desenvolvimento de software são também analisados, com o intuito de fundamentar o desenvolvimento de um protótipo de sistema para automação laboratorial.

Esse protótipo denomina-se Sistema de Automação Laboratorial - SAL e faz parte de estudos preliminares voltado para o apoio a alguns laboratórios de pesquisa existentes na Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA).

1.2 SITUAÇÃO E CARACTERIZAÇÃO DO PROBLEMA

Atualmente, alguns construtores de software vêm dedicando boa parte de seu tempo ao projeto e desenvolvimento de sistemas que possam ser utilizados como interface entre equipamentos de laboratório e um processador.

Baseado em informações de vários técnicos de laboratório da EMBRAPA, que já tiveram contato com alguns desses softwares, pode-se dizer que eles possuem bons recursos para aquisição, manipulação, análise de dados, controle de processos e emissão de gráficos.

Embora seja reconhecida por parte dos técnicos de laboratório da EMBRAPA a existência de bons recursos, principalmente nos softwares desenvolvidos no exterior, o item interface homem-máquina e a falta de um mecanismo simples para modificar o tipo de interface de comunicação e seus respectivos parâmetros previamente definidos ainda continuam sendo pontos que têm deixado a desejar. Em decorrência desta falha, vários softwares são sub-utilizados e, em alguns casos, descartados precocemente.

Identificou-se, também, que muitos técnicos de laboratório, hoje em dia, estão tentando adquirir conhecimentos de programação para poder superar os problemas encontrados para construir um software de interface entre um computador e um equipamento de laboratório. Este fato tem levado à construção de programas ou mesmo de sistemas inadequados e, também,

provocado o desvio do técnico de laboratório de suas reais funções, o que nem sempre é desejável.

Outro fator crítico é que vários softwares existentes só atendem a um determinado tipo de equipamento. Além disso, há situações onde existem o computador, o software e o equipamento a ser conectado e todo o processo de aquisição, manipulação e análise de dados é feito manualmente, o que constitui um risco muito grande e pouco aceitável nos dias atuais.

Segundo [TOZZI,83], [GATES,84] e [ENGLEMANN,84], um software destinado a realizar aquisição de dados e controle de processos necessita ter as seguintes características:

a. possuir um conjunto de sub-rotinas que possam ser chamadas por uma linguagem de programação de alto nível. Estas destinam-se a realizar a aquisição de sinal, conversão do sinal do formato analógico para o digital e vice-versa, realizar amostragem [NETTO,88], controlar o ganho do sinal, executar operação como mudança de escala de engenharia, linearização, cálculo do erro [PARK,88], análise de dados nos modos on-line ou off-line, exibir os dados em tempo real, etc.;

b. ter facilidades para que o usuário possa interagir com o software no modo on-line e programar o equipamento ainda que não seja um especialista em programação;

c. possuir mecanismo de sincronização, isto é, fazer com que o computador entenda uma solicitação externa para execução de um determinado processo. O mesmo suspenderá o processo que se encontra em execução, armazenando os dados, fazendo com que

o processo solicitado entre em operação e, ao seu final, este mecanismo reconstituirá todo o ambiente e voltará a atuar. Tudo isso realizado dentro de um determinado intervalo de tempo;

d. dispor de um mecanismo controlador de processos com a função de detectar erros e gerar um sinal que permita uma intervenção no equipamento, como por exemplo, a abertura de uma válvula, e

e. possuir um mecanismo para armazenamento de dados e programação do relógio de tempo real existente no computador.

1.3 OBJETIVOS DO TRABALHO

O objetivo deste trabalho, é lançar as bases para o desenvolvimento de um sistema que permita ao técnico de laboratório com pouco ou nenhum conhecimento de uso de computador, construir programas de forma eficiente, segura e bastante simplificada, para automatizar a coleta e análise de dados oriundos de equipamentos de laboratório, como por exemplo Cromatógrafo e Instron, que existem nos laboratórios de pesquisa (Análise Sensorial de Alimentos, Cromatografia) da EMBRAPA ou em outros laboratórios semelhantes. Com o desenvolvimento do sistema, pretende-se alcançar as seguintes metas: construir um sistema genérico o suficiente para ser utilizado em experimentos laboratoriais diversos dentro de um domínio de aplicação bem definido e tenha um mecanismo que facilite diversos tipos de interfaceamento entre o computador e

equipamento.

1.4 IMPORTÂNCIA E JUSTIFICATIVA DO TRABALHO

A computação laboratorial é o resultado lógico da evolução do uso do computador na ciência de uma maneira geral [SHOENFELD,77]. Em épocas passadas, os computadores existentes em laboratórios eram utilizados em estudos do aprendizado humano, comportamento dos animais e dos insetos e de vários sistemas sensoriais (próprios para transmitir sensações), bem como do sistema nervoso. Observou-se que foi gradativa a introdução do computador no laboratório, mas sua consolidação se deu de forma bastante rápida em virtude das facilidades por ele introduzidas.

Durante a década de 70, o uso do computador era bastante restrito, contrariamente aos tempos atuais, em que o computador está presente em quase todas as atividades humanas. Nos laboratórios modernos, o computador é uma peça fundamental, permitindo que equipamentos de diversas categorias se comuniquem, executando tarefas de aquisição de dados, controle e supervisão de processos, armazenamento e análise de dados nos modos on-line ou off-line.

A ausência do computador no laboratório, principalmente quando se executam tarefas de aquisição de dados e controle de processos, pode ocasionar dificuldades, tais como:

- a. atraso na reconstituição correta de uma seqüência de

controles;

b. impossibilidade de reativação em tempo bastante reduzido de um controlador;

c. demora para que um instrumento responda a algum estímulo. Em alguns casos, a demora em se obter uma resposta pode acarretar a perda total de um experimento ou ter o mesmo suas condições iniciais alteradas; e

d. baixa confiabilidade dos dados coletados.

Com a introdução do SAL nos laboratórios da EMBRAPA, principalmente nos do CTAA, o processo de coleta e tratamento de dados poderá ser feito de forma bem mais rápida, mais fácil e com menor ocorrência de erros. Um outro aspecto importante é que o SAL contribuirá para tornar os técnicos de laboratório mais independente dos técnicos de computação da EMBRAPA no que diz respeito a construção de programas para um determinado domínio de aplicação bem definido. A quantidade de tempo gasto e o número de treinamento em técnicas e linguagem de programação voltados para os técnicos de laboratório também serão reduzidos. Com esse sistema vários equipamentos de laboratório da EMBRAPA que hoje são utilizados de forma pouco adequada terão condições de ser melhor aproveitados. No geral, o desenvolvimento do sistema justifica-se, principalmente, pelo potencial de ganho em termos de produtividade e qualidade de resultados gerados por experimentos laboratoriais desenvolvidos na EMBRAPA.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

No capítulo 2, revisam-se vários tipos de interfaces de comunicação e suas respectivas características. Aspectos relevantes para construção de interface homem-máquina e pesquisas na área de ambientes de desenvolvimento de software também são discutidos. Nesse capítulo descrevem-se também vários sistemas destinados a realizar processos que automatizam rotinas de trabalho de laboratórios. Entre elas estão a aquisição, armazenamento e análise de dados e emissão de gráficos. Finalizando, é apresentada uma avaliação dos métodos, técnicas e softwares revisados.

O capítulo 3 apresenta as especificações, os requisitos para a implementação do SAL e as decisões de projeto. A arquitetura do protótipo e a descrição de cada módulo que compõe o sistema são também apresentados. O capítulo 4 mostra o protótipo implementado e uma avaliação preliminar dos resultados alcançados. O capítulo 5 apresenta as conclusões e sugestões para trabalhos futuros.

CAPÍTULO 2

REVISÃO DA LITERATURA

2.1 CONSIDERAÇÕES INICIAIS

A revisão da literatura apresentada neste capítulo tem como propósito dar uma visão geral das publicações utilizadas neste trabalho para formar a base de conhecimento necessária para a construção de um protótipo de sistema para automatizar da EMBRAPA.

Na seção 2.2 são introduzidos conceitos relativos à interfaces de comunicação e suas respectivas características. Em seguida são abordados aspectos relativos à programação visual, aplicáveis na construção de interfaces homem-máquina, além de uma comparação entre o uso de figuras e textos, complementando-se com dois exemplos de interfaces homem-máquina. A seguir são apresentados alguns tipos de ambientes de programação e suas respectivas características, bem como aspectos básicos sobre geradores de aplicação por se tratar de ferramentas que agilizam o processo de desenvolvimento de software.

Na seção 2.3 encontram-se descritos alguns sistemas voltados para automação de laboratórios. Na seção 2.4 é realizada uma comparação entre os vários sistemas de automação

laboratorial discutidos e, finalmente, conclusões sobre o capítulo como um todo são apresentadas na seção 2.5.

2.2 FUNDAMENTOS

Nesta seção são introduzidos conceitos importantes para a construção de um sistema para a automação laboratorial.

2.2.1 INTERFACES DE COMUNICAÇÃO

Para a troca de informações entre equipamentos de laboratório e um computador é necessário o uso de um veículo confiável que possibilite a execução dessa tarefa. Esse veículo é conhecido como interface de comunicação. Os meios físicos mais comuns são os cabos coaxiais, fio de telefone (par trançado), placas analógicas-digitais-analógicas e digitais-analógicas-digitais.

Em alguns artigos o termo interface de comunicação é substituído por protocolo de comunicação. Este pode ser definido como sendo formado por um software mais um dos meios físicos já mencionados, apresentando características especiais.

Observou-se que apesar de muitos equipamentos de laboratório serem constituídos pelos mesmos componentes, não se pode dizer que suas interfaces de comunicação sejam as mesmas. A seguir serão abordados alguns tipos de interfaces de comunicação.

2.2.1.1 TIPOS DE INTERFACES DE COMUNICAÇÃO

CONVERSORES ANALÓGICOS-DIGITAIS-ANALÓGICOS(A/D/A)

O uso destes conversores é a forma mais barata de se interligar equipamentos de laboratório ao microcomputador. Eles são utilizados em situações onde é necessário que sinais analógicos de um dispositivo eletrônico sejam convertidos para números no formato digital.

O processo de conversão analógico-digital é talvez a aplicação mais crítica que se realiza em laboratório, porque erros nesta etapa causam uma distorção no número gerado [FORD,85]. Segundo [CLUNE,85], os conversores analógicos-digitais apresentam as seguintes limitações:

- a. somente realizam amostragem de uma única voltagem no tempo;
- b. são lentos;
- c. por serem de baixo custo, sua linearidade não é muito boa; e,
- d. são muito sensíveis a ruídos.

Apesar de apresentarem limitações, os conversores analógicos-digitais são bastante utilizados quando eles substituem o ser humano no processo de conversão de sinais.

Os conversores digitais-analógicos são dispositivos mais simples do que os conversores analógicos-digitais e sua tarefa é a de decodificar um dado que está no formato digital

em um sinal elétrico análogo [ENGLEMAN,84] e [WIRTH,86]. Usuários finais raramente projetam conversores desta natureza, a não ser que sua aplicação requeira um volume de conversão bastante grande ou que a mesma não seja uma aplicação comum.

INTERFACE DIGITAL/DIGITAL(D/D) e RS-232-C

Se um conversor analógico-digital não satisfaz as necessidades do usuário, então é necessário um equipamento stand alone, que possa transferir a informação digital para um computador via uma interface digital/digital [CLUNE,85].

A primeira vantagem que a interface digital/digital tem sobre o conversor analógico/digital é que os dados podem ser avaliados em alta velocidade e uma cópia instantânea da análise no formato digital é armazenada no buffer do equipamento.

Existem dois tipos de interfaces D/D: serial, que realiza a transferência bit a bit e a paralela que realiza transferência byte a byte. A interface serial mais usada é a RS-232-C. Esta apresenta as seguintes contra-indicações [CLUNE,85]:

- a. não é padrão;
- b. não verifica os tipos de dados enviados;
- c. produz bastante ruído;
- d. é lenta; e,
- e. é capaz de interligar somente dois periféricos.

Como ponto forte, apresenta a capacidade de transmitir

informações a longas distâncias e é a única que está disponível nos instrumentos de laboratório mais antigos [CLUNE, 85] .

INTERFACE GPIB/IEEE-488 e HPIB

Funciona tanto no modo serial quanto no modo paralelo. Segundo [CLUNE,85], os pontos fortes apresentados pela interface IEEE-488 são:

- a. sua resistência a interferências;
- b. permite que sejam conectados ao mesmo tempo até quinze dispositivos;
- c. é rápida para um microcomputador; e,
- d. é padrão e está disponível.

A principal desvantagem é que o tamanho do cabo utilizado não pode ser maior que vinte metros sem que se tenha um modem. Usada em situações adversas, a transmissão será de baixa qualidade e susceptível a ruídos. Apesar de apresentar limitações é amplamente utilizada. Os equipamentos ligados a esta interface podem enviar ou receber dados em ambas as direções.

A interface IEEE-488 pode ser programada para atuar em dois modos de operação. O modo de operação "comando" é utilizado em controle de processos, onde é possível programar o equipamento de laboratório para realizar uma leitura, por exemplo. No modo "dado", o dado é transferido do mandante para o receptor através da interface.

A IEEE-488 é composta por vinte e quatro linhas, das quais oito formam a base. As dezesseis restantes são divididas em três grupos:

- a. oito linhas de dados bi-direcionais;
- b. cinco linhas de gerenciamento; e,
- c. três linhas de controle.

O padrão IEEE-488 garante a compatibilidade elétrica entre os instrumentos mas não assegura que dois instrumentos irão se entender. Diante deste fato, alguns instrumentos transmitem dados em diferentes formatos. Isto levou a Hewlett Packard Corporation a desenvolver uma interface que se compatibiliza com as transmissões de dados.

A referida interface chama-se Hewlett-Packard Interface Bus (HPIB). Esta apresenta as seguintes características:

- a. todas as informações são transferidas no formato ASCII;
- b. as informações são transferidas da esquerda para a direita; e,
- c. uma transferência de dados termina com o código ASCII treze ou ASCII dez.

A vantagem de se fazer uso do padrão é que os dados podem, por exemplo, ser enviados diretamente para uma impressora, produzindo uma saída formatada, em uma aplicação onde os dados são coletados continuamente.

INTERFACES UNIVERSAIS

Além dos padrões de comunicação mostrados anteriormente, existem vários outros [BARKER,84]. Entre eles podemos citar RS-422-A, RS-423-A [ENGLEMANN,84]. Estes padrões referem-se somente a níveis elétricos e tipos de sinais presentes nos elos de comunicações.

Nos padrões RS-422-A e RS-423-A as informações podem ser codificadas usando os seguintes formatos:

- a. ASCII para troca de informações;
- b. binário;
- c. BCD (Binary-Code Decimal); e,
- d. hexadecimal.

As interfaces Universais podem transmitir dados na forma síncrona ou assíncrona. Quando um dispositivo reivindica compatibilidade com a interface RS-232-C, uma interface que codifique informações no formato ASCII é utilizada.

Padrões tais como HDLC (High-Level Data-Link Control), SDLC (Synchronous Data-Link Control) e X.25 [ENGLEMANN,84] são de interfaces de comunicação assíncronas que definem, não somente a interface elétrica (nível físico), mas também o formato do dado, endereços, controle e códigos para checagem de erros.

2.2.2 INTERFACE HOMEM-MÁQUINA

A construção de sistemas não é uma tarefa simples, ainda mais quando se trata de um sistema que possui, de alguma forma, interação homem-máquina. Sistemas desta natureza não são julgados somente pelos seus aspectos computacionais. Aspectos relacionados a estilo de interação, fatores humanos e linguagem de comunicação, entre outros, possuem um peso muito grande.

Nos tempos atuais, usar computador não é privilégio somente de técnicos especializados em computação e sim da sociedade como um todo. Para que cada vez mais pessoas utilizem computadores, é necessário entre outras coisas construir interfaces homem-máquina que:

- a. não induzam os usuários a cometer erros;
- b. sejam assimiladas facilmente;
- c. contenham realmente as funções que as pessoas necessitam;
- e,
- d. sejam fáceis e agradáveis de serem utilizadas.

O projeto e implementação de uma interface homem-máquina que satisfaça aos quatro itens acima mencionados, requer uma equipe multidisciplinar, envolvendo psicólogos, analistas de sistemas, o usuário e o projetista da interface. Durante a elaboração da interface, questões tais como:

- a. Quem usará o sistema ?
- b. Em que momento ocorrerá a interação homem-máquina ?
- c. Qual o grau de conhecimento que o usuário possui sobre a

tarefa que irá executar ?

devem ficar bem esclarecidas. Além destas questões, uma boa interface homem-máquina deve englobar [SANTOS, 89]:

a) **Browsing** - A interface deve permitir que o usuário obtenha informações de qualquer parte da mesma, diferente daquela que está sendo utilizada.

b) **Condições de absorver habilidade do usuário** - A interface deve permitir que usuários de todos os níveis de habilidade trabalhem efetivamente.

c) **Prevenção de erro e aceitabilidade** - A interface deve prevenir ou minimizar a possibilidade de erro do usuário. A mesma deve ser projetada e construída com a suposição de que o usuário está correto. Todas as entradas, requisições e ações devem ser interpretadas e executadas nesse contexto.

d) **Facilidade de Aprendizado** - A interface deve ter todas as vantagens da tecnologia do computador para inicialmente ensinar novos usuários e fornecer aprendizado rápido aos usuários experientes.

e) **Ajuda** - A interface deve ser capaz de fornecer explicação referente a comandos, dados, operações, etc.

f) **Conversação** - A interface deve comportar-se como uma máquina e não como uma pessoa. Frases ou palavras tais como, "por favor", "voce deseja" e "se voce quer" podem ser sistematicamente eliminadas para aumentar a clareza. A interface deve ser fácil de ser utilizada, ajudando ao usuário através da sequência de operações.

2.2.2.1 PROGRAMAÇÃO VISUAL

Quase todas as linguagens de programação têm uma sintaxe baseada em texto, requerendo do usuário, conhecimentos e experiência em programação. Dessa forma, quando um usuário que não possui nem experiência e nem conhecimento de programação utiliza o computador para desenvolver suas próprias aplicações, ele geralmente encontra dificuldades. Uma maneira de resolver esse problema, é utilizar a Programação Visual(PV) [ICHIKAWA,90]. Este tipo de programação fornece ao usuário um conjunto de figuras na tela do computador, que após serem agrupadas ou selecionadas através de um mouse, por exemplo, executam uma ou mais funções.

Programação Visual tem se mostrado um estilo de programação bastante interessante por não requerer memorização dos comandos, por oferecer maior flexibilidade na construção do programa e também por permitir a construção de programas baseados na técnica de programação-por-exemplo. Isto é, especificam-se os valores para os dados e objetos são designados como exemplos para serem processados através do programa e, em seguida, é exibido o comportamento do programa em resposta aos dados e aos objetos selecionados.

Embora, atualmente, este estilo de programação esteja sendo bastante utilizado, principalmente em ambientes de desenvolvimento de software, três fatores ainda causam uma certa resistência por parte de alguns usuários. São eles

[ICHIKAWA,90]:

a) Programação baseada em figuras pode ser bom para programadores iniciantes mas não para profissionais em computadores - A vantagem de programação baseada em figuras sobre a programação textual é que isto torna o processo de entendimento e aprendizado mais fácil, especialmente para usuários que não são especialistas em computação.

a) Programação baseada em figuras é bom para pequenos programas e não para grandes programas - Um ponto negativo para programação que faz uso de figuras, é que gasta bastante espaço na tela do computador, tornando difícil a construção de grandes programas. Um outro ponto desfavorável, é que utilizando figuras não se consegue ver todo o programa e sim uma pequena parte, comparando com o que pode ser visto quando se utiliza programação textual.

b) Existem de fato abordagens razoáveis para projeto de boas interfaces baseadas em figuras ? - Várias pesquisas na área têm concluído que nem sempre interfaces baseadas em figuras são a melhor escolha para se comunicar com o usuário. O maior problema é a dificuldade no projeto e na construção das figuras. O projeto das figuras deve ser feito de forma cuidadosa, para que os usuários entendam o seu significado de forma intuitiva e correta.

2.2.2.2 FIGURA x TEXTO

É sabido que a mente humana é orientada fortemente para aspectos visuais e que seres humanos conseguem perceber e compreender o significado de uma figura de forma mais rápida do que lendo um texto [RAEDER,85]. Dentro da visão de Raeder, serão apresentados aspectos que permitem comparar o uso de figuras e o uso de texto.

Acesso randômico x Acesso sequencial - Os olhos do ser humano permitem acesso randômico a qualquer parte de uma figura e visões detalhadas e total da mesma. Já o texto é sequencial.

Dimensões - Texto é uma cadeia de palavras de uma única dimensão. Figuras são muito mais ricas, permitem ao projetista projetá-las e construí-las em duas ou mais dimensões, usar cor, direcionamento, textura, etc. Dessa forma as linguagens que fazem uso de figuras são mais ricas do que textos; a codificação de cada pedaço da informação pode ser mais compacta, resultando na diminuição do tempo de codificação.

Taxa de transferência - Comparando figuras e textos, torna-se claro que figuras geralmente transferem informações mais rapidamente do que o texto. Tanto o acesso quanto a decodificação são mais eficientes.

Figuras sem Identificação - Os objetos que estão dentro do nosso alcance (visão), tanto aqueles do mundo real quanto os programas baseados em gráficos, podem ser referenciados através do processo de seleção utilizando um mouse, por exemplo, ou

referenciados através de nomes. Nomes introduzem um segundo nível de referência que pode ser um obstáculo. Nas linguagens convencionais de programação, o único caminho para referenciar um objeto é pelo nome. Isto faz com que todos os objetos sejam referenciados indiretamente; eles são invisíveis para o programador. Nomes são desnecessários quando se usa figuras, desde que os objetos estejam presentes.

2.2.2.3 EXEMPLOS

Em seguida serão apresentados dois exemplos de sistemas, onde aplicou-se conceitos discutidos na seção 2.2.2.

UM EDITOR DE DIAGRAMA DE BLOCOS PARA PROJETOS DE SISTEMAS DE CONTROLE

Segundo [FONTANINI,88], a utilização do Editor de Diagrama de Blocos para Projeto de Sistema de Controle tem como finalidade possibilitar ao usuário definir e manipular de forma bastante simples a representação em diagrama de blocos de um sistema de controle. Além da definição e manipulação, é possível também definir funções de transferência de cada componente do sistema, criar interativamente a estrutura em diagrama de blocos do sistema e, conseqüentemente, obter uma representação em termos de entrada e saída.

O editor transforma um diagrama correto em uma ou mais

equações simbólicas empregando técnicas de redução desenvolvidas por Munro-Abrahmas [FONTANINI,88].

Para suportar as características já mencionadas, um ambiente interativo foi construído de maneira a permitir a modelagem de um sistema genérico em termos de diagrama de blocos, podendo conter um número arbitrário de malhas com várias entradas.

Para a construção deste editor, foi utilizado um método que transformasse as representações gráficas em estruturas computacionais manipuláveis. O editor contém um mecanismo para manipulação direta dos elementos geométricos com significado próprio dentro do contexto de sistema de controle. A figura 2.1 exibe os símbolos usados para construção de um diagrama de blocos.

O processamento direto dos elementos geométricos constitui-se em um sério problema. Diante das circunstâncias, foi utilizado um método de transformação do diagrama de blocos em um grafo de fluxo de sinal equivalente e toda manipulação computacional é feita sobre este grafo. A figura 2.2 exibe os símbolos e os subgrafos correspondentes utilizados na conversão.

A interação homem-máquina é feita através de um monitor de vídeo, tendo como auxiliares uma impressora e discos flexíveis. Para construir um diagrama basta posicionar o cursor na tela, escolher o elemento geométrico desejado e inseri-lo no lugar correto. O diagrama pode ser salvo, impresso e

recuperado. Comandos para limpar a tela e eliminar elementos geométricos, entre outros, estão presentes. A interface gráfica sobre a tela do microcomputador foi dividida em quatro áreas. A área de diálogo é responsável pelo envio de mensagens de erros do sistema e informações intermediárias para execução de cada comando. A área de símbolos contém os elementos geométricos primitivos que podem ser usados na construção do diagrama. A área de comandos contém a lista dos comandos que podem ser utilizados e, finalmente, a área de edição, onde efetivamente o diagrama é construído.

Função de Transferência Entrada pela Esquerda ou Direita
Somadores com Saída para Direita
Somadores com Saída para Esquerda
Conexão Tipo "T"
Conexão Tipo "L"
Conexão Horiz. Vertical

Figura 2.1 - Símbolos Utilizados para Construção de Blocos.

Figura 2.2 - Símbolos e seus Respectivos Subgrafos Correspondentes.

INTERFACE GRÁFICA HOMEM-MÁQUINA PARA APLICAÇÃO EM AUTOMAÇÃO E CONTROLE DE PROCESSOS

Segundo [PERKUSICH,88], a necessidade de comunicação eficiente com o operador inclui o acesso a todas as informações existentes no banco de dados do sistema de controle. Perkusich também afirma que uma interface homem-máquina deve dispor de meios que possibilitem a modificação de valores, parâmetros de algoritmos de controle, reconfiguração do sistema de controle, ação manual do operador em válvulas e chaves, etc. A mesma também deve conter funções de sinalização (alarme), verificação das funções de processo e amostragem das tendências das

variáveis do processo. Para que pudesse ter condições de implementar todas as características acima mencionadas, entre outras, o autor utilizou-se de um controlador gráfico colorido de dezesseis bits de alto-desempenho, UPD 7220. Este foi implementado em um cartão Euro Card e barramento ECB (Euro Card Bus). O cartão possui uma memória de imagem de até 512 Kbytes, permitindo uma resolução de até 1024 x 1024 pontos e oito cores ou ainda 4096 x 4096 pontos em preto e branco. A taxa de vídeo é de 15.7 MHz ou 24.8 MHz.

O software desenvolvido consiste de dois programas. O primeiro é uma biblioteca gráfica que implementa sessenta e sete funções gráficas compatíveis com o pacote GKS [NEWMAN,81]. O segundo programa é responsável pela passagem de parâmetros entre a linguagem de alto nível e a biblioteca. Para a edição de programas foi desenvolvido um editor gráfico, figura 2.3, dedicado a aplicações em automação e controle de processos, possuindo uma biblioteca padrão de símbolos gráficos utilizados em plantas de processos. O módulo principal responsabiliza-se pela inicialização da estação de trabalho, apresentação da tela inicial e execução do gerenciamento de funções do editor. Um segundo módulo implementa a função de desenhar a tela inicial. O terceiro implementa o acesso à estação de trabalho e o gerenciamento da interface homem-máquina. Um quarto módulo inicia os parâmetros da estação de trabalho e o último módulo implementa as funções de desenho de primitivas, controle de cursor e gerenciamento da tela.

O referido editor possui funções de edição (alteração de parâmetros, apagar primitivas, etc) e funções de armazenamento e cópia (ler tela gráfica de um arquivo, imprimir tela, etc) conforme mostrado na figura 2.4. Neste editor foram implementadas facilidades para o desenho de quadros resumidos, diagramas unifilares, etc. Outra característica do editor é que o usuário pode ter acesso às primitivas do GKS através dele.

Figura 2.3 - Estrutura do Editor Gráfico.

Figura 2.4 - Tela Principal do Editor Gráfico.

Para a construção desta interface gráfica, Perkusich abordou os seguintes aspectos:

- a. funções do operador;
- b. organização da interface;
- c. codificação da informação; e,
- d. regras para projeto de tela.

2.2.3 AMBIENTES DE DESENVOLVIMENTO DE SOFTWARE

Existem várias abordagens e ferramentas para melhorar as habilidades dos seres humanos no processo de desenvolvimento

de software de alta qualidade a um custo reduzido, mencionando-se:

a. linguagens de programação com mecanismos poderosos para aumentar a produtividade do programador, como por exemplo, ADA;

b. novos paradigmas de programação, fornecendo melhores condições aos programadores no processo de desenvolvimento de software e permitindo melhor estruturação e organização dos programas desenvolvidos;

c. definições corretas dos processos de desenvolvimento de software, que aumentam a qualidade do produto; e,

d. reuso de software existente, que é um bom caminho para reduzir custos do processo de desenvolvimento de sistema.

Com o crescimento do uso de computadores em vários campos das ciências, novos sistemas foram surgindo e, conseqüentemente, os desenvolvedores desses sistemas sentiram necessidade de se organizarem cada vez mais, visando a aumentar a produtividade do processo de desenvolvimento de software, oferecendo um suporte mais eficiente. O uso do computador para apoio ao processo de desenvolvimento de sistemas baseia-se em três razões fundamentais:

a. a possibilidade de automação de várias tarefas que não exigem grandes habilidades por parte dos desenvolvedores;

b. as dificuldades em manter códigos consistentes com as informações de projeto, tais como: necessidades, documentos, estratégias de teste, etc.; e,

c. o potencial das estações de trabalho, com sofisticados

dispositivos de entrada e saída, proporciona redução no custo do software, além de oferecer aos programadores condições para construir interfaces homem-máquina de boa qualidade em menos tempo.

Pesquisas em ambientes de desenvolvimento de software têm mostrado que as mesmas podem ser vistas sob três dimensões:

a) **Atividade** - Define o conjunto de atividades (modelagem, identificação de requisitos, especificação, projeto, implementação, teste, correção e expansão) a ser utilizado nas fases de desenvolvimento e manutenção do software.

b) **Domínio** - Define a classe de software que o ambiente irá suportar.

c) **Mecanismo** - Define os mecanismos que caracterizam a estrutura interna (componentes) e externa do ambiente.

A seguir, alguns ambientes gráficos de desenvolvimento de software são apresentados.

2.2.3.1 HI-VISUAL ICONIC PROGRAMMING

HI-VISUAL é visto como um ambiente de programação icônica, isto é, dirigido por representações gráficas, fornecendo as seguintes facilidades [HIRAKAWA,87]:

a. navegação através de figuras durante o desenvolvimento de programas e nas operações realizadas no sistema;

b. mecanismo de interpretação das figuras utilizadas para confeccionar os programas, baseado no conceito de programação

orientada a objeto [THOMAS,89];

c. projeto de interfaces homem-máquina;

d. desenvolvimento de programa utilizando a abordagem top-down; e,

e. integração de (sub)sistemas existentes.

Neste ambiente, uma figura é a representação de um objeto a ser gerenciado no sistema. A mesma é composta de duas partes. A parte interna dá o significado do objeto que a figura representa, enquanto que a parte externa é a visualização do significado.

HI-VISUAL dispõe de facilidades de programação-por-exemplo e programming-in-the-large, isto é, o ambiente de desenvolvimento de software possui ferramentas que apoiam todo ou quase todo o ciclo de desenvolvimento de uma aplicação. Em programação-por-exemplo, os programas são especificados através de mecanismos de ajuda. Este esquema de programação é efetivo especialmente quando o sistema é aplicado a um ambiente onde as tarefas não são bem definidas.

No modo de programação interativa, o usuário seleciona uma figura do menu e a coloca no lugar apropriado. Esse processo será repetido pelo usuário até que seu programa esteja completo. A interpretação e a execução dos procedimentos que estão por trás das figuras são executadas durante o processo de desenvolvimento do programa. Na figura 2.5 encontra-se um exemplo de desenvolvimento de programa usando procedimentos pré-definidos.

Figura 2.5 - Desenvolvimento de Programas Utilizando Procedimentos Pré-Definidos.

Para o gerenciamento da programação e execução dos programas, as seguintes funções são requeridas:

- a. criação de primitivas;
- b. execução das primitivas;
- c. manipulação de figuras;
- d. classificação e registros das figuras na base de dados de acordo com seus tipos e conceitos;
- e. recuperação de figuras;
- f. esclarecimento das figuras quanto aos tipos de dados de entrada e saída; e,
- g. exibição e ligação das figuras na tela do computador.

A arquitetura do HI-VISUAL, é composta por cinco

módulos básicos:

- a. núcleo;
- b. módulo de inferência;
- c. gerenciador de execução de figuras;
- d. gerenciador do sistema de figuras; e,
- e. ferramentas.

O núcleo possui primitivas para controlar processos, gerenciar dispositivos, gerenciar telas e janelas e gerenciar bases de dados.

O módulo de inferência oferece funções que interpretam as ações do usuário, classificam e gerenciam figuras, respondem às solicitações de ajuda e facilitam a ação de navegação do usuário.

O gerenciador de execução de figuras gerencia a interpretação e a execução das mesmas. O gerenciador do sistema de figuras, além de supervisionar os outros quatro módulos, gerencia várias representações gráficas de objetos exibidos na tela no momento, interpreta a ação do usuário e chama os módulos requisitados pelo usuário.

2.2.3.2 THE TINKERTOY GRAPHICAL PROGRAMMING ENVIRONMENT

Segundo [EDEL,88], Tinkertoy é uma interface gráfica para Lisp, onde os programas são "construídos" ao invés de escritos. Tinkertoy é excitante porque representa uma interface homem-máquina que pode facilmente superar a velocidade de

interação dos melhores editores de textos. O mesmo fornece um modelo consistente de interação através de edição e execução de comandos.

Os programas nesse ambiente são representados graficamente. Estruturas que não podem ser representadas em textos podem ser claramente descritas e novos tipos de informação podem ser incorporados dentro dos programas.

As representações gráficas do Tinkertoy são muito parecidas com as do Apple Macintosh. A diferença é que elas possuem posições de entrada e saída através das quais podem ser conectadas a outras figuras, bem como formarem estruturas.

A edição de programas utilizando este ambiente é muito simples, basta selecionar uma representação gráfica e conectá-la a outras, formando assim uma estrutura. A habilidade para compor e manipular as estruturas usadas no processo de edição serve de comandos para a interface Tinkertoy. Neste ambiente também é permitida a digitação de pequenos textos. O entendimento do fluxo de controle e de estruturas de dados complexas é facilitado, porque a forma gráfica se parece mais com uma estrutura de lista do que uma representação interna. Alguns conceitos de difícil representação no formato textual podem ser facilmente representados neste ambiente.

Tinkertoy oferece também facilidades que não surgem das estruturas e sim das informações (tipo de dados, número de argumentos, etc) que podem ser acopladas às representações gráficas. O maior problema enfrentado pelo Tinkertoy é ser

muito extenso. Este problema tem atrapalhado bastante outras experiências com linguagens de programação gráfica [EDEL,88].

Janelas são muito importantes para este ambiente, porque elas oferecem aos usuários uma visualização separada dos contextos. Apesar da sua grande importância, as janelas apresentam um problema conceitual, ou seja, elas requerem tratamentos especiais para interseção das figuras.

2.2.3.3 VU - VISUAL USER-INTERFACE DESIGN

Segundo [SINGH,90], VU é um ambiente de programação interativo que possibilita ao projetista de interfaces homem-máquina construir interfaces altamente interativas, gráficas e que possam ser manipuladas diretamente.

VU combina programação visual [SHU,88], [SINGH,90] e [GRAFTON,85], projeto orientado a objetos e um sistema de ajuda para ativar as facilidades existentes. Visibilidade, manipulação direta de objetos e capacidade de interação no desenvolvimento da interface são os pontos fortes deste ambiente.

A abordagem convencional de desenvolvimento de software [PRESMAN,87], segundo [SINGH,90], não é adequada para o desenvolvimento de interfaces homem-máquina interativa, porque é bastante difícil de se construir a especificação de uma interface estável antes da implementação. Outro fator agravante é que a abordagem convencional consome bastante tempo e

dinheiro.

O sistema VU fornece ao projetista um caminho mais natural para a especificação dos aspectos gráficos e comportamentais da interface com o usuário e implementação da mesma. Esta abordagem remove as abstrações o mais rápido possível, permitindo ao projetista pensar e executar de uma forma mais visual e direta.

Uma interface gráfica pode ser tratada como uma coleção de objetos, tais como janelas, menus e outras técnicas de interação. O projetista pode dividir sua aplicação em vários objetos. Utilizando a técnica de divisão [SIBERT,86], [BARTH,86] e [LINTON,89], o mesmo pode projetar interfaces simples, consistentes e de alta aceitação. No VU, comandos para direcionar, retirar, mudar o tamanho e designar a cor dos objetos funcionam de forma idêntica para todos os objetos. Isto faz com que se reduzam as chances de inconsistência na interface e aumente a facilidade e a eficiência da implementação do projeto do sistema.

Manuais são muito pouco usados nesse ambiente. O sistema está habilitado a explicar o significado e o uso de todos os comandos e os objetos estão visíveis para o projetista o tempo todo. Cinco são os subsistemas que compõe o VU:

- a) objeto;
- b) atualização;
- c) simulação;
- d) ajuda; e,

e) menu.

O subsistema objeto manipula a hierarquia dos objetos da interface fornecida pelo sistema. O subsistema de atualização permite que vários dos atributos de muitos objetos possam ser modificados. Este módulo suporta uma estratégia de projeto incremental, fornecendo facilidades para o armazenamento parcial do projeto e recuperação do mesmo para possíveis modificações.

VU oferece condições ao projetista para testar a interface como se o mesmo fosse o usuário final. O projetista entra no subsistema de simulação e começa a interagir com a interface segundo algumas restrições.

O sistema de ajuda é projetado para reduzir a necessidade de memorização das funções de cada comando e de qual é a função apropriada para cada objeto.

O desenvolvimento de um sistema em VU é dividido em duas partes. A primeira é utilizada para interagir no desenvolvimento da interface enquanto que a segunda trata do desenvolvimento da parte semântica das rotinas da aplicação. Cada uma das partes pode ser realizada por grupos distintos. A figura 2.6 exibe a sequência do desenvolvimento de uma aplicação interativa usando VU.

Figura 2.6 - Esquema de Desenvolvimento de Aplicação Usando VU.

2.2.4 GERADORES DE APLICAÇÃO

Na início da era computacional a principal preocupação em relação ao desenvolvimento de sistemas baseados em computador era o desenvolvimeto do hardware utilizado nesses sistemas. Devido ao crescimento tecnológico ocorrido na área de microeletrônica, o poder computacional aumentou bastante, a um custo cada vez mais reduzido. Baseado nesse fato, aumentou muito a demanda por sistemas de software que utilizassem esses recursos em toda a sua potencialidade.

Devido à necessidade de acompanhar essa demanda e a ausência de uma forma sistematizada e organizada no processo de desenvolvimento de software, vários problemas ocorreram, entre eles a produção de software de baixa qualidade e o alto custo

de manutenção dos sistemas.

Tentando reduzir esses problemas e estabelecer e usar princípios de engenharia a fim de obter-se economicamente sistemas confiáveis, foi criada a disciplina denominada Engenharia de Software. Nela os custos e a produtividade de software são dois aspectos importantes. Um outro aspecto relevante é a qualidade dos sistemas produzidos. O ideal, então, seria encontrar meios para reduzir os custos e ao mesmo tempo aumentar a produtividade, mantendo um bom nível de qualidade do software.

Segundo [MEIRA,91] os geradores de aplicação são ferramentas de apoio por computador ao desenvolvimento de software que, além de reduzirem os custos e aumentarem a produtividade, ajudam a melhorar a qualidade dos sistemas produzidos. Geradores de aplicação são ferramentas que recebem como entrada especificações em uma linguagem de alto nível e produzem um produto em um nível mais concreto, sem a intervenção do ser humano nesse processo de tradução [MASIERO,93]. O resultado é geralmente um programa de aplicação em uma linguagem fonte, mas poderia ser documentação de software, estruturas de dados, dados de teste, etc.

Os geradores de aplicação oferecem muitas vantagens e algumas dificuldades. Além de aumentarem a produtividade na fase de desenvolvimento de software, os geradores facilitam a fase de manutenção, pois para corrigir um erro de especificação, ao invés de se ter que modificar diretamente o

código fonte, basta modificar a especificação de entrada e fornecê-la novamente ao gerador.

Entre as várias dificuldades inerentes aos geradores de aplicação, as seguintes merecem destaque: eles podem ser usados efetivamente apenas em poucas situações e, freqüentemente, reconhecer onde poder usá-los ocorre em fase adiantada do desenvolvimento, desestimulando o reinício do mesmo. Os geradores de aplicação são difíceis de serem construídos, pois requerem linguagens de especificação cuidadosamente projetadas, um conhecimento íntimo do domínio da aplicação e a habilidade de projetar unidades de software genéricas e confiáveis.

Um gerador de aplicação compreende três módulos básicos: um módulo de interface, um módulo analisador de especificações e um módulo gerador do produto. O módulo de interface não é essencial ao gerador e, nesse caso, a entrada de dados pode ser feita por um editor de texto seguindo as regras definidas pela especificação da linguagem. Estando disponível, aumenta a facilidade de interação homem-máquina e esconde os detalhes da linguagem de especificação e dificuldades potenciais. Esse módulo pode ser implementado no formato de um editor gráfico ou em vários estilos diferentes de diálogo: pergunta e resposta, menus, ou linguagens de comando.

O módulo analisador de especificações armazena toda a sintaxe da linguagem de especificação. O conhecimento semântico está dividido entre este módulo e o módulo gerador do produto. Especificações corretas são aceitas e as rotinas semânticas são

ativadas para armazená-las em um formato interno padrão, que será utilizado posteriormente pelo módulo gerador do produto.

O produto requerido pelo usuário final é obtido no final pelo módulo gerador de produto. Este módulo controla um conjunto de rotinas que permite ter acesso às estruturas de dados internas, criadas pelo módulo analisador de especificações. O módulo gerador de produto também é capaz de analisar e interpretar sua própria linguagem de comando, que indica como manipular a estrutura de dados interna e como formatar o produto final do usuário. Se o produto final for um código em uma linguagem de programação de alto nível de propósito geral, ele pode então ser compilado e executado.

Segundo Masiero e Meira em [MASIERO,93], os geradores de aplicação podem ser classificados através de dois eixos de abstração: o suporte dado a diferentes domínios de aplicação e a abrangência das aplicações que podem ser desenvolvidas com o gerador, isto é, a quantidade de diferentes tipos de aplicações que podem ser geradas. Com base nessa classificação eles dividem os geradores em duas categorias quanto às suas arquiteturas: SDSA (Único Domínio-Única Aplicação) e MDMA (Múltiplo Domínio-Múltiplas Aplicações). Os geradores SDSA são usualmente conhecidos como geradores de aplicação e os MDMA são conhecidos como meta-geradores ou geradores de geradores de aplicação.

A figura 2.7 mostra as arquiteturas dos geradores SDMA e MDMA. Geradores SDSA (figura 2.7a) são específicos, sempre

produzem o mesmo tipo de produto. A maior tarefa do usuário é entrar com a especificação do problema sem ter acesso ao gerador e sem poder reprogramar o seu comportamento. O produto é especificado pelo projetista do domínio usando qualquer técnica disponível para definir o produto a ser gerado. Nesse tipo de gerador todo ou parte do domínio e o conhecimento da aplicação podem estar embutidos no código.

Nos geradores classificados como MDMA o usuário tem liberdade para definir o domínio da aplicação e diferentes aplicações dentro de cada domínio. No modelo apresentado na figura 2.7b, o gerador de aplicação deve oferecer ferramentas para a definição de novos domínios, e essas ferramentas devem ser, e usualmente o são, construídas com suporte de geradores SDSA. Geradores MDMA são mais apropriados para serem utilizados pelos projetistas de domínio, requerendo algum conhecimento de técnicas de compilação e linguagens formais.

Cleaveland em [CLEAVELAND,88] sugere um método para construção de geradores de aplicação composto de sete passos. Eles podem não ser aplicados explicitamente ou na seqüência sugerida, entretanto, constituem em uma abstração útil para construção de geradores de aplicação. Os sete passos são:

Figura 2.7 - Arquiteturas de geradores de aplicação. a, SDSA.b, MSMA

1. Reconhecimento do Domínio - reconhecer o domínio de aplicação é o passo mais difícil durante a construção de um gerador de aplicação. Uma solução para o reconhecimento de domínios é a identificação de padrões. Padrões ocorrem a nível de código, como código repetitivos, rotinas similares, em forma de programas, projetos ou arquiteturas semelhantes.

2. Definição dos Limites do Domínio - esse passo determina o alcance do gerador, definem-se quais características o gerador deve possuir e quais devem ser excluídas.

3. Definição do Modelo Básico - um gerador, provavelmente, é mais compreensível, consistente e completo se for baseado num modelo matemático. Alguns modelos matemáticos são: conjuntos, grafos direcionados e modelos computacionais, como por exemplo, máquinas de estados finitos.

4. Definição das Partes Variantes e Invariantes - uma parte

variante é algo sob controle do projetista de sistema, enquanto que a parte invariante é algo que não deve ser modificado.

5. Definição da Entrada da Especificação - a entrada para os geradores pode tomar várias formas: uma linguagem ou notação textual que dê todas as informações necessárias, diálogos, abordagens interativas(menus, manipulação direta de objetos na tela, linguagens de consulta) e diagramas(redes, fluxogramas ou grafos).

6. Definição do Produto - o produto de um gerador pode ser um programa, uma documentação, estrutura de dados, etc. Esse produto deve ser útil para diversas aplicações do domínio. Sua estrutura e a organização da implementação devem ser genéricas a fim de atender às necessidades de todas as aplicações do domínio.

7. Implementação do Gerador - o último passo consiste em implementar o gerador. O projetista de domínios determina como implementar o gerador a partir dos requisitos fornecidos pelo analista de domínios.

2.3 SISTEMAS PARA AUTOMAÇÃO LABORATORIAL

Nessa seção são apresentados resumos de sistemas utilizados no processo de automação laboratorial.

2.3.1 LABTECH NOTEBOOK

O LABTECH NOTEBOOK é um dos software mais completos para realização dos processos de aquisição, manipulação, análise de dados e controle de processos, apresentando uma quantidade bastante grande de funções [LABTECH,87a e 87b]. O seu processo de aquisição de dados é realizado de dois modos: em velocidade normal e em alta velocidade. No primeiro modo, o computador executa outras tarefas em primeiro plano e a aquisição de dados é colocada em segundo plano. No segundo, o computador é usado integralmente para aquisição de dados.

O controle de processos executado pelo LABTECH NOTEBOOK pode ser efetuado por um controle em malha aberta (chave liga-desliga) ou por um controle em malha fechada (retroalimentação). O controle do tipo malha fechada faz uso dos controladores Integral-Proporcional-Derivado [BERTONUCCI,90]. Ele possui também um mecanismo para exibir e receber dados de outros softwares em tempo real. Incorporado a este software existe um módulo chamado CHROM, que possibilita o interfaceamento de um ou vários cromatógrafos (equipamentos destinados a separar compostos químicos com base no peso molecular) ao computador.

Usando as linguagens de programação Pascal, C, Basic ou a própria linguagem do LABTECH NOTEBOOK (MAGIC/L), pode-se automatizar a aquisição dos dados e o controle de processo. Técnicos de laboratório que não possuem conhecimento de

programação podem utilizar este software para tarefas mais simples, pois a interface é bastante simples, sendo baseada em menus auto-explicativos.

Mecanismos para realização de operações binárias, comparações, funções estatísticas, ajuste de curvas, visualização de dados em tempo real em até quinze janelas e mecanismos para importar e exportar dados são algumas das outras facilidades do LABTECH NOTEBOOK.

2.3.2 ASYST

O ASYST também é considerado um software bastante poderoso. Ele pode ser usado na indústria de automóveis, em bioquímica, cardiologia, química, eletrônica e em outras áreas. Ele é composto por quatro módulos: o primeiro e o segundo compõem o sistema básico, análise de dados e parte gráfica, enquanto que os outros dois são responsáveis pela aquisição de dados e controle de processos. Os pontos fortes deste software são [ASYST,88]:

- a. integração das tarefas de aquisição e análise de dados e emissão de gráficos;
- b. possibilidade de exibir dados em tempo real;
- c. sistema de ajuda on-line;
- d. editor de textos integrado; e,
- e. identificação de erros de programação.

O ASYST permite que instrumentos de laboratório possam

ser facilmente anexados ao microcomputador, bastando para isto que o instrumento seja portador de uma interface de comunicação RS-232-C ou GPIB/IEEE-488. O processo de aquisição de dados é limitado somente pela velocidade de conversão da placa analógica-digital e pela velocidade de acesso ao disco. A análise de dados pode ser feita utilizando-se processos estatísticos, numéricos ou com processamento do sinal analógico. Dentre os vários tipos de gráficos emitidos por este software, encontram-se os de barra e de sobreposição de múltiplos pontos.

O ASYST possui mecanismos para realizar a conversão de e para arquivos do tipo ASCII, BASIC e LOTUS 1-2-3. O mesmo permite o uso de operações aritméticas, envolvendo funções trigonométricas, tipo de dados real, complexo, de precisão dupla e simples e operações baseadas em matrizes.

2.3.3 DADISP

O DADISP é uma planilha eletrônica usado para realizar aquisição e análise de dados [DADISP,88]. Sua interface é baseada em menus e não requer do usuário nenhum conhecimento de programação para usá-la. A análise de dados pode ser feita tanto graficamente quanto numericamente. Uma outra característica importante é o recurso para manusear separadamente sessenta e quatro janelas, cada uma podendo apresentar um tipo diferente de informação. Mecanismos

flexíveis para importar e exportar dados também estão presentes. O módulo DADiSP-488 é responsável pelo controle e transferência dos dados para os equipamentos que possuem uma interface GPIB/IEEE-488.

2.3.4 ASYSTANT +

O ASYSTANT + destina-se a engenheiros e cientistas e tem como objetivo a automação das tarefas de aquisição e análise dos dados. Segundo [WRIGHT,87], a versão básica do ASYSTANT + transforma o microcomputador em uma calculadora sofisticada, enquanto que a versão mais avançada possui facilidades para que se possa controlar a aquisição de dados através do interpretador FORTH [TELLO,84].

A interação com o usuário é feita através de menus, figura 2.8, sendo que quatro deles correspondem a facilidades oferecidas para programação da calculadora. O quinto contém opções para que se possa ter acesso a outras partes do programa, tais como processamento de sinais, emissão de gráficos e ajuste de curvas.

Fazem parte deste software funções que executam análises estatísticas, funções para transformada de Fourier, filtragem de sinais, operação para manipulação de arquivos, geração e processamento de sinais, etc. Uma outra facilidade bastante útil é a presença de um editor de textos que tanto serve para a parte textual quanto para a parte gráfica. O

software permite que sejam utilizados os comandos do sistema operacional MS-DOS sem que haja necessidade de sair do ambiente.

Figura 2.8 - Menus do ASYSTANT +.

O ASYSTANT + possui um módulo para controlar o hardware na tarefa de aquisição de dados. As opções de controle são mostradas na figura 2.9. No módulo calculadora, o ASYSTANT + apresenta um bom desempenho, segundo o fabricante. Quando são usadas as funções avançadas, o desempenho é reduzido. No uso do módulo de aquisição de dados, ASYSTANT + leva em consideração o potencial do microcomputador.

Figura 2.9 - Opções de Controle do Hardware.

2.3.5 LABVIEW

Segundo [LABVIEW,86] e [SANTORI,90], LABVIEW o (Laboratory Virtual Instrument Engineering Workbench) é uma poderosa ferramenta gráfica que oferece aos pesquisadores e técnicos de laboratório com pouca ou nenhuma experiência em programação grandes facilidades para criação de painéis de comandos.

O software LABVIEW transforma o MacIntosh em uma ferramenta de laboratório de propósito geral. Ele trabalha com instrumentos isolados conectados à saída serial do MacIntosh e suporta a interface GPIB-MAC. O LABVIEW possui os seguintes módulos:

- a. um painel frontal, composto de várias representações gráficas para construção de painéis de comandos;
- b. uma linguagem de programação gráfica;
- c. um mecanismo para conectar painéis de comandos, através das linguagens de programação C, Assembler e Fortran;
- d. funções de entrada e saída para ler e gravar dados de/para discos ou portas de comunicação; e,
- e. funções para operações estatísticas e manipulações de matrizes.

O painel frontal contém controles de entrada e saída (mostradores, interruptores e telas de saída) que representam um dado entrando e saindo do instrumento. Durante a formação do painel de comandos, um diagrama de blocos vai sendo contruído. Este conterá o programa a ser executado posteriormente. A figura 2.10 exhibe um exemplo de um painel de comandos e de seu respectivo diagrama de blocos.

Esta ferramenta permite que uma rotina que realize a função de aquisição de dados, escrita em uma das linguagens de programação já referenciadas, seja incorporada ao painel frontal. Para que isto seja possível, é necessário a criação de um ícone para mostrar a ação desempenhada pela rotina e a incorporação da mesma ao painel. A partir deste ponto, a mesma estará disponível para ser utilizada em qualquer lugar.

Figura 2.10 - Painel de Comandos e seu Respetivo Diagrama de Blocos

A linguagem de programação inclusa no LABVIEW chama-se G. Ela faz uso de um conjunto de representações gráficas no lugar de palavras chaves, mas permite uso de pequenos textos. Esta linguagem oferece um conjunto de representações para as estruturas de fluxo de controle (sequência, case, for e while) e um mecanismo para uni-las. Outras características da linguagem são:

- a. a presença de funções aritméticas padrão e especiais (raiz quadrada, geração de números aleatórios, etc);
- b. funções destinadas à entrada e saída, manipulação de

cadeias de caracteres; e,

c. um conjunto de funções para processamento de sinal, correlação, integração, etc.

Dados do tipo real, matriz de real, boolean, matriz de boolean, cadeia de caracteres e matriz de cadeia de caracteres são estruturas de dados encontradas na linguagem de programação do LABVIEW.

2.3.6 SISTEMA DE AQUISIÇÃO DE DADOS - SAD

SAD é um sistema desenvolvido para realizar aquisição e análise de dados pertinentes a experimentos eletroquímicos [TAKAI,87]. Inicialmente, SAD foi desenvolvido para um microcomputador de oito bits, com uma placa analógica/digital AD574. O equipamento de laboratório conectado ao microcomputador é um potenciostato. A interface AD574 no SAD não só realiza as leituras das informações como envia um byte de informação que, se multiplexado, pode enviar 256 sinais de um bit para equipamentos externos. A primeira versão do SAD foi projetado para suportar:

a. uma série de operações que não pertencem à configuração básica do sistema (rotinas de manipulação de arquivos e exibição de diretório de um disco);

b. um conjunto de rotinas para implementar modelos matemáticos pré-fixados; e,

c. facilidades de escolha de relatórios segundo o modelo

matemático processado.

Em uma segunda versão do SAD [TAKAI, 88a] utilizou-se um microcomputador IBM PC, um potenciostato/galvanostato modelo 273 da EGG - Princeton Applied Research e uma interface de comunicação padrão internacional IEEE-488. Para esta versão, foi necessário desenvolver:

- a. um módulo que permitisse editar um programa na linguagem 273, linguagem de programação utilizada no Potenciostato/Galvanostato;
- b. mecanismo com capacidade de armazenar a programação feita em 273;
- c. mecanismo para enviar para o equipamento a programação estabelecida;
- d. dispositivo para ativar o experimento; e,
- e. mecanismo para analisar e imprimir dados e gráficos.

Com o objetivo de atender aos quatros primeiros itens relacionados anteriormente, foi necessário implementar um Sistema de Interpretação de Interface (SII) [TAKAI, 88b]. Para manipulação do SII foi criada uma linguagem voltada para programação de experimentos. Esta tinha como objetivo estabelecer um ambiente de trabalho computacional para que o usuário do sistema pudesse interagir de maneira bastante simplificada com o computador e o equipamento de laboratório em questão, através da interface GPIB.

A linguagem de programação desenvolvida apresentava como características principais dois tipos de procedimentos. O

primeiro, inerente à programação, é utilizado para definir procedimentos da própria linguagem, e o segundo, inerente à interface GPIB, é utilizado para definir um conjunto de comandos do equipamento a ser enviado através da interface GPIB.

A linguagem é composta de um conjunto de símbolos especiais (palavras reservadas, identificadores de procedimentos, funções, comandos ou variáveis), constituindo o vocabulário da linguagem. O SII é constituído dos seguintes módulos:

- a. interface;
- b. linguagem; e,
- c. avaliação parcial das sintaxes dos comandos.

O módulo de interface tem como objetivo enviar comandos ao equipamento interligado pela interface GPIB. O módulo de linguagem é responsável por carregar um programa do usuário, codificá-lo e executá-lo. Com relação ao terceiro, trata-se de um módulo de verificação da sintaxe dos comandos de forma local. Seja, por exemplo, o comando `IF a < 2 Then...`, localmente a sintaxe esta correta mas, globalmente, pode não estar porque a definição dessa linguagem obriga que todo IF possua um comando `End` no final.

2.4 ANÁLISE COMPARATIVA DOS SISTEMAS PARA AUTOMAÇÃO LABORATORIAL

A figura 2.11, apresenta um resumo das principais

características dos sistemas de automação laboratorial revisados neste capítulo. Com base neste quadro conclui-se que os sistemas são, em geral, ferramentas de boa qualidade e que possuem um conjunto de rotinas bastante específicas. Na maioria deles o estilo de interação homem-máquina adotado é o de menus auto-explicativos.

Também ficou claro que os sistemas de automação mencionados não oferecem flexibilidade com relação ao item interface de comunicação. Se, por exemplo, um sistema foi projetado para enviar e/ou receber dados de um equipamento que possui uma interface GPIB e se for necessário utilizar o sistema em outro equipamento com interface de comunicação diferente, os sistemas apresentados não irão atender, pois não possuem mecanismos que permitam de forma simplificada informar que uma nova interface de comunicação será utilizada para estabelecer a comunicação entre o computador e o equipamento.

Conforme mostrado na figura 2.11 vários sistemas possuem linguagem de programação própria, exigindo do usuário conhecimentos de programação caso ele queira usar outras funções não disponíveis. Baseado no que foi apresentado sobre os sistemas de automação laboratorial, pode-se dizer que eles têm condições de atender de forma satisfatória um conjunto abrangente de tarefas que são realizadas nos laboratórios de

SISTEMAS	C A R A C T E R Í S T I C A S								
	A.D	L.P.P	I.U	C.P	I.C	E.T	Aq.D	E.P	
LABTECH NOTEBOOK	SIM	SIM	MENU	SIM	MODULO CHROM	NAO	SIM	SIM	
ASYST	SIM	--	MENU	SIM	RS-232C OU IEEE-488	SIM	SIM	SIM	
DADISP	SIM	NÃO	MENU	SIM	IEEE-488	--	SIM	NÃO	
ASYSTANT +	SIM	NÃO	MENU	SIM	----	SIM	SIM	NÃO	
LABVIEW	SIM	SIM	FIGURAS	SIM	GPB-Mac	SIM	SIM	SIM	
SAD	SIM	SIM	MENU E LING. O. a Sintaxe	SIM	IEEE-488	NÃO	SIM	SIM	

LEGENDA : --- NÃO INFORMADO
 A.D ANÁLISE DE DADOS
 L.P.P LINGUAGEM DE PROGRAMAÇÃO PRÓPRIA
 I.U INTERFACE COM USUÁRIO
 C.P CONTROLE DE PROCESSO
 I.C INTERFACE DE COMUNICAÇÃO
 E.T EDITOR DE TEXTO
 Aq.D AQUISIÇÃO DE DADOS
 E.P EXIGE PROGRAMAÇÃO
 O ORIENTADA

FIGURA 2.11 - PRINCIPAIS CARACTERÍSTICAS DOS SISTEMAS ABORDADOS NA SEÇÃO 2.3.

cardiologia, eletrônica e na indústria de automóvel.

Baseado nas definições apresentadas para geradores de na seção 2.3 não se enquadram na definição de geradores de aplicação conforme [MASIERO,93] e [CLEAVELAND,88], mas podem ser vistos como linguagens de muito alto nível ou linguagens de quarta geração no sentido usado por [MARTIN,85]. Já o SAL, é um sistema destinado a atender um domínio bem específico, gerando um código que é executado dentro do ambiente, podendo esse código ser armazenado e executado posteriormente. Embora o SAL também apresente características de linguagem de quarta geração, será aqui classificado como um gerador de aplicação, em função do SAL gerar um código fonte na linguagem definida para esse domínio de aplicação, conforme mostrada no apêndice, podendo este ser armazenado e posteriormente utilizado. É importante lembrar que essa distinção não é muito nítida, mas a classificação como um gerador de aplicação influencia o método como o SAL será desenvolvido e sua arquitetura geral.

Como pode-se notar na figura 2.11 todos os sistemas apresentam um mesmo núcleo básico composto dos processos de aquisição e tratamento de dados. As diferenças estão nas rotinas complementares: controle de processo, cálculos utilizando uma planilha, etc. Entre todos os sistemas apresentados, o LABVIEW é o que possui um conjunto de idéias que pretende-se adotar no SAL.

2.5 CONSIDERAÇÕES FINAIS

A revisão bibliográfica mostrou que vários caminhos estão abertos ou estão se abrindo para acolher novas propostas, idéias, conceitos e abordagens para construção de software.

O estudo referente a sistemas destinados à automação de laboratórios apresentados na seção 2.3, embora não abrangendo todos os aspectos, mostrou ser satisfatório para indicar quais características básicas podem ser melhoradas, assim como os aspectos relacionados à interação homem-máquina, principalmente quando estes são utilizados por pessoas com pouco ou nenhum conhecimento de programação. Conforme foi apresentado, o número de interfaces de comunicação e suas características são bastantes variadas, tornando assim a construção de um protocolo de comunicações que atenda mais que um tipo de interface de comunicação uma tarefa não muito simples.

Devido ao crescente uso de computadores em vários ramos da ciência, diversos tipos de sistemas interativos estão sendo criados. Em função disso, diversas técnicas e dispositivos de interação e conceitos estão sendo utilizados, tornando assim o uso do computador mais agradável e reduzindo o trauma que muitos usuários ainda possuem quando interagem com esta ferramenta. Com o surgimento de novas técnicas e dispositivos de interação e novos conceitos aplicados na interação homem-máquina, cada vez mais pessoas sem nenhum conhecimento de programação estão resolvendo partes dos seus problemas sem

precisar do auxílio de um especialista em computação.

Atualmente vários profissionais de outras áreas estão utilizando computadores. Diante deste quadro é necessário cada vez mais que os projetistas e implementadores de sistemas interativos conheçam a respeito do contexto onde o sistema irá atuar, qual o grau de instrução e de conhecimento do usuário, qual a periodicidade de uso do sistema, etc. Para isto é necessário que o projetista e o implementador do sistema tenham uma cultura bastante ampla ou recorram a especialistas, em outras áreas como em psicologia, ergonomia, etc.

A interação gráfica fornece vários benefícios sobre os métodos convencionais de interação, tais como, de edição e expansibilidade de programas e execução de comandos. O uso de figuras trouxe uma melhoria significativa aos métodos de entrada de comandos, em virtude da edição ser elaborada de forma simplificada. Para a execução de uma ação, basta selecioná-la com o mouse e em seguida, a ação será disparada.

Em relação aos ambientes gráficos de desenvolvimento de software discutidos, ficou claro que um dos seus objetivos é permitir que pessoas com pouco conhecimento de programação possam criar suas próprias aplicações de forma simplificada e rápida. Estudos e ferramentas estão sendo desenvolvido na área de Engenharia de Software de forma a permitir ao programador de aplicações aumentar cada vez mais a qualidade do software desenvolvido e sua produtividade, reduzindo os custos de desenvolvimento e manutenção de um sistema. Um dos fatores que

contribui para isso são as ferramentas integradas de desenvolvimento de software que estão sendo construídas, apoiando o construtor de sistemas em várias fases do ciclo de desenvolvimento do software.

CAPÍTULO 3

ESPECIFICAÇÃO E DESENVOLVIMENTO DO SISTEMA DE AUTOMAÇÃO LABORATORIAL - SAL

3.1 CONSIDERAÇÕES INICIAIS

Neste capítulo descrevem-se os passos e decisões de projeto para o desenvolvimento do Sistema de Automação Laboratorial, permitindo ao leitor ter uma idéia de sua implementação.

Inicialmente, na seção 3.2, são apresentados os requisitos gerais do sistema, descrevendo-se a metodologia utilizada para desenvolvimento do SAL. A seção 3.3 apresenta de que forma cada requisito mencionado na seção anterior é atendido. Na seção 3.4, são apresentados os módulos básicos que compõe o sistema. Finalmente, na seção 3.5, são mostrados os aspectos relacionados à configuração do protótipo implementado.

3.2 REQUISITOS DO SISTEMA DE AUTOMAÇÃO LABORATORIAL

O termo domínio de aplicação engloba vários domínios específicos, bem como domínios mais abrangentes, com grande potencial para reuso de componentes tais como formatação de relatórios, cálculos estatísticos e sistemas de base de dados.

Considera-se que um domínio é determinado por uma linguagem restrita [BENTLEY,86]. A necessidade de definir uma nova linguagem restrita ou estendê-la para especificar necessidades de um novo problema requer mudança no domínio. Nesse caso tem-se um domínio mais amplo. Isso implica na disponibilidade de ferramentas especialmente construídas que dêem ao usuário condições de codificar diretamente suas necessidades em uma linguagem de propósito geral.

Utilizando os sete passos sugeridos por Cleaveland [CLEAVELAND, 88] para desenvolver um gerador de aplicação, deu-se então início à definição dos requisitos para o sistema de automação laboratorial.

1. Reconhecimento de Domínios - o domínio de automação laboratorial é bastante abrangente. Dessa forma procurou-se estudar um sub-domínio composto de operações que são executadas com constância, formadas por blocos bem definidos e que podem ser agrupadas dando origem a um programa. Após investigações em diversos laboratórios da EMBRAPA, concluiu-se que um conjunto de operações formando um domínio de aplicação coerente é composto de:

- a. definição de equipamento, interfaces de comunicação e parâmetros relacionados a ela;
- b. definição de dados para o processo de coleta de dados;
- c. coleta de dados;
- d. cálculo de estatísticas;

- e. emissão de gráficos; e,
- f. armazenamento e leitura de dados.

2. Definição dos Limites do Domínio - os limites do domínio da aplicação adotados para o Sistema de Automação Laboratorial estão diretamente ligados ao conjunto de operações mencionadas anteriormente. Os tipos de estatísticas limitam-se a: média, variância, desvio padrão, moda, mediana, correlação, regressão, máximo e mínimo. Os tipos de gráficos são: barra, barra 3D, pontos, linhas e setores(torta). Para que esse domínio torne-se mais abrangente, o domínio de controle de processo deveria estar contido no SAL. Por requerer novas investigações, estudos em teoria de controle de processos e experiência nessa área, optou-se por excluí-lo desse primeiro protótipo, e incluí-lo nas próximas versões do SAL.

3. Definição do Modelo Básico - o modelo básico definido para permitir que essas operações possam ser agrupadas e entendidas como parte de um programa, é uma linguagem formal descrita em notação BNF [], anexo 1.

4. Definição das Partes Variantes e Invariantes - as partes invariantes desse sistema são as rotinas destinadas a executar as operações já mencionadas acima. As partes variantes são as informações que essas rotinas necessitam para serem executadas. Elas são passadas às partes invariantes através de opções selecionada dos menus, seleção de botões ou digitação de pequenos textos.

5. Definição da Entrada da Especificação - definiu-se que para o usuário especificar uma aplicação nesse sistema, ele terá a sua disposição um módulo de interface interativo, baseado em janelas, selecionando botões, menus e formulários.

6. Definição do Produto - os produtos gerados por esse sistema são relatórios(listagens e gráficos), cálculos estatísticos e dados, que podem ser armazenados ou lidos de/para um arquivo.

7. Implementação do Gerador - conforme discutido em [MEIRA,91] e [MASIERO,93] várias são as arquiteturas existentes para um gerador de aplicação. A arquitetura escolhida para o SAL foi a SDSA, mostrada na figura 2.7a. Optou-se por essa arquitetura porque: 1) não estava disponível um gerador do tipo MDMA que pudesse ser instanciado; 2) necessidade de se ter um sistema eficiente e 3) o sistema gera sempre o mesmo tipo de produto. O módulo de interface no SAL faz parte explicitamente da arquitetura, de acordo com as decisões tomadas no passo 5 acima.

De acordo com esse primeiro conjunto de informações, mostra-se através da figura 3.1 uma visão geral do domínio de aplicação para o SAL. No decorrer deste capítulo maiores detalhes sobre as decisões de implementação são apresentadas.

Numa segunda etapa, outros requisitos para o sistema de Automação Laboratorial foram definidos, buscando-se responder às seguintes questões:

Figura 3.1 - Uma Visão Geral do Domínio de Aplicação para SAL.

1. Quais as reais necessidades dos técnicos de laboratório ?
2. Qual o grau de conhecimento dos técnicos de laboratório em relação ao computador e à programação?
3. Quais os equipamentos que podem ser interconectados a um microcomputador tipo IBM/PC e suas características ?
4. Que tipos de interfaces de comunicação e placas

conversoras analógica/digital são utilizadas nos equipamentos e quais informações são necessárias para se estabelecer a comunicação entre equipamento e o computador ?

5. Que procedimentos são executados de forma rotineira e manual e são comuns entre os laboratórios e experimentos, oferecendo condições de serem automatizados ?

6. Qual a melhor forma de interação homem-máquina no sistema como um todo ?

Além das visitas aos laboratórios existentes na EMBRAPA, outros mecanismos foram utilizados para levantar esse conjunto de informações:

- a. entrevistas diretas com os técnicos dos laboratórios;
- b. conversas informais; e,
- c. leituras de manuais dos equipamentos, artigos, etc.

Baseado nas informações levantadas, constatou-se que para satisfazer os requisitos de um sistema para automação laboratorial, é necessário que o mesmo satisfaça os seguintes requisitos gerais:

1. Possua um ambiente de trabalho que possibilite ao usuário interagir com o sistema como um todo de forma bastante simples.
2. Disponha de um editor de texto simples que permita ao usuário armazenar informações relevantes ao programa e ao experimento a serem desenvolvidos.
3. Forneça um conjunto de serviços para manipulação de arquivos de dados.
4. Possua um método de composição de programas que seja

fácil e simples e que não requeira do técnico de laboratório conhecimentos sofisticados de computação.

5. Possua uma forma de execução de programa simples e transparente para o usuário.

3.3 ASPECTOS SELECIONADOS DO PROJETO DO SAL

Para atender ao primeiro requisito criou-se um ambiente de trabalho composto por seis módulos (Gerenciador do Sistema, Editor, Arquivo, Compor, Executar e Encerrar), com o objetivo de proporcionar ao técnico de laboratório facilidades de interação com sistema como um todo.

Para ativar cada um dos módulos do sistema, basta o usuário selecionar, com auxílio do mouse, cada uma das opções dispostas em uma barra de menu na janela principal do sistema, e em seguida, uma nova janela será exibida contendo um conjunto de novas opções, se for o caso.

Com relação às facilidades de interação homem-máquina, optou-se por utilizar no sistema todo as idéias e conceitos de interação denominados de **Abordagem Interativas** [SANTOS,89].

Com o objetivo de atender ao segundo requisito, foi criado um pequeno editor de texto (EDSAL) que permite ao usuário editar, apagar, mostrar, gravar e recuperar textos construídos utilizando o SAL.

Para ativar a entrada de textos, basta acionar através do mouse a opção **Editar**, que se encontra no topo da tela, e em

seguida informar o número da linha a ser digitada. Ao término da digitação de cada linha, um novo número aparecerá automaticamente, informando ao usuário qual a próxima linha a ser digitada. Para encerrar a digitação de um texto, basta ao usuário pressionar a tecla <enter> ao lado do número e imediatamente o processo será desativado. O número de cada linha no processo de edição tem um papel importante, pois é através do mesmo que pode-se eliminar ou incluir novas linhas.

Para armazenar um texto, definiu-se uma lista duplamente encadeada, porque essa é uma estrutura de dados dinâmica. A representação dessa estrutura de dados é mostrada a seguir.

```
struct line {  
    char texto[81]; /* linha de texto */  
    int num;        /* numero da linha */  
    struct line *next; /* ponteiro p/ próxima linha */  
    struct line *prior; /* ponteiro p/ linha anterior */  
};
```

O EDSAL permite inserir linhas em qualquer ponto do texto. O meio geral de operação deste editor baseia-se em lista duplamente encadeada ordenada. Além de inserção de linha facilmente no meio do texto, especificando o número da mesma, pode-se também eliminá-la com a mesma facilidade. Uma vez removida ou incluída uma linha, um mecanismo de renumeração de linhas é ativado, permitindo assim manter o texto sempre ordenado. Para essa versão do EDSAL facilidades tais como palavras e/ou textos sublinhados, negritos, inclusão de

caracteres no meio de uma linha entre outros recursos comuns encontrados nos editores de texto conhecidos, não encontram-se disponíveis.

Para exibir um texto, basta acionar a opção **Mostrar** que se encontra no topo da tela. Em seguida será mostrado o número da linha e o texto correspondente.

Para Gravar ou Recuperar um texto, basta acionar as respectivas opções, e posteriormente, informar o nome do arquivo. A opção **Abandonar**, uma vez acionada, fará com que o EDSAL seja desativado e o controle do sistema retornará ao módulo principal do sistema.

Visando proporcionar ao usuário condições para que ele possa manipular arquivos de dados, o terceiro requisito, criou-se um mecanismo que ao ser ativado, através da opção **Arquivo**, permite apagar arquivo, ver o conteúdo de um diretório e copiar um arquivo de dados de/para o disco, bastando para isto informar o nome do mesmo.

Para atender ao quarto requisito, composição de um programas, inicialmente definiu-se um conjunto de funções que representa atividades comuns no processo de automação de experimentos laboratoriais. Todas as funções são representadas externamente através de botões. O referido conjunto divide-se em dois sub-conjuntos: funções-programa e funções-serviço.

As funções-programa são utilizadas na composição de programas, enquanto que as funções-serviço são utilizadas para auxiliar no processo de composição de programas. O conjunto de

funções- programa é formado por:

Coleta - coleta de dados de um equipamento.

Gráfico - emite gráficos a partir dos dados coletados ou lidos de um arquivo.

Estatística - análise estatística dos dados a partir dos dados coletados ou lidos de um arquivo.

Lê - lê um arquivo de dados previamente coletados.

Grava - grava em arquivo os dados coletados durante um experimento.

Exibe - lista os dados coletados ou lidos de um arquivo na tela.

Liga - liga um equipamento de laboratório especificado no sistema.

Desliga - desliga um equipamento de laboratório especificado no sistema.

Calibra - calibra um equipamento previamente definido.

Define - define um equipamento, uma interface de comunicação e os valores para os parâmetros da respectiva interface.

Fim - encerra a construção de um programa.

O conjunto de funções-serviço é formado por:

Ajuda - ativa o sistema de ajuda.

Apaga - apaga totalmente o programa que está sendo construído.

Confirma - confirma a função escolhida e as informações a ela atribuída.

Cancela - cancela a função escolhida e as informações a ela

atribuída.

Em seguida à definição das funções programa e serviço, definiu-se uma linguagem de programação intermediária com o objetivo de permitir a execução de um programa. Para especificá-la utilizou-se a notação BNF. A definição da gramática da linguagem encontra-se no Apêndice 1.

Posteriormente, regras foram especificadas para a formação de programas. São elas:

1. Para elaborar um programa que venha a utilizar-se de um equipamento que permita enviar e receber dados de/para um computador, deve-se proceder da seguinte forma:

a. Em primeiro lugar, especifica-se a parte fixa do programa. Entende-se por parte fixa do programa um conjunto de funções vitais para o bom funcionamento do programa e que devem estar dispostas na seguinte ordem: 1) definição do equipamento, interface de comunicação e valores para os parâmetros da interface, caso seja preciso; 2) indicação para acionar o equipamento; 3) calibração do equipamento e 4) definição de coleta de dados.

b. Em seguida, especifica-se a parte variável do programa. Entende por parte variável um conjunto de ações que, dependendo da necessidade, fará ou não parte do programa. Nessa parte a ordem de especificação fica a critério do usuário. Dentro da linguagem especificada, as funções de cálculo de estatística, geração de gráfico, exibição de dados, gravação de dados e desligar o equipamento podem ser utilizadas.

c. Por último, indica-se o encerramento do programa.

2. Para elaborar um programa que venha a utilizar-se de um equipamento que somente envie dados para o computador, deve-se proceder da seguinte forma:

a. Em primeiro lugar, especifica-se a parte fixa do programa. Essa deve conter os funções dispostas na seguinte ordem: 1) definição do equipamento, interface de comunicação e valores para os parâmetros da interface, caso seja preciso; 2) calibração do equipamento e 3) definição de coleta de dados.

b. Em seguida, especifica-se a parte variável do programa. Nessa parte a ordem de especificação fica a critério do usuário. Dentro da linguagem especificada, as funções de cálculo de estatística, geração de gráfico, exibição de dados, gravação de dados.

c. Por último, indica-se o encerramento do programa.

3. Se o programa a ser criado inicia a partir da leitura de dados que estão armazenados em um arquivo, a regra válida para formação do programa é a seguinte:

a. Em primeiro lugar deve vir a identificação de leitura de arquivo de dados.

b. Em seguida, as funções de cálculo de estatística, geração de gráfico, exibição de dados e gravação de dados, dispostas de acordo com a necessidade do usuário.

c. Por último, indica-se o encerramento do programa.

Vale ressaltar que o aumento dessas regras está atrelado ao crescimento do número de equipamentos que o sistema

tem condições de suportar e também à necessidade do usuário e aos tipos de experimentos laboratoriais. O aumento das funções-programa, principalmente, acontecerá se surgir, por exemplo a necessidade de um novo tipo de gráfico ou nova técnica de análise estatística, etc.

O método de construção de programas adotado no SAL baseou-se nos métodos adotados em [EDEL,88] e [HIRAKEWA,87]. Para compor um programa utilizando o SAL, os seguintes passos devem ser seguidos:

1. Inicialmente ativa-se o módulo Compor.

2. Posteriormente, seleciona-se uma das funções-programa, e em seguida são fornecidas as informações solicitadas. Após a escolha da função, o botão correspondente a ela será destacado (iluminado). O processo de destacamento é utilizado para informar quais funções-programa foram selecionadas no processo de composição do programa. Tanto a identificação interna da função quanto as informações pertinentes à mesma, e também a ordem de execução das funções, são armazenadas na memória RAM, para posteriormente serem utilizadas pelo módulo de execução de programas. Este processo repete-se até que o botão que representa a função de fim de composição de programa seja selecionado.

O método de composição de programa exposto foi adotado em função de proporcionar ao usuário facilidades durante o processo de composição de um programa, isto é, não exige do mesmo memorização de comandos e também por ser de fácil

aprendizado e uso, reduzindo o tempo de construção de um programa e a quantidade de erros que são cometidos quando utiliza-se o método de programação tradicional, isto é, conhecimento de uma linguagem de programação, digitação de um programa, compilação e geração do código executável.

Para dotar o SAL de flexibilidade no aspecto de interfacemento equipamento-computador, isto é, permitir modificar o tipo de interface de comunicação e os valores dos parâmetros relacionados à interface, criou-se a função Define. Para especificá-la, elaborou-se uma tabela que mapeia o relacionamento existente entre o equipamento e a interface de comunicação a qual o equipamento suporta. Com base nas informações adquiridas durante a fase de levantamento, elaborou-se a seguinte tabela:

Equipamentos	Interface de Comunicação
HPLC	RS232C
Instron	AI-13
Teclado	Placa Principal
Secador	AD5812

Tabela 3.1 - Relacionamento entre Equipamento e Interface de Comunicação.

Em uma segunda etapa, foram definidos os parâmetros das

interfaces de comunicação que podem ter seus valores alterados. Para as interfaces de comunicação AD5012 e RS232C os parâmetros são:

RS232C	AD5012
Canal de Comunicação	Número do Cartão
Taxa de Comunicação	Tempo
Paridade	Modo
Número Stop-Bit	Carbo
Número de Bit	Contador
_____	Canal
_____	Canal Inicial
_____	Canais
_____	Tamanho

Tabela 3.2 - Interface de Comunicação e seus Parâmetros.

O processo atual de inclusão de novos tipos de interfaces de comunicação e definição dos parâmetros com seus respectivos valores só pode ser realizado alterando-se o código fonte. Na próxima versão do SAL pretende-se elaborar um mecanismo independente para resolver esse problema.

Terminado o processo de mapeamento e identificação dos parâmetros, projetou-se o seguinte mecanismo de execução para a função Define.

Ao selecionar a função Define, um menu de equipamento é

apresentado. Nele encontram-se os nomes dos equipamentos que poderão ser interligados ao computador. Definido o equipamento, um novo menu, de configuração, é exibido, possibilitando ao usuário informar ao sistema se deseja ou não definir uma nova configuração, isto é, especificar uma nova interface de comunicação para o equipamento e alterar os valores dos parâmetros da interface definida.

Havendo necessidade de configuração, um novo menu, o de Interface, é apresentado. Nele encontram-se os tipos de interface de comunicação que o sistema suporta. Definida a interface, uma nova janela contendo os parâmetros da interface de comunicação escolhida é apresentada, permitindo ao usuário informar quais os novos valores para os parâmetros. Confirmadas as entradas, elas são armazenadas em arquivos de dados para posteriormente serem utilizadas no processo de configuração da interface.

Não havendo necessidade de configuração, o processo de definição é desativado. Nesse caso, a interface de comunicação a ser configurada para possibilitar a comunicação entre o equipamento e o computador é a que está associada ao nome do equipamento e os valores utilizados no processo são aqueles que estiverem no arquivo de dados de mesmo nome da interface.

Um outro serviço importantíssimo que se encontra no módulo Compor é o sistema de ajuda. O mesmo é ativado através do botão Ajuda. Após ativá-lo, define-se de qual função se necessita obter informação, e em seguida uma explicação será

mostrada na tela do computador. Para desativar este serviço basta selecionar o botão Cancela.

Existem várias abordagens para execução de um programa, entre elas estão: uso de interpretador; transformação de código fonte em código executável, e as formas visuais dos ambientes gráficos de desenvolvimento de software. Baseado nas idéias apresentadas, projetou-se a seguinte forma de execução de programas criados através do SAL.

Inicialmente, é verificado se a composição do programa está correta. Baseado no nome do equipamento especificado, identifica-se qual regra de formação de programa deve ser seguida. Uma vez aprovada a regra de formação do programa, a estrutura de dados onde encontra-se registrada a ordem de execução das funções que compõe o programa começa ser lida. Identificada uma função, os parâmetros a ela atribuídos são aceitos e imediatamente ela entra em ação. Os resultados gerados podem ser imediatamente exibidos na tela do computador ou ser armazenados em memória ou em disco para serem utilizados posteriormente. Esse processo é executado até que seja encontrada a marca de fim de programa. Em seguida, uma listagem do programa será exibida na tela do computador e também será armazenada em disco se o usuário desejar.

A escolha dessa estratégia de execução deve-se ao fato de que ela é simples, rápida e transparente. A estratégia de execução de programa é rápida porque informações passadas durante o processo de composição de programa são utilizadas

pelo processo de execução sem necessidade de tratamento, ao contrário do método de execução de programa que exige transformar um código fonte em um código executável. É dito também transparente pois uma vez que o usuário tenha especificado o programa, ele não se preocupa mais com detalhes para executá-lo. Ao invés de se verificar se uma determinada regra de formação do programa está sendo cumprida durante a fase de composição, optou-se por verificá-la na fase de execução, para que o usuário tenha mais liberdade na hora de compor o programa. O algoritmo básico é mostrado a seguir.

Início

Verificar se o programa foi bem formado

Se (aprovado) então

 i = 0

 Ler vetor_comando[i]

 Enquanto (vetor_comando[i] != fim) faça

 Se (vetor_comando[i] = coleta) execute coleta de dados

 Se (vetor_comando[i] = estatística) execute estatística

 Se (vetor_comando[i] = gráfico) execute gráfico

 Se (vetor_comando[i] = exibe) exibe dados

 Se (vetor_comando[i] = le) ler arquivo de dados

 Se (vetor_comando[i] = grava) grava arquivo

 Se (vetor_comando[i] = liga) liga equipamento

 Se (vetor_comando[i] = desliga) desliga equipamento

 Se (vetor_comando[i] = calibra) calibra equipamento

 i = i + 1

 ler vetor_comando

 Fim Enquanto

 Emite listagem do programa

 Deseja gravar o programa(S/N)

 Se (S) então grava o programa

 senão

 emite mensagem de erro

 Fim se

Fim

Verificar se um programa foi bem formado significa

observar se a composição dele segue uma das regras de formação de programa especificadas.

3.4 ARQUITETURA DO PROTÓTIPO

De forma a permitir uma boa estruturação do protótipo, dividiu-se o mesmo em vários módulos, que por sua vez se dividem em vários outros módulos, facilitando a implementação e a manutenção. A figura 3.2 exhibe a arquitetura do protótipo. A seguir, são descritos cada um dos módulos componentes do mesmo.

Módulo Gerenciador do Sistema (MGS) - Tem como função gerenciar as chamadas para outros módulos do sistema. Ao ser ativado através da sigla SAL, o MGS exhibe na tela do computador uma janela dividida em três áreas. Na primeira encontra-se o título do sistema. Na segunda, encontra-se uma barra de menu, contendo cinco botões, que representam as ações principais do sistema. A terceira destina-se à área de trabalho.

Para que o técnico de laboratório ative os outros módulos do sistema, basta que ele selecione o botão desejado e em seguida uma outra ação será disparada.

Módulo Editor de Texto (EDSAL) - Tem como função permitir a edição de pequenos textos. O EDSAL, é composto de cinco funções. São elas: Editar, Eliminar, Mostrar, Gravar e Recuperar textos. Para ativar cada uma das funções utiliza-se o mouse. A descrição do funcionamento de cada uma das funções encontra-se no item 3.3.

Módulo de Operações em Arquivos (MOA) - Tem como função permitir ao técnico de laboratório executar serviços para manipulação de arquivos armazenados em disco. Uma vez ativado, um menu contendo quatro opções é exibido. Escolhidas as opções Copiar ou Apagar, uma janela é exibida, solicitando ao técnico que digite o nome do arquivo. Uma vez confirmado o serviço escolhido, este será ativado imediatamente. Definida a opção de exibir diretório (Dir), será mostrada na tela do computador a listagem do diretório corrente. Escolhida a opção Retornar, este módulo é desativado, retornando o controle de sistema para o MGS.

Figura 3.2 - Arquitetura do Protótipo.

Módulo de Composição de Programa (MCP) - Tem como função permitir ao usuário compor um programa através do mecanismo especificado na seção 3.3.

Módulo de Execução de Programa (MEP) - Tem como função executar um programa construído através do MCP. O mecanismo de execução de programa e o porque da sua escolha também encontram-se descritos no item 3.3.

Módulo de Encerramento de Sistema (MES) - É responsável por encerrar a execução do sistema, limpando a tela do computador e retornando para o sistema operacional.

Módulo Base de Dados - É responsável por armazenar os dados gerados por um programa construído e executado através do SAL, armazenar textos gerados através do EDSAL e um conjunto de outras informações que se fazem necessárias para um bom funcionamento do sistema.

3.5 CONFIGURAÇÃO DO PROTÓTIPO

Os recursos básicos para implementação foram: um microcomputador IBM/PC contendo um monitor monocromático CGA (640 x 200 pontos), 1 MByte de memória RAM, sistema operacional MS-DOS 3.3, um mouse de três botões, um drive de 5 1/4" e um winchester de 40 MBytes.

A linguagem de programação escolhida foi a Turbo C, por possuir um conjunto de recursos que permite a construção de blocos de programas para manipular informações de baixo nível,

construções estas bastante utilizadas nos sistemas destinados ao processo de automação laboratorial. Um outro fator que influenciou na escolha desta linguagem foi o fato da mesma possuir um conjunto de rotinas gráficas que podem ser utilizadas com bastante facilidade.

No que se refere ao volume de programas, o protótipo foi desenvolvido baseado nos seis módulos básicos, conforme mostrado na figura 3.2, que por sua vez subdividem-se em vários outros módulos perfazendo um total de aproximadamente 30 módulos. O programa executável tem o tamanho de 152 Kbytes tendo aproximadamente 5500 linhas de código. Para executar esse sistema, na versão atual, não é requerido uma grande quantidade de memória RAM, podendo dessa forma ser utilizado em microcomputadores convencionais.

O mecanismo de definição de equipamento e de interface de comunicação apresentado na seção 3.3 foi projetado para atender inicialmente aos equipamentos HPLC, Instron, Secador de Feijão e o teclado do computador e as interfaces de comunicação GPIB, RS232C e a placa conversora analógica digital AD5012. Os equipamentos HPLC, Instron e o Secador de Feijão foram escolhidos porque são bastantes utilizados nas pesquisas desenvolvidas no CTAA, e porque apresentam melhores condições para em um curto espaço de tempo serem interligados ao computador. Já o teclado foi inserido na lista de equipamentos porque oferece condições de simular experimentos laboratoriais de forma bastante simplificada.

Decidiu-se trabalhar inicialmente com as interfaces de comunicação citadas acima porque estas são as mais comuns nos equipamentos de laboratório adquiridos ou desenvolvidos pelo CTAA.

O módulo de gerenciamento do sistema resume-se em um único programa que ativa os outros módulos. Já o de edição de texto foi implementado por completo. No módulo de operações em arquivos somente a rotina de remoção de arquivos foi implementada.

O módulo de composição de programas, considerado o núcleo básico do sistema, é composto pelos seguintes sub-módulos: Coleta, Estatística, Gráfico, Exibe, Lê, Grava, Liga, Desliga, Calibra, Define, Apaga e Ajuda. A seguir é apresentada uma visão mais detalhada de cada um deles.

O sub-módulo Coleta é responsável pela tarefa de coletar dados de um determinado equipamento e armazená-los em memória para posterior uso. Existem várias formas de implementar esse sub-módulo. Essas formas estão ligadas ao tipo de experimento, equipamento, de interface de comunicação que está sendo utilizada, etc. Nessa versão do sistema, optou-se por um processo de coleta de dados que utiliza de três variáveis para controlar o seu funcionamento. São elas: Início, Fim e Intervalo. A variável Início tem como valor a hora em que o processo de coleta de dados deve ser disparado. Já a variável Fim tem como o valor a hora em que este processo deve ser encerrado, e a variável Intervalo armazena o espaço de tempo

entre uma coleta de dados e outra. Além das variáveis já comentadas, uma outra variável denominada hora_sistema é utilizada. Seu valor é extraído do relógio do computador. O algoritmo básico utilizado na implementação desse sub-módulo é mostrado a seguir.

```
Início
  Ler hora do sistema (computador)
  Enquanto (não for hora de disparar a coleta) faça
    Ler hora do sistema (computador)
  Fim de Enquanto
  Enquanto (não fim de coleta) faça
    Se (for hora de coleta) então
      coleta-se o dado
      atualiza hora da próxima coleta
      atualiza-se hora do sistema (computador)
    Fim
  Se Ler hora do sistema (computador)
  Fim de Enquanto
Fim
```

A decisão por essa implementação do sub-módulo de coleta de dados foi tomada em função de ainda não se dispor no momento de conhecimentos de engenharia elétrica e eletrônica que permitirão no futuro implementar outras variações para o processo de coleta de dados. Dessa forma, o SAL terá condições de atender a diversas situações referentes ao processo de coleta de dados.

O sub-módulo Estatística é composto por dez rotinas. São elas: Básicas, Média, Desvio Padrão, Variância, Mediana, Correlação, Regressão, Máximo, Mínimo. A rotina denominada Básicas engloba as rotinas de cálculo da média, desvio padrão, mediana e moda. Essa função foi criada porque diversas vezes em experimentos laboratoriais dados precisam passar primeiro por

uma avaliação para verificar se aquele conjunto de informações permitirá ao pesquisador obter conclusões satisfatórias do experimento. O que se observou nos laboratórios visitados é que os quatros tipos de cálculos mencionados são os mais utilizados por esse processo de investigação, daí então a necessidade de se contar com essa função no módulo de estatística. Além dessas funções, todas as outras encontram-se implementadas.

O sub-módulo Gráfico segue a mesma filosofia do sub-módulo anterior. Para compô-lo, cinco funções foram definidas. São elas: Gráfico de Barras em duas e três dimensões, Gráfico de Pontos, Gráfico de Linha e Gráfico de Setores(Torta). Das cinco funções encontram-se implementadas o Gráfico de Barras em duas dimensões e de Setores(Torta).

Os sub-módulos denominados Lê e Grava exercem funções de leitura e gravação de dados oriundos do processo de coleta de dados. Ambos encontram-se implementados. Para atuarem de forma precisa, eles requerem como parâmetro de entrada o nome do arquivo. Já o sub-módulo Exibe tem como função listar na tela do computador os dados coletados de um equipamento ou lidos de um arquivo. Esse também foi implementado.

Os sub-módulos Liga e Desliga foram projetados para ligar e desligar equipamentos. Para que sub-módulos possam realizar suas funções é necessário conhecer o nome do equipamento. Já o sub-módulo Calibra foi projetado com o objetivo de permitir ao usuário ajustar os parâmetros que definem as condições ideais de uso do equipamento em um

determinado tipo de experimento. O conjunto de parâmetros é passado ao equipamento através de uma linguagem de comandos existente no próprio equipamento, de maneira que tudo aconteça de forma totalmente transparente para o programador do experimento.

Já o sub-módulo Define exerce uma função vital para o processo de programação. É através dele que todo o sistema fica sabendo qual equipamento será utilizado pelo experimento, se haverá necessidade de definir uma nova interface para o equipamento e novos valores para os parâmetros de uma determinada interface. O mecanismo básico de funcionamento desse sub-módulo encontra-se descrito na seção 3.3 deste capítulo.

Os sub-módulos Liga, Desliga, Calibra e Define não foram implementados nessa versão do sistema, pois exigem, além de tempo, estudos e conhecimento de outras áreas, entre elas Engenharia Elétrica e Eletrônica. Estas implementações fazem parte da segunda parte deste trabalho.

O módulo de execução de programa encontra-se implementado conforme o algoritmo básico apresentado, na seção 3.3. Além da função de executar um programa construído pelo módulo de composição de programa, este também exerce a função de verificar se um programa está obedecendo às regras estabelecidas para boa formação de programa.

3.6 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os requisitos gerais do Sistema de Automação Laboratorial e de que forma procurou-se atendê-los. Foram apresentadas de forma mais detalhada também a arquitetura do protótipo, seus módulos básicos e suas funções. Finalmente, foram discutidas as limitações encontradas no protótipo e os motivos pelos quais elas ocorrem.

CAPÍTULO 4

DEMONSTRAÇÃO E AVALIAÇÃO DO PROTÓTIPO IMPLEMENTADO

4.1 CONSIDERAÇÕES INICIAIS

Este capítulo tem o objetivo de demonstrar o uso do protótipo do Sistema de Automação Laboratorial implementado incorporando-se as idéias e conceitos apresentados no capítulo 3.

No capítulo 3 foram apresentadas as especificações e os requisitos considerados importantes para essa versão do sistema. Neste capítulo são destacados os pontos principais da implementação do protótipo desenvolvido através de um exemplo de uso do sistema. Finalmente, uma avaliação preliminar da implementação realizada é apresentada.

4.2 UM EXEMPLO DE USO DO SISTEMA

Com os objetivos de apresentar o protótipo e validar o conjunto de idéias e conceitos apresentados na revisão da literatura, escolheu-se uma situação, simples mas comum quando se trata de secagem de grãos.

Deseja-se construir um programa que colete um conjunto de dados oriundos de um experimento de secagem de feijão e em seguida realize cálculos estatísticos básicos que são

necessários para analisar o comportamento inicial dos dados coletados. Em seguida, um gráfico de barras deve ser gerado. Os dados devem ser listados e posteriormente armazenados em um arquivo, sequencialmente.

Para melhor visualizar esse experimento a figura 4.1 mostra as partes envolvidas na coleta de dados relativos ao experimento de secagem de feijão.

Figura 4.1 - Uma Visão Global do Experimento de Secagem de Feijão.

De acordo com o domínio de aplicação definido para o SAL no capítulo 3, é mostrada na figura 4.2 a especificação do programa a ser construído para atender ao exemplo.

Figura 4.2 Especificação do Programa.

Embora vários tipos de investigações possam ser realizadas, neste caso deseja-se investigar a variação de umidade no interior (parte central) do secador durante um determinado período de tempo do dia. Portanto, o dado a ser coletado é a umidade da parte central do secador. Utiliza-se nesse exemplo o teclado para simular a entrada de dados, uma vez que esse é, na realidade, similar a qualquer outro equipamento com comunicação serial.

Uma placa conversora analógica digital AD5012 é utilizada para transformar um sinal analógico em um número. Para que essa placa atue de forma precisa é necessário, entre outras coisas, estabelecer valores para um conjunto de parâmetros vinculados a essa placa. Os parâmetros

são: número do cartão; tempo, ganho, modo, contador, canal inicial, canais e tamanho. O processo de coleta de dados inicia-se as 9:00 hs e termina as 9:03 hs. O intervalo entre as coletas é de 10 segundos.

O primeiro passo dado para a estruturação do programa é especificá-lo na forma de ações que venham ser utilizadas no processo de composição do programa. Em seguida, é definir os parâmetros e seus respectivos valores a serem utilizados em cada ação. Na tabela 4.1 encontram-se as ações, os parâmetros e seus respectivos valores, caso sejam necessários.

Com base na especificação do programa e dos parâmetros a serem utilizados, é mostrada a solução adotada para a construção do programa. À medida que a composição do programa vai sendo desenvolvida, são apresentadas algumas janelas do SAL, correspondendo a fase de especificação.

Ativação do Sistema - para ativar o sistema basta digitar a palavra SAL. Nesse momento o sistema é carregado e em seguida é exibida na tela do computador uma janela. Essa janela divide-se em três partes e corresponde à janela principal do sistema. Para ativar qualquer uma das opções dispostas na área de menu basta apontá-la com o mouse e apertar o botão da esquerda. A figura 4.3 apresenta a janela principal do sistema.

Ação	Parâmetro	Valor
1) Definição de Equipamento	Equipamento	Secador de Feijão (Teclado)
2) Definição de Configuração	Configura	Sim
3) Definição de Interface	Interface	AD5012
4) Definição dos Parâmetros da Interface	Cartão Tempo Ganho Modo Contador Canal Inicial Canais Tamanho	0 10 segs 1 0 1 1 1 2 1
5) Calibração do Equipamento	Início de Coleta Fim de Coleta Intervalo	9:00 hs 9:03 hs 10 segs
6) Coleta	—	—
7) Estatística	Tipo	Básicas
8) Gráfico	Tipo	Barra
9) Exibe Dados	—	—
10) Armazenamento de Dados	Arquivo	feijão93.dat
11) Fim	—	—

Tabela 4.1 - Ações, Parâmetros e seus Respectivos Valores para especificação do programa.

Figura 4.3 - Janela Principal do SAL

Registro da Especificação do Programa - para realizar essa tarefa ou mesmo registrar comentários diversos sobre o experimento, o SAL dispõe do EDSAL, representado pelo botão Editor, disposto na janela principal. Uma vez acionado, nova janela é exibida. Utilizando-se do processo de edição já apresentado o usuário pode então iniciar seu trabalho. As figuras 4.4 e 4.5 apresentam as janelas utilizadas nesse processo.

Figura 4.4 - Janela Básica do EDSAL

Este sistema foi desenvolvido para fornecer condições aos técnicos de laboratório para desenvolver programas para coletar dados de equipamentos de laboratório e em seguida analisá-los. O sistema não exige do usuário profundos conhecimentos de programação para utilizá-lo.....

Figura 4.5 - Janela de Edição do EDSAL

Composição de Programa - definida essa ação através do botão **Compor** do menu principal e de posse da tabela 4.1, inicia-se o processo de composição do programa para atender ao exemplo. A figura 4.6 apresenta a janela principal desse processo.

Figura 4.6 - Janela de Composição de Programas

A seguir segue a descrição do significado de cada parâmetro da interface de comunicação utilizada no exemplo. Na tabela 4.2 encontram-se os valores válidos para cada um dos parâmetros.

Cartão - número do cartão a ser inicializado.

Tempo - define o intervalo de tempo para que uma leitura seja aceita.

Ganho - configura o ganho do amplificador de entrada do

cartão.

Modo - Configura o modo de operação do cartão. Os modos são: normal, trigger externo e com interrupção.

Contador - número de leituras a serem armazenadas.

Canal - indica o canal do multiplexador que será lido pelo conversor Analógico/Digital.

Inicial - número do canal do multiplexador onde será feita a primeira leitura.

Canais - número de canais do multiplexador que serão usados.

Tamanho - tamanho do buffer para recepção dos dados.

Parâmetro	Validade
Cartão	0 a 3
Tempo	0 a 65535
Ganho	0 a 3
Modo	0 a 7
Contador	0 a 65535
Canal	0 a 15
Canal Inicial	0 a 15
Canais	1 a 16
Tamanho	0, 4 a 32767

Tabela 4.2 - Faixa de Validade dos Parâmetros da Interface AD5012.

Definição de Equipamento - esse processo é ativado através da seleção do botão Define. Em seguida um menu contendo os tipos de equipamentos que o sistema suporta é mostrado. No exemplo o equipamento escolhido é o Secador.

Definição de Configuração - esse processo exibe um menu para que o usuário informe se haverá ou não necessidade de configuração da interface de comunicação. No exemplo a resposta é Sim.

Definição do Tipo de Interface de Comunicação - segue a mesma filosofia empregada para definição de equipamento. Para esse exemplo a resposta é AD5012. A figura 4.7 mostra a janela e menus utilizados nesse processo de definição. A figura 4.8 mostra a janela para entrada de dados para os parâmetros da interface de comunicação definida.

Figura 4.7 - Menus de Definição.

Figura 4.8 - Janela de Entrada de Dados para os Parâmetros da Interface AD5012.

Calibração do Equipamento - nesse momento o usuário informa ao sistema os valores dos parâmetros que vão determinar as condições ideais para o funcionamento do experimento. Para esse caso o usuário determina os horários de início e fim do processo de coleta de dados. A figura 4.9 apresenta a janela para entrada de dados para o processo de calibração de equipamento.

Figura 4.9 - Painel do Secador.

Coleta - selecionada essa opção o sistema reconhece que um processo de coleta de dados deve ser disparado quando o relógio interno do computador atingir o horário especificado como o de início da coleta processo de calibração do equipamento.

Estatística - acionado o botão denominado **Estatist** que se encontra na janela de composição de programa, uma nova janela, figura 4.10, exibe os tipos de estatística que o sistema suporta. Nesse exemplo escolheu-se o tipo **Básicas**, que encontra-se descrito no capítulo 3.

Gráfico - Segue a mesma filosofia aplicada ao processo de definição do tipo de estatística. A figura 4.11 exibe os tipos de gráficos do sistema. No exemplo escolheu-se o tipo **Barra**.

Figura 4.10 - Janela de Tipos de Estatísticas.

Figura 4.11 - Janela de Tipos de Gráficos.

Exibe - selecionada essa opção o sistema reconhece que o processo de listagem dos dados que foram coletados deve ser

disparado em seguida ao processo de cálculos das estatísticas básicas.

Grava - nesse momento o usuário informa o nome do arquivo onde serão armazenados os dados coletados do experimento. Para esse exemplo o nome do arquivo é `feijao93`. A figura 4.12 mostra a janela utilizada para entrada da informação.

Figura 4.12 - Janela para Entrada do Nome do Arquivo.

Fim - ao ser selecionado, indica o fim do processo de composição do programa.

Execução do Programa - para executar o programa construído através do Módulo de Composição de Programa, aciona-se a opção **Executa**, no menu principal. Após o processo de calibração do equipamento o sistema mostra uma janela indicando que o processo de coleta entrou em execução. A janela utilizada no processo de coleta de dados é mostrada na figura 4.13.

Coletor de Dados

Início da Coleta	→	9:00
Término da Coleta	→	9:03
Intervalo entre Coleta	→	10

Entre com o Dado

■

Figura 4.13 - Janela do Coletor de Dados.

Terminada a etapa de coleta de dados, as respostas das funções escolhidas, estatística, gráfico e exibição de dados, começam a ser mostradas na tela do computador. Terminado o processamento das três fases acima mencionadas, é mostrada na tela do computador uma listagem da especificação do programa. Em seguida, é solicitado ao usuário que informe o nome do arquivo onde serão gravadas as informações utilizadas pelo programa que foi construído. Os resultados gerados pelo programa são mostrados nas figuras 4.14, 4.15, 4.16 e 4.17.

A Média e':	23.277779
O Desvio Padrão 'e:	1.850092
A Mediana e':	21.000000
A Moda e':	21.000000

Figura 4.14 - Estatísticas Básicas.

Figura 4.15 - Gráfico de Barras dos valores de umidade em relação ao tempo.


```

item 1: 20.000000
item 2: 22.000000
item 3: 21.000000
item 4: 23.000000
item 5: 20.000000
item 6: 24.000000
item 7: 23.000000
item 8: 22.000000
item 9: 25.000000
item 10: 23.000000
item 11: 24.000000
item 12: 25.000000
item 13: 26.000000
item 14: 27.000000
item 15: 28.000000
item 16: 26.000000
item 17: 25.000000
item 18: 24.000000

```

Figura 4.16 - Listagens dos Dados.

```

Define (secador sim)
Calibra (9:00 9:30 00:00:10)
Coleta (9:00 9:00 00:00:10)
Estatistica (basicas)
Grafico (barra)
Exibe
Grava (feijao.dat)
Fim

```

Figura 4.17 - Listagem da Especificação do Programa.

4.3 AVALIAÇÃO PRELIMINAR DO PROTÓTIPO IMPLEMENTADO

Nessa seção é apresentada uma avaliação preliminar do protótipo implementado ressaltando os aspectos principais e aqueles que ainda merecem ser melhor investigados e implementados.

O editor de texto na forma como foi implementado mostrou ser útil desde que seja utilizado como um quadro para pequenas anotações. Por não possuir recursos sofisticados, recursos esses existentes nos editores de textos comerciais, seu uso é limitado.

Conforme foi apresentado, o processo de composição de programas mostrou-se ser de fácil manipulação. Percebeu-se que a linguagem de programação implementada é limitada quando deseja-se criar de forma rápida novas funções. Por outro lado, constatou-se a linguagem utilizada no SAL e o mecanismo de composição de programas implementado reduzem o tempo de criação de programas e o número de erros, comparado com o processo tradicional de construção de programas.

O mecanismo de definição de equipamento e interface de comunicação apresentado, anexado a um conjunto de rotinas que possibilitem reconhecer a interface de comunicação a ser utilizada e como trocar informações entre o computador e o equipamento proporcionará ao SAL um certo grau de independência no aspecto de interfaceamento computador-equipamento de laboratório. Dessa forma, pretende-se chegar a um nível de

independência de forma que o interfaceamento computador-equipamento de laboratório não se constitua num problema quando for necessário utilizar o sistema para coletar dados de equipamentos que usam diferentes tipos de interface de comunicação. A falta de um mecanismo dessa natureza constitui-se num ponto falho dos sistemas para automação laboratorial apresentados no capítulo 2.

Em se tratando de um sistema voltado para técnicos de laboratório que normalmente têm pouca habilidade no uso computador, os estilos de interação homem-máquina utilizados são satisfatórios, pois reduzem o número de erros e evitam que o usuário tenha que memorizar comandos. Um outro ponto importante no SAL é a existência de um sub-sistema de ajuda. O sub-sistema de Ajuda implementado no SAL é útil por oferecer informações curtas mas que atendem de imediato as necessidades do usuário, reduzindo o uso de manuais.

O mecanismo de execução de programas também pode ser considerado satisfatório em virtude de requerer do usuário o mínimo de intervenção, por ser rápido e de fácil manuseio. Um aspecto interessante que esse sistema apresenta é o fato de fornecer para o usuário uma listagem em uma linguagem simples. Essa foi projetada para servir como parte da documentação do experimento desenvolvido.

Um outro aspecto importante mas que não foi implementado nessa versão é um mecanismo que permita ao usuário reexecutar um programa já construído de forma que ele somente

altere alguns poucos parâmetros, mantendo a estrutura básica do programa.

O SAL permite ao técnico de laboratório ter em suas mãos resultados mais precisos e mais rápido, permitindo-lhe tomar decisões com relação ao experimento de forma mais imediata. Um outro aspecto bastante importante é que hoje em dia existem equipamentos na EMBRAPA sendo sub-utilizados por falta de um sistema do tipo do SAL. Esse fator pode levar a EMBRAPA a substituir seus equipamentos sem que haja realmente necessidade. Embora ainda não esteja completo, o SAL pode contribuir para o aumento da produtividade no que diz respeito à construção de aplicações voltadas para o domínio especificado, aumentando a qualidade e a confiabilidade do código gerado e também reduzindo custos de treinamento e manutenção e incidência de erros.

Para que o SAL possa incorporar todas as características consideradas relevantes, é necessário reunir especialistas de várias áreas (Engenharia Elétrica, Eletrônica, de Software, etc) formando assim uma equipe multidisciplinar, o que não é simples.

Em laboratórios de pesquisa, de maneira geral, diversas tarefas (armazenagem, lavagem, secagem, etc) são realizadas antes de desenvolver o processo de investigação do produto. Devido a variedade de produtos, diferentes métodos de preparação e o fato dessas tarefas não serem rotineiras e executadas da mesma forma, dificultam o desenvolvimento de um

sistema totalmente integrado, ou seja, que atue desde da fase de preparação até a obtenção de resultados de experimentos os quais utilizaram o produto. Diante dessa questão, o SAL só é utilizado quando tem-se um conjunto de rotinas padronizadas e que são executadas de forma rotineira desde que se encaixe dentro do domínio especificado.

Acredita-se que o trabalho atingiu seu objetivo na medida de demonstrar a viabilidade de desenvolver um sistema baseado nos princípios discutidos por Masiero [MASIERO,93], Cleaveland [CLEAVELAND,88] e Martin [MARTIN,85] e que contribui para facilitar o trabalho dos técnicos de laboratório quando esses estão diante de uma situação onde é necessário ter em mãos de forma rápida um programa que seja capaz de coletar dados de um experimento e em seguida analisá-los.

4.4 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado um exemplo de como o SAL pode ajudar os técnicos de laboratório, principalmente os da EMBRAPA, a construir seus próprios programas para coletar e tratar dados oriundos de experimentos laboratoriais por eles realizados.

Inicialmente, foi apresentado de que forma os usuários devem se organizar para construir um programa de forma rápida e simplificada. Em seguida, mostrou-se como o sistema deve ser utilizado para desenvolvimento de um programa. Finalizando, uma

avaliação preliminar foi apresentada, onde procurou-se mostrar os pontos fortes e os pontos ainda considerados fracos do Sistema de Automação Laboratorial.

CAPÍTULO 5

CONCLUSÕES

5.1 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo principal lançar bases para o desenvolvimento de um sistema para automação laboratorial voltado para técnicos de laboratórios com pouco ou nenhum conhecimento de programação e uso de computador.

Para atingir esse objetivo foi necessário definir, levantar e estudar um conjunto de aspectos considerados básicos para o domínio de aplicação em questão, além de identificar sistemas voltados para o processo de automação laboratorial e suas características principais. Dessa forma conseguiu-se estabelecer uma base preliminar que permitiu construir o SAL.

Como foi visto, a forma de interação homem-máquina utilizada no SAL não se constitui em uma barreira para que pessoas não ambientadas com o uso do computador. A rapidez com que se desenvolve um programa utilizando o SAL possam utilizá-lo eficientemente, a constistência dos resultados gerados e a redução do número de erros, são fatores que possibilitam aumentar a produtividade e permitem que decisões sobre um determinado experimento sejam tomadas de forma rápida. Tudo

isso visa a obtenção de resultados de pesquisa de boa qualidade e a um custo reduzido.

5.1 SUGESTÕES PARA FUTURAS PESQUISAS

Para que esse sistema possa atender de forma integral aos objetivos esperados, julgou-se necessário desenvolver trabalhos de extensões e em paralelo iniciar novas pesquisas.

Como extensões ao sistema atual merecem ser citadas as seguintes: desenvolver rotinas que permitam a coleta de dados de equipamentos que possam ser interligados ao computador, independente do tipo de interface de comunicação que esteja sendo utilizado; definir uma estratégia que permita ao técnico de laboratório reaproveitar a estrutura básica do programa construído para que, com poucas alterações, ele possa reutilizá-lo em outro experimento. Em linhas gerais essa estratégia poderia ser implementada da seguinte forma: 1) recuperação do código gerado na linguagem BNF especificada; 2) permitir ao usuário alterar um conjunto de informações através do Módulo de Composição de Programa; submeter essa nova especificação a um parser e 4) ter um mecanismo que através dessa nova especificação reestabeleça o ambiente interno permitindo assim a execução do programa.

A integração de um gerenciador de janelas e a construção de um editor gráfico que possibilite ao usuário desenvolver um diagrama durante a fase de especificação do

programa, de forma que esse se transforme no programa a ser executado, seria uma extensão importante. Dessa forma ter-se-ia um sistema mais amigável, fazendo com que o técnico de laboratório se preocupasse apenas com os detalhes da especificação do programa.

A construção ou adaptação de um núcleo de um sistema de tempo real voltado para o ambiente DOS/PC também seria uma extensão interessante, pois permitiria o uso de programação concorrente, monitoramento e controle de mais de um experimento "ao mesmo tempo". Dessa forma, o sistema poderá ser utilizado em outras etapas do processo de automação laboratorial, como por exemplo: controlar uma caldeira.

Com relação a novas pesquisas dentro do ambiente laboratorial, pode-se pensar na identificação de novos domínios de aplicação. Para que o sistema se torne mais abrangente no seu domínio de aplicação, novos estudos são necessários para definir funções que possam ser utilizadas no controle de equipamentos ou de processos, de forma rotineira, padronizada e aplicável em diversos laboratórios, principalmente os da EMBRAPA. Um estudo desse tipo pode, na realidade, definir um novo domínio de aplicação, que seria o domínio Controle de Processo dentro de laboratórios com determinadas características a serem especificadas.

BIBLIOGRAFIA

- [ASYST,88] ASYST 2.0. Technical Software for Demanding Engineering and Scientific Applications, Rochester, Asyst Software Technologies, 1988. 11p.
- [BARKER,84] BARKER, P.G. Computer in Analytical Chemistry. Londres, Pergamon Press, 1984. (Pergamon Series in Analytical Chemistry, 6).
- [BARTH,86] BARTH P.S. An Object-Oriented Aproach to Graphical Interfaces. ACM Trans Graphics, V. 5, N. 2, p. 149 - 172, 1986.
- [BENTLEY,86] BENTLEY, J. Little Languages. Communications of ACM. N. 29, V.8, pg. 711 - 21, 1986.
- [BERTONUCCI,90] BERTONUCCI,V. Desenvolvimento de Controladores PID.São Carlos EESC-USP 1990, 130p. (Dissertação Mestrado).
- [CLEAVELAND,88] CLEAVELAND, J.C. Building Application Generators. IEEE Software, V. 5, N. 4, p. 25 - 33, 1988.
- [CLUNE,85] CLUNE T. R. Interfacing For Date Acquisition. Byte, V. 10, N. 2, p. 269 - 82, Feb. 1985.

- [DADISP,88] DADISP WORKSHEET. Cambridge, DSP Development Corp., 1988, 5p.
- [EDEL,88] EDEL M. The Tinkertoy Graphical Programming Environment. IEEE Transactions on Software Engineering, V. 14, N. 8, p. 1100 - 15, Aug. 1988.
- [ENGLEMANN,84] ENGLEMANN, B.; ABRAHAM, M. Personal Computer Processing. Byte, V. 9, N. 4, p. 94 - 109, Apr. 1984.
- [FONTANINI,88] FONTANINI,W.; FILHO O.S.S.; BARBARA,A.S. Um Editor de Diagrama de Blocos para o Projeto de Sistema de Controle, In: Congresso Brasileiro de Automática, 7. São Paulo, 1988. Anais. V.1. p. 179 - 85.
- [FORD,85] FORD, L.E. Laboratory Interfacing. Byte, V. 10, N. 2, p. 263 - 6, Feb. 1985.
- [GATES,84] GATES, S.C. Laboratory Data Collection with an IBM PC. Byte, V. 9, N. 5, p. 366 - 77, May 1984.
- [GRAFTON,85] GRAFFTON,R.B.; ICHIKAWA T.. Visual Programming. Computer, V. 18, N. 8, p. 6 - 9, Aug. 1985.

- [HIRAKAWA,87] HIRAKEWA, M. et al. HI-VISUAL Iconic Programming. In: 1987 Workshop on Visual Language. Linkping, 1987, p. 305 - 314.
- [LABTECH,87a] LABTECH NOTEBOOK AC1530/AC1529. In: Analog Devices (Noorwod, U.S.A) PC Bus Compatible Products 1987. Noorwod, MA, 1987, sec. 2, p.13 - 5.
- [LABTECH,87b] LABTECH Real Access AC1909. In: Analog Devices (Noorwod, U.S.A) PC Bus Compatible Products 1987. Noorwod, MA, 1987, sec. 2, p. 17 - 8.
- [LABVIEW,86] LABVIEW: Laboratory Virtual Instrument Engineering Workbench. Byte, V. 11, N. 9, p. 84 - 92, 1986.
- [LINTON,89] LINTON,M.A.; VLISSIDES J.M.; CALDEN P.R. Composing User Interfaces with Interviews. IEEE Computer, V. 22, N. 2, p.8 - 22, 1989.
- [MARTIN,85] MARTIN, J.M. Fourth Generation Languages. New Jersey, Prentice-Hall, 1985.
- [MASIERO,93] MASIERO, P.C.; MEIRA, C. A. A. Development and Instantation of Generic Application Generation. J. Systems Software, 1993. (to appear).

- [MEIRA,91] MEIRA, C.A.A. Sobre Geradores de Aplicação. São Carlos, ICMSC-USP 1991, 92p. (Dissertação de Mestrado).
- [NETTO,88] NETTO, A.T. Projeto e Construção de um Digitalizador e Promediador de Dois Canais para Tomografia por Resonância Magnética Nuclear. São Carlos, IFQSC-USP 1988. 135p. (Dissertação Mestrado).
- [NEWMAN,81] NEWMAN W.M.; SPROUL R.F. Principles of Interactive Computer Graphics. McGraw-Hill, 1981.
- [PARK,88] PARK, K. et al. Desenvolvimento de Produtos e Processos. Campinas, FEA/UNICAMP, 1988, 56p, (R.T.02).
- [PERKUSICH,88] PERKUSICH A., MPRAIS M.E., PINTO F.V. Interface Gráfica Homem Máquina para Aplicação em Automação e Controle de Processos. In: Congresso Brasileiro de Automática, 7. São Paulo, 1988, Anais. V.1. p. 186 - 91.
- [PRESMAN,87] PRESSMAN,R.S. Software Engineering: A Practitioner's Approach. 2 ed., Singapore, McGraw-Hill, 1984.

- [RAEDER,85] READER G. A Survey of Current Graphical Programming Techniques. IEEE Computer, V. 18, N. 8, p. 11 - 25, Aug. 1985.
- [SANTORI,90] SANTORI,M. An Instrument that isn't really. IEEE Spectrum, V. 27, N. 8, p. 36 - 39. Aug. 1990.
- [SANTOS,89] SANTOS,A.C. Tutorial: Uma Visão sobre Fatores Humanos em Projeto de Sistema de Computação, Universidade Federal de São Carlos - Departamento de Computação, p. 27.(Comunicação Pessoal).
- [SCHOENFELD,77] SCHOENFELD, R.L. et al. The Microprocessor in the Biological Laboratory. Computer, V. 10, N. 5, p. 56 - 67, May 1977.
- [SHU,88] SHU N. Visual Programming. New York, Van Nostrand 1988.
- [SIBERT,86] SIBERT J.L.; Hurley W.D.; Bleser T.W. An Object Oriented User-Interface Management System. Computer Graphic, V. 20, N. 4, p. 256 - 68, 1986.
- [SINGH,90] SINGH G., VU: Visual User-Interface Design. The Visual Computer, V. 6, N. 4, p. 230 - 44, Aug. 1990.

- [STD] STD - Sistemas Técnicos Digitais S.A. Manual do Usuário STD-5012: Conversor A/D rápido, Brasília, STD, 1989.
- [TAKAI,87] TAKAI, O.K. Desenvolvimento de Programas Computacionais para Controle de Experimentos em Eletroquímica. São Carlos, Depto. Química, UFSCar, 1987. 37p. (Relatório no. 1.)
- [TAKAI,88a] TAKAI, O.K. Desenvolvimento de Programas Computacionais para Controle de Experimentos em Eletroquímica. São Carlos, Depto. Química, UFSCar, 1988. 48p. (Relatório no. 2).
- [TAKAI,88b] TAKAI, O.K. Desenvolvimento de Programas Computacionais para Controle de Experimentos em Eletroquímica, São Carlos, Depto. Química, UFSCar, 1988. 44p. Relatório no. 3).
- [TELLO,84] TELLO, E. PolyForth and PC/Forth. Byte, V.9, N. 12, p. 303 - 13, Nov. 1984.
- [THOMAS,89] THOMAS, D. What's in an Object ? Byte, V. 14, N. 4, p. 231 - 40, Mar. 1989.
- [TOZZI,83] TOZZI,C.L.; NETO M.L.A; MELNIKOFF S.S.S. Introdução a Sistemas de Tempo Real para Controle de Processo. In: Congresso Nacional de Automação Industrial - CONAI,São

Paulo, 1983.

- [WIRTH,86] WIRTH, P.; FORD, L.E. Five Laboratory Interfacing Packages, Byte, V. 11, N. 7, p.303 - 12, Jul. 1986.
- [WRIGHT,87] WRIGHT, V. Desktop Data Acquisition. PC Tech Journal, V. 5, N. 2, p. 106 - 122, Feb. 1987.

APÊNDICE 1

GRAMÁTICA DA LINGUAGEM PARA AUTOMAÇÃO LABORATORIAL

<programa > ::= <função_programa> fim

<função_programa> ::= <função> | <função> <função_programa>

<função> ::= Coleta (<lista_números>)

| Gráfico (<tipo_1>)

| Estatística (<tipo_2>)

| Le (<nome>)

| Grava (<nome>)

| Exibe

| Liga (<nome>)

| Desliga (<nome>)

| Calibra (<lista_números>)

| Define (<nome>, <configura>)

<tipo_1> ::= barra

| barra_3d

| pontos

| linhas

| torta

<tipo_2> ::= básicas

| média
| desvio_padrão
| moda
| variância
| mediana
| correlação
| regressão
| máximo
| mínimo

<lista_números > ::= <número> | <número> <lista_números>

<número> ::= <digito> | <número> <digito>

<digito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<nome> ::= <letra> | <nome>

<letra> ::= a | b | ... | z | A | B | ... | Z

<configura> :: = sim | não