

Universidade Federal de Santa Catarina
Centro Tecnológico
Pós-graduação em Metrologia Científica e Industrial

**Emprego da orientação a objetos para
caracterização de recursos em um
ambiente de simulação de
instrumentos**

Dissertação submetida à Universidade Federal de Santa Catarina para
obtenção do grau de Mestre em Metrologia

Carlos Aurélio Pezzotta

Florianópolis, 8 de novembro de 2001

Emprego da orientação a objetos para caracterização de recursos em um ambiente de simulação de instrumentos

Carlos Aurélio Pezzotta

Esta dissertação foi julgada adequada para obtenção do título de
Mestre em Metrologia
e aprovada na sua forma final pelo
Programa de Pós-graduação em Metrologia Científica e Industrial



Prof. Carlos Alberto Flesch, Dr. Eng.
Orientador

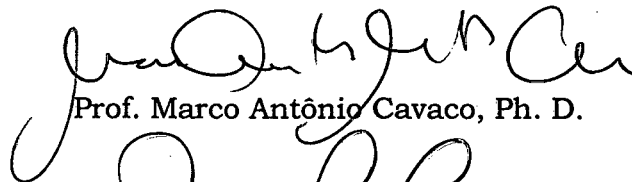


Prof. Armando Albertazzi Gonçalves Jr., Dr. Eng.
Coordenador do Curso de Mestrado em Metrologia Científica e Industrial

Banca Examinadora:



Prof. Armando Albertazzi Gonçalves Jr., Dr. Eng.



Prof. Marco Antônio Cavaco, Ph. D.



Prof. Werner Kraus Jr., Ph. D.

Aos meus pais
Carlos Roberto e Nícia

À minha irmã Valéria Cristina
e minha sobrinha Raquel

À minha namorada Analucia

Agradecimentos

Aos meus pais pela paciência e pelo amor dedicado durante toda a minha vida acadêmica.

Ao LABMETRO, pelas instalações, pelos professores e pela oportunidade da realização deste mestrado.

Ao governo brasileiro que, através da CAPES e do Programa de Integração Graduação/Pós-graduação - PROIN, garantiu o suporte financeiro necessário para a concretização deste trabalho.

Ao meu orientador e amigo, Prof. Carlos Alberto Flesch, pelas suas sábias palavras e críticas, sempre construtivas e incentivadoras.

Aos meus amigos, mestres e futuros mestres do LABMETRO, que participaram dos momentos de alegria, correria e tristeza, fornecendo cada um seu particular apoio: André Dias de Oliveira, Antonio Carlos Xavier, Daniel Willemann, Gláucio Andrey Maas, José Ricardo Menezes e Sílvia Regina Darrigo.

À Analucia Vieira Fantin, pela força, dedicação, paciência e amor em especial.

À Rosana Magali Vieira, pelo trabalho sempre prestativo e certo nos assuntos diversos do laboratório.

Aos professores, que me despertaram o gosto pela pesquisa, mas que sobretudo me formaram cidadão.

Ao bolsista Augusto De Nardin, pela colaboração e presteza no cumprir de seu trabalho.

Sumário

Sumário	iv
Lista de Figuras	vii
Resumo	viii
Abstract	ix

Capítulo 1

Introdução	10
1.1 Justificativa	10
1.2 Proposta do trabalho	11
1.3 Estrutura da dissertação	12

Capítulo 2

Simulador de Instrumentos	14
2.1 Uma visão geral sobre a tecnologia de simuladores.....	15
2.1.1 Quando utilizar simuladores	16
2.1.2 Uso de simuladores no ensino	17
2.2 Os laboratórios virtuais	18
2.2.1 EE-Lab – Portland State University	18
2.2.2 Cyberlab – Stanford University	19
2.3 Localização do trabalho de mestrado	20
2.3.1 Simulador de processos	21

2.3.2 Sistema de aquisição de dados de processos reais.....	21
2.3.3 Servidor de dados de processos.....	22
2.4 O simulador de instrumentos.....	22
2.4.1 Definição dos requisitos do ambiente de simulação e caracterização das linguagens de desenvolvimento.....	23
2.4.2 Seleção das linguagens de programação do simulador.....	25
2.4.3 Desenvolvimento de recursos de simulação em Labview e Visual Basic.....	26
2.4.4 Abordagem metrológica do ambiente de simulação.....	28
2.4.5 Estrutura básica do ambiente de simulação de instrumentos.....	30

Capítulo 3

Análise do Problema na Visão da Orientação a Objetos	31
3.1 Reconhecimento do problema de software	32
3.2 Programação orientada a objetos com foco no ambiente de simulação ...	34
3.3 Proposta de implementação da solução do problema de software.....	38
3.3.1 Interface.....	39
3.3.2 Ferramentas de simulação	40
3.3.3 Instrumentos	41
3.3.4 Recursos de análise	45

Capítulo 4

Implementação do Ambiente de Simulação	46
4.1 A interface do ambiente de simulação	47
4.2 O núcleo operacional do sistema	49
4.2.1 Classe clsElemento	51
4.2.2 Classe ctlInstrumento – Controle	53
4.2.3 Classe ctlLinhaReta – Controle.....	54
4.2.4 Classe clsFerramenta.....	55
4.2.5 Classe clsInstrumento.....	57
4.2.6 Classe clsTerminal	58
4.2.7 Classe clsLigação	60
4.2.8 Classe frmSimul.....	62
4.3 Dificuldades enfrentadas na implementação do núcleo operacional.....	64

4.3.1 A linguagem Visual Basic e a orientação a objetos.....	64
4.3.2 Objetos com relacionamento bidirecional	65
4.3.3 Interfaceamento e representação gráfica.....	66
4.4 Análise e avaliação das soluções implementadas	67

Capítulo 5

Conclusões e Sugestões para Trabalhos Futuros	69
5.1 Conclusões.....	69
5.2 Melhorias e desenvolvimentos a serem implementados para o ambiente de simulação	72
Referências Bibliográficas	74

Lista de Figuras

Figura 2.1. Diagrama geral da Estação Laboratorial	21
Figura 2.2. Concepção física do Sistema de Simulação de Instrumentos.....	28
Figura 2.3. Cadeia de medição genérica.....	29
Figura 3.1. Exemplo de classe e objeto	36
Figura 3.2. Padrão para parametrização de modelos de simulação	43
Figura 3.3. Modelo para simulação de um termistor	44
Figura 4.1. O ambiente de simulação	47
Figura 4.2. A edição no ambiente de simulação	48
Figura 4.3. Inter-relacionamento entre classes do sistema.....	51
Figura 4.4. As classes clsElemento, ctlLinhaReta, ctlInstrumento e eleJanela .	52
Figura 4.5. As classes clsFerramenta, ferSeleção, ferFio e ferInstrumento	56
Figura 4.6. As classes clsInstrumento, insVoltmetro e insTermopar	57
Figura 4.7. A classe clsTerminal.....	59
Figura 4.8. Classe clsLigação	61
Figura 4.9. Classe frmSimul.....	62

Resumo

Com a redução dos recursos disponíveis para a manutenção e atualização dos laboratórios, a Universidade Federal de Santa Catarina, mais especificamente o Laboratório de Metrologia e Automatização – LABMETRO, vem desenvolvendo um projeto intitulado “Estação Laboratorial Multidisciplinar para Suporte ao Aprendizado Teórico e Prático”.

Este trabalho é parte integrante desse projeto, tendo como alvo o módulo denominado ‘Simulador de instrumentos’. Ele objetiva a estruturação de um ambiente de simulação utilizando a linguagem Visual Basic e a idealização e implementação de classes de objetos pertinentes à interface gráfica desse simulador.

O trabalho inicialmente trata dos principais conceitos necessários ao entendimento do escopo onde ele se insere, detalhando com maior rigor o módulo de simulação de instrumentos. Para esse módulo são enunciados e detalhados todos os recursos necessários para o atendimento dos requisitos do software de simulação, passando então para uma discussão sobre a Programação Orientada a Objetos aplicada ao problema em questão.

O trabalho traz também a implementação de classes pertinentes à interface gráfica do sistema, mostrando o detalhamento técnico, as principais dificuldades encontradas nesse processo e uma análise e avaliação das soluções técnicas assumidas no seu desenvolvimento.

Abstract

Due to the reduction of available resources to maintain and update laboratory environments, the Automation and Metrology Laboratory of the Federal University of Santa Catarina has been developing a project called "Multidisciplinary Laboratorial Workbench to Support Practical and Theoretical Learning".

This research is part of the project, more specifically the module called "Instrument Simulator". Its scope is the definition of the structure of a simulation environment using the Visual Basic programming language. It is also the idealization and implementation of object classes pertaining to the graphic interface of the simulator.

The study introduces the main concepts needed to understand the entire project. It details the Instrument Simulation module explaining the resources needed to attain the simulation system software and the Object Oriented approach.

The study also comments on the object classes implementation related to the graphical interface of the system. It shows technical details, main difficulties found during their completion and an evaluation of the technical solutions assumed during the development of the system.

Capítulo 1

Introdução

1.1 Justificativa

A atividade laboratorial é responsável por importante parte do aprendizado da engenharia e outras ciências [1].

Com a redução dos recursos disponíveis para atualização tecnológica desse ambiente de aprendizado, alternativas diferenciadas e menos onerosas são necessárias para manter os laboratórios ativos e contribuindo concretamente para a formação acadêmica e técnica de novos profissionais [2].

Nesse sentido, a Universidade Federal de Santa Catarina, na figura do Laboratório de Metrologia e Automatização – LABMETRO, vem desenvolvendo um projeto intitulado “Estação Laboratorial Multidisciplinar para Suporte ao Aprendizado Teórico e Prático”.

Esse projeto tem seu foco no desenvolvimento de um sistema baseado em instrumentação virtual. Essa abordagem, que tem no software sua maior concentração de importância, apresenta custo reduzido se comparada com laboratórios convencionais [3].

A Estação Laboratorial vem como um elemento intermediário entre a sala de aula e o laboratório, contribuindo positivamente para o ensino e o aprendizado no que diz respeito à instrumentação e automatização de sistemas de medição.

Este trabalho faz parte do conjunto de atividades sendo executado no LABMETRO em prol do desenvolvimento dessa Estação Laboratorial. Ele tem seu enfoque localizado no módulo intitulado “Simulador de Instrumentos”, mais especificamente no desenvolvimento da interface desse simulador. Ele contempla aspectos teóricos e estruturais de sua criação, passando por detalhes de implementação que tem sua origem no planejamento do software até sua concepção final.

1.2 Proposta do trabalho

Este trabalho faz parte do desenvolvimento de um ambiente de simulação capaz de permitir a montagem de cadeias de medição seguindo uma estrutura composta pelos seguintes módulos funcionais:

- grandeza a medir;
- transdução;
- condicionamento de sinais;
- indicação.

Nesse sentido, o trabalho tem seu foco direcionado para duas frentes:

- o projeto de um ambiente de simulação utilizando a linguagem Visual Basic, estabelecendo requisitos e definindo sua estruturação;
- a idealização e implementação de classes de objetos pertinentes à interface gráfica do simulador, com vistas à possibilidade da montagem de experimentos sem a necessidade de programação, bem como a expansão futura através da adição de novos módulos de instrumentos.

A partir do desdobramento do objetivo geral, definiram-se os seguintes objetivos específicos:

- estabelecimento dos requisitos técnicos do ambiente de simulação;
- caracterização e seleção da linguagem de programação a ser utilizada para a implementação do sistema;
- estabelecimento de grupos de recursos necessários para a implementação do ambiente de simulação;

- definição e implementação de uma estrutura de classes de objetos necessários para o desenvolvimento do sistema.

Para a organização do trabalho, empregou-se um software de apoio denominado Rational Rose [61]. Essa ferramenta implementa um sistema de auxílio à especificação orientada a objetos baseada em UML (Unified Modeling Language) [31]. No entanto, devido à cultura local e à necessidade urgente de resultados, optou-se por não se aprofundar no entendimento e uso de técnicas padronizadas de especificação.

1.3 Estrutura da dissertação

No Capítulo 2, são apresentados os principais conceitos necessários ao entendimento deste trabalho, bem como no que consiste o sistema de simulação de instrumentos e o contexto no qual ele está inserido. Para tal, inicialmente é apresentada uma visão geral sobre simuladores e sua aplicação ao ensino. Em seguida, é estabelecido o posicionamento do trabalho frente ao projeto “Estação Laboratorial Multidisciplinar para Suporte ao Aprendizado Teórico e Prático”. Finalmente, o módulo de simulação de instrumentos é detalhado através de uma minuciosa análise que vai desde a definição de requisitos do sistema, até a estruturação do ambiente de simulação.

No Capítulo 3, são enunciados e detalhados todos os recursos necessários para o atendimento dos requisitos de software do sistema de simulação. Dessa forma, primeiramente é caracterizado o problema a ser resolvido, passando-se para uma discussão sobre Programação Orientada a Objetos aplicada à simulação. Por fim, é apresentada uma visão detalhada no que diz respeito aos grupos de recursos definidos e considerados principais no desenvolvimento do ambiente de simulação.

O Capítulo 4 traz os detalhes envolvidos no processo de implementação da interface do sistema de simulação. Assim, é mostrada a interface gráfica do sistema e o detalhamento técnico da implementação das classes desenvolvidas. Em seguida são apresentadas as principais dificuldades enfrentadas no processo de desenvolvimento do ambiente de simulação. Finalmente é realizada a análise e a avaliação das soluções assumidas no desenvolvimento do sistema.

Por fim, no Capítulo 5 são apresentadas as conclusões do trabalho de mestrado. São sintetizados os principais resultados obtidos e ressaltados tópicos relevantes presentes durante todo o período de desenvolvimento da dissertação. Adicionalmente são apresentadas sugestões para a realização de futuros trabalhos que aprimorem e dêem continuidade à implementação do ambiente de simulação de instrumentos.

Capítulo 2

Simulador de Instrumentos

Este capítulo tem como propósito apresentar os principais conceitos utilizados neste trabalho, assim como demonstrar, teoricamente, no que consiste o sistema de simulação de instrumentos e o contexto no qual ele está inserido.

Primeiramente é apresentada uma visão geral sobre simuladores e sua aplicação ao ensino, contemplando as características benéficas dessa combinação e posicionando essa ferramenta frente às técnicas convencionais de ensino.

Após, são apresentados dois casos de laboratórios virtuais já implementados por universidades renomadas, de forma a elucidar e exemplificar sua utilização como ferramenta auxiliar ao ensino tradicional.

Em seguida é apresentado o contexto no qual este trabalho está inserido, com uma breve explicação sobre cada módulo constitutivo do projeto “Estação Laboratorial Multidisciplinar para Suporte ao Aprendizado Teórico e Prático”.

Finalmente, é detalhado o módulo de simulação de instrumentos, passando por uma minuciosa análise que vai desde a definição de requisitos do sistema, até a estruturação do ambiente de simulação.

2.1 Uma visão geral sobre a tecnologia de simuladores

O termo simulação é definido de forma genérica como uma técnica através da qual se procura imitar o comportamento de alguma situação ou sistema [4]. É uma modalidade experimental de pesquisa que visa a retirada de conclusões através de modelos que buscam representar a realidade, seja de sistemas já implementados ou de projetos sob investigação [5][6].

A palavra “simulação” traz à mente de muitas pessoas sofisticadas idéias relacionadas com simuladores de voo ou realidade virtual. Entretanto, sua realização envolvendo a colocação de pessoas em ambientes realistas e complexos, é apenas um dos meios de se fazê-la. Pesquisas indicam que mesmo simulações com baixo nível de realismo, produzem efeitos significativos para treinamentos e avaliações [8]. Isso possibilita que plataformas de baixo custo como computadores pessoais, aparelhos dedicados, videogames, entre outros, possam ser utilizados para esse propósito.

Os modelos utilizados pelas simulações podem conservar ou não as características físicas e lógicas do sistema real imitado [5]. Quando essas características são mantidas, há apenas o processo de miniaturização ou representação parcial de um sistema real. O caso em que estamos particularmente interessados ocorre quando o modelo não conserva as características físicas do sistema real. Assim temos a chamada simulação simbólica, onde a parte lógica do sistema real, que é conservada, é expressa através de várias equações matemáticas, com suas variáveis representando os componentes do sistema [5][9].

A simulação simbólica pode ser classificada em dois tipos:

- simulação de problemas determinísticos;
- simulação de problemas estocásticos ou probabilísticos.

A simulação de problemas determinísticos refere-se à resolução de questões que envolvam equações diferenciais, integrais, matrizes, entre outros, através de processos experimentais em computadores.

A simulação de problemas estocásticos ou probabilísticos abrange os casos nos quais a própria natureza estocástica do problema impede sua

solução através de métodos matemáticos convencionais, sendo a simulação o melhor, ou muitas vezes o único método de resolução.

2.1.1 Quando utilizar simuladores

Os simuladores vêm sendo largamente utilizados tanto no âmbito industrial quanto no educacional. De uma forma mais precisa, a razão para a utilização de simuladores advém, principalmente, da impossibilidade de realização de experimentos com sistemas reais devido a fatores como custo e segurança [7][10].

A simulação é utilizada com freqüência em sistemas que apresentam características como [9]:

- modelos muito complexos com muitas variáveis e interações entre componentes;
- funções que apresentam comportamento não-linear;
- modelos que contêm variáveis aleatórias.

Além desses motivos, existem outros méritos atribuídos à ferramenta de simulação que a torna adequada em circunstâncias específicas. São eles[8]:

- a habilidade que a simulação tem de permitir a experimentação de situações realistas sem enfrentar riscos financeiros ou de segurança;
- o escalonamento do tempo real em tempo simulado, significando a condensação (ou expansão) de uma variedade de situações, que tipicamente ocorreriam em um longo (ou curto) intervalo, para um período de tempo menor (ou maior) de simulação.

Seguindo essa linha, podem ser inferidos casos gerais nos quais a simulação pode atuar como ferramenta auxiliar de grande importância [5][7][8][9][10][11]:

- para experimentação e avaliação, isto é, na tentativa de prever as conseqüências de mudanças de condições ou métodos sem ter de fazê-las no sistema real com grandes gastos e com riscos de obtenção de resultados inesperados;
- como maneira de estudar novos sistemas a fim de reprojotá-los ou refiná-los;

- como ferramenta para ensinar e treinar equipes com equipamentos ou sistemas ainda não operacionais;
- para a verificação ou demonstração de uma nova idéia, sistema ou maneira de resolução de um problema;
- como meio de projeção no futuro, isto é, como ferramenta de previsão e planejamento quantitativo.

2.1.2 Uso de simuladores no ensino

O computador é instrumento fundamental tanto como meio de aprendizado quanto para o treinamento de funções a serem exercidas na vida profissional.

Na engenharia, um dos mais tradicionais e freqüentes usos do computador se dá como ferramenta de simulação [12].

A adição da simulação computacional às aulas de laboratório convencional vem colaborar no processo ensino/aprendizagem através da incorporação de diversas vantagens. Entre elas está a possibilidade de utilização de simulação em sala de aula, e o reduzido custo de manutenção e atualização de software e hardware computacional em relação aos equipamentos de laboratório.

Ainda nesse sentido, a simulação habilita os estudantes a examinar, de maneira interativa, teorias e aplicações de diversas áreas, pois possibilita um retorno instantâneo e confiável do experimento que estiver sendo realizado [13][14]. A simulação também oferece a oportunidade de testar diferentes opções e avaliar seus resultados rapidamente, incentivando o aprendizado e o teste de hipóteses. A interatividade possibilita que novas situações e configurações possam ser criadas, enriquecendo assim as conclusões e o entendimento do conteúdo alvo do estudo.

Em uma pesquisa realizada na Universidade de Illinois [13], foi investigada a eficiência da utilização de simuladores como auxiliares no processo de ensino e aprendizagem. Na pesquisa realizada, foi verificado que o índice de satisfação de alunos participantes de aulas de laboratório auxiliadas por simulador, era maior se comparado com alunos que realizavam aulas de laboratório convencionais. Isso devido à possibilidade de se poder executar,

com sucesso e em um curto intervalo de tempo, mais experimentos simulados que convencionais, estimulando a pesquisa e a curiosidade pelo saber.

Apesar disso, é de suma importância ressaltar que qualquer metodologia resultante da utilização de ferramentas de simulação não substitui instrutores ou aulas expositivas, mas sim potencializa os resultados em termos de aprendizagem. Eles vêm como métodos adicionais aos tradicionais, passando a fazer parte do rol de recursos disponíveis para os profissionais da área da educação.

2.2 Os laboratórios virtuais

Como uma alternativa ao ensino laboratorial convencional, a utilização de laboratórios virtuais está sendo vista como uma área estratégica de pesquisa. Essa abordagem pioneira está sendo adotada principalmente por instituições de ensino da área tecnológica, preocupadas com a qualidade e a manutenção da atualização de seus laboratórios.

Nesse sentido, de forma a elucidar e exemplificar a utilização de laboratórios virtuais como ferramenta auxiliar ao ensino, foram selecionados dois casos já implementados em universidades conceituadas mundialmente. Cabe destacar que a abordagem e a tecnologia escolhida para o funcionamento de cada laboratório virtual muda de caso para caso, demonstrando a variabilidade de situações na qual os mesmos podem ser aplicados como meios alternativos de ensino.

2.2.1 EE-Lab – Portland State University

O laboratório virtual desenvolvido em Portland consiste em um sistema composto por instrumentos baseados em padrões industriais como VISA, GP-IB e VXI, ferramentas computacionais acessórias como Matlab e PSpice e uma forte carga em software [1].

O laboratório, em conjunto com seus experimentos, é voltado para aplicações da engenharia elétrica, como medições e simulação de circuitos. Ele é totalmente baseado em instrumentos virtuais, tendo sua funcionalidade assegurada tanto com instrumentação física ou simulada, local ou remota.

O sistema foi concebido de maneira a não exigir programação por parte do realizador do experimento. Para tal, foi desenvolvido uma interface gráfica chamada “Experimenter Toolkit” onde é possível especificar graficamente os ensaios, contemplando os parâmetros dos sinais e os tipos de medição a serem realizadas [15].

2.2.2 Cyberlab – Stanford University

O “Cyberlab” é um ambiente laboratorial virtual para acesso remoto às dependências dos laboratórios do “Stanford Instructional Television Network”.

Ele foi criado com a intenção de satisfazer a necessidade que os estudantes têm de ficar expostos aos processos e aos principais conceitos discutidos em sala de aula [16].

Um projeto piloto baseado em uma aplicação da óptica teve seu início de operação em 1998, com o intuito de possibilitar a experimentação de conceitos básicos dessa teoria através da INTERNET.

Apesar da escolha de um experimento ligado especificamente à área de física, o conceito do “Cyberlab” permite sua utilização em diversas áreas de conhecimento incluindo engenharia, biologia, química, medicina, entre outros.

O ambiente “Cyberlab” disponibiliza bibliografias de referência, ferramentas de análise computacional, um sistema para agendamento de experimentos, além de um memorial de laboratório individualizado para cada usuário do sistema. Esse último é considerado como fator chave para uma utilização mais proveitosa do mesmo. Ele disponibiliza funções clássicas de armazenamento de informações e atua como um repositório central de dados que descreve o progresso do estudante no laboratório, mostrando o resultado dos testes, os dados obtidos durante o processo de experimentação, a comunicação com os professores, bem como um completo histórico de todos os passos tomados pelo estudante para a obtenção dos resultados. Dessa forma, os professores podem lançar mão da utilização desse memorial para avaliar de forma mais eficiente seus estudantes [17].

O “Cyberlab” foi desenvolvido utilizando um computador padrão PC de 300 MHz, uma placa de comunicação GPIB para controlar e monitorar um diodo laser com comprimento de onda de 635 nm, bem como uma placa de

aquisição de imagens para digitalizar sinais de vídeo provenientes de uma câmera padrão.

2.3 Localização do trabalho de mestrado

Com o avanço tecnológico em ritmo acelerado, manter os laboratórios com estrutura suficiente para garantir o ensino com tecnologia de ponta se tornou uma tarefa muito dispendiosa. A popularização e a redução do custo dos computadores fizeram com que soluções intermediárias e menos onerosas fossem vislumbradas, transformando o modo tradicional como os laboratórios eram vistos. Essa transformação é conferida como uma mudança do pólo de importância do instrumental laboratorial, passando do hardware para o software, indicando a atual tendência da utilização de instrumentos simulados ou controlados por software e a redução da utilização de equipamentos caros e de difícil atualização.

Seguindo essa linha, encontra-se em desenvolvimento no Laboratório de Metrologia e Automatização da Universidade Federal de Santa Catarina – LABMETRO, um projeto denominado “Estação Laboratorial Multidisciplinar para Suporte ao Aprendizado Teórico e Prático”. Esse trabalho se configura como uma proposta inovadora de desenvolvimento de um sistema que servirá de elemento intermediário entre a sala de aula e o laboratório, permitindo o aprendizado local e remoto de técnicas e práticas laboratoriais dentro da linha de instrumentação e automatização de sistemas de medição.

A Estação Laboratorial Multidisciplinar é uma estrutura baseada em instrumentação virtual. Essa abordagem faz com que ela tenha seu custo reduzido pelo fato de que, diferentemente dos usuais laboratórios de ensino, ela tem no software sua maior concentração de importância, sendo o hardware o meio de suporte [2].

A Figura 2.1 apresenta a concepção da Estação Laboratorial, dividindo-a em quatro blocos principais.

O enfoque deste trabalho encontra-se no módulo intitulado “Simulador de instrumentos”, o qual é detalhado no item 2.4. Para os outros módulos, são dadas informações suficientes para o entendimento macro do sistema, de forma

a posicionar a contribuição deste trabalho no contexto da “Estação Laboratorial”.

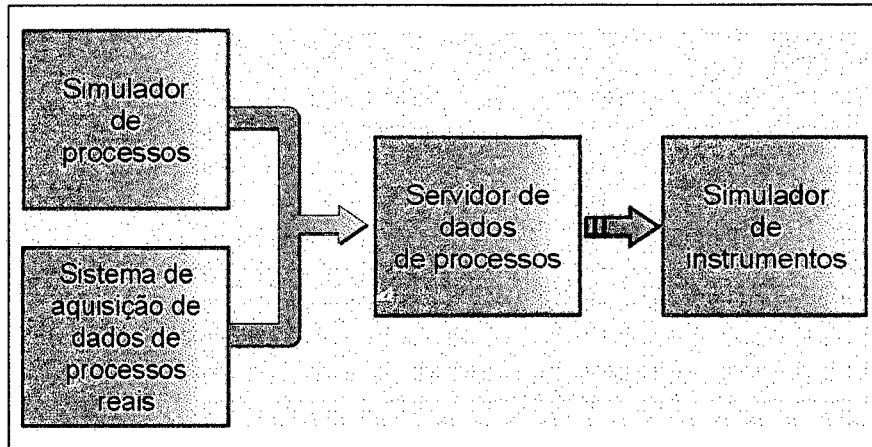


Figura 2.1. Diagrama geral da Estação Laboratorial

2.3.1 Simulador de processos

O “Simulador de processos” consiste em um módulo de software que contém recursos necessários que objetivam a simulação de processos físicos reais. Ele apresenta como saída um conjunto de grandezas representativas de um processo físico. Essas grandezas são dinamicamente geradas de forma coerente e inter-relacionada, representando o comportamento real de sistemas físicos que estejam sendo simulados.

2.3.2 Sistema de aquisição de dados de processos reais

Esse módulo consiste de uma estrutura de aquisição de sinais aplicável a uma grande gama de processos físicos reais. Para tal, tem-se um laboratório real com aquisição automatizada de dados trabalhando em processos com enfoques didáticos, e que fornecem dados para serem “medidos” pelo módulo “Simulador de instrumentos”. Prevê-se a evolução futura com a disponibilização contínua, previamente programada e divulgada, de experimentos em tempo real.

2.3.3 Servidor de dados de processos

Este módulo faz a obtenção dos dados dos módulos “Simulador de processos” e “Sistema de aquisição de dados de processos reais” e os disponibiliza de forma padronizada através de canais de comunicação.

Como meios de comunicação entre o “Servidor de dados de processos” e o “Simulador de instrumentos”, são empregadas formas de comunicação usuais, como serial, GPIB, TCP-IP, LAN e OLE. Essas interfaces permitem a comunicação tanto em casos em que o “Servidor de dados de processos” e o “Simulador de instrumentos” estejam separados por pequenas ou grandes distâncias, bem como quando ambos residem em um mesmo computador.

2.4 O simulador de instrumentos

O “Simulador de instrumentos” é o módulo através do qual é permitido ao usuário acessar informações disponibilizadas pelo “servidor de dados de processos”. Ele consiste de um sistema capaz de processar informações de temperatura, pressão, força, deslocamento, torque e demais grandezas de processos, se valendo de instrumentos simulados computacionalmente que procuram reproduzir de maneira fiel o comportamento de instrumentos reais de mercado[19].

A observação do comportamento de tais grandezas por meio desse simulador é feita através da montagem de cadeias de medição coerentes com a grandeza que se deseja investigar, aproximando sua funcionalidade aos sistemas reais de medição[19].

Os tipos de dados exigidos pelo simulador são aqueles que expressam grandezas de processos físicos em geral, tais como temperatura, pressão, deslocamento, entre outros. Eles independem de sua origem, podendo ser provenientes de sistemas reais implementados laboratorialmente ou através de módulos de simulação de processos.

2.4.1 Definição dos requisitos do ambiente de simulação e caracterização das linguagens de desenvolvimento

Para a definição dos requisitos do software de simulação, foram levantadas algumas características julgadas essenciais para o seu desenvolvimento. São elas:

- ser desenvolvido para execução em plataforma Windows;
- executar simulação de cadeias de medição seguindo a configuração: grandeza a medir → transdução → condicionamento de sinais → indicação;
- possuir uma biblioteca básica de instrumentos composta por transdutores, condicionadores de sinal e dispositivos mostradores;
- apresentar uma interface gráfica amigável e intuitiva, privilegiando a praticidade e a facilidade de utilização;
- oferecer riqueza de recursos gráficos para promover rápida interpretação e compreensão dos resultados simulados;
- executar a montagem de experimentos sem a necessidade de conhecimento de qualquer linguagem de programação;
- possibilitar a expansão futura, através da adição de novos módulos de instrumentos;
- ter facilidade de distribuição no meio acadêmico e industrial, de forma a não exigir a aquisição de softwares adicionais para seu funcionamento ou interpretação.

Com os requisitos definidos, é possível estabelecer critérios para a seleção da linguagem a ser utilizada para a implementação do simulador.

Sendo assim, para a seleção de ferramentas adequadas de desenvolvimento, é necessário traduzir os requisitos de software em especificações técnicas [18][21][32]. Essa ação resultou nas seguintes características desejáveis:

- grande poder de implementação de funções matemáticas;
- capacidade de geração de programas executáveis, com códigos otimizados e de tamanho reduzido;

- possibilidade de trabalhar com importação dinâmica de códigos, realizada no Windows através de integração com controles ActiveX e DLLs;
- facilidade e suporte para desenvolvimento de interfaces gráficas complexas;
- suporte para programação com abordagem baseada em formulários (forms) e orientada a objetos;
- estreita integração com a plataforma Windows de forma a possibilitar um melhor aproveitamento de seus recursos;
- disponibilidade de bibliotecas internas voltadas para a área de instrumentação e simulação;
- disponibilidade de ferramentas e componentes desenvolvidos por terceiros;
- facilidade de aprendizado da linguagem.

De forma a elucidar o inter-relacionamento entre os requisitos do software de simulação e as especificações desejadas para a linguagem de programação, estão explicitados na Tabela 2.1 as respectivas ligações entre esses dois tópicos.

Tabela 2.1. Inter-relações entre requisitos do simulador e características desejáveis da linguagem de implementação

Requisitos do simulador	Características desejáveis da linguagem
- Utilizar plataforma Windows	- Estreita integração com a plataforma Windows
- Executar a simulação de cadeias de medição sem a necessidade de qualquer programação e possuir biblioteca básica de instrumentos composta por transdutores, condicionadores de sinal e mostradores	- Poder de implementação de funções matemáticas - Disponibilidade de bibliotecas internas voltadas para a instrumentação e simulação - Disponibilidade de componentes desenvolvidos por terceiros
- Possuir interface gráfica amigável, intuitiva e rica em recursos	- Facilidade para o desenvolvimento de interfaces gráficas complexas - Integração com controles ActiveX e DLLs - Disponibilidade de componentes desenvolvidos por terceiros

Requisitos do simulador	Características desejáveis da linguagem
- Possibilitar a expansão futura, através da adição de novos módulos de instrumentos	- Integração com controles ActiveX e DLLs - Programação com possibilidade de utilização de orientação a objetos - Linguagem largamente difundida, com grande número de usuários - Facilidade de aprendizado da linguagem
- Ter facilidade de distribuição no meio acadêmico e industrial	- Possibilidade de geração de programas executáveis

2.4.2 Seleção das linguagens de programação do simulador

O processo de escolha de uma linguagem de programação consiste, logo de início, em decidir quais são os seus requisitos e a sua importância relativa, uma vez que é praticamente impossível satisfazê-los igualmente bem, com uma única escolha [20]. Sendo assim, as linguagens disponíveis devem ser comparadas em relação a uma lista de requisitos, que deve contemplar [21][32]:

- a complexidade computacional e algorítmica;
- o ambiente em que o software será executado;
- considerações de desempenho;
- o conhecimento da equipe de desenvolvimento do software;
- a disponibilidade de um bom compilador.

Sabendo que o escopo do projeto onde este trabalho se insere está voltado para o ensino e treinamento em metrologia, infere-se que seu foco de utilização se encontra dividido entre as escolas de nível técnico e superior e a indústria. Assim sendo, os motivos que levaram à escolha das linguagens de implementação são provenientes não somente de aspectos técnicos como os listados anteriormente, mas também permeiam questões econômicas e diretamente relacionadas aos principais utilizadores potenciais do simulador em questão.

Com base nos pontos definidos na Seção 2.4.1, na lista de requisitos apresentada anteriormente, na experiência da equipe que compõe o Labmetro e na disponibilidade dos softwares para desenvolvimento, foram selecionadas duas linguagens de programação que satisfizeram de forma suficiente os requisitos técnicos observados. São elas: Visual Basic 6.0 (Microsoft) [57][58][59] e Labview 6i (National Instruments) [46][47][48].

De forma a detalhar o processo realizado para a escolha das linguagens, pode-se observar na Tabela 2.2 uma comparação entre Labview e Visual Basic[2]. Essa comparação leva em consideração as características desejadas para a implementação do software de simulação, de acordo com a Tabela 2.1.

Tabela 2.2. Quadro comparativo entre Labview e Visual Basic

Característica Desejável	Labview 6i	Visual Basic 6.0
Integração com a plataforma Windows	Forte	Forte
Disponibilidade de funções matemáticas avançadas	Forte	Fraca
Disponibilidade de bibliotecas de instrumentos e de simulação	Muita	Pouca
Disponibilidade de componentes desenvolvidos por terceiros	Pouca	Muita
Integração com controles ActiveX e DLLs	Média	Alta
Capacidade para programação orientada a objetos	Não	Sim
Facilidade de aprendizado	Alta	Média
Facilidade para o desenvolvimento de interfaces gráficas complexas	Alta (*)	Média
Possibilidade de geração de programas executáveis	Sim (**)	Sim

(*): Caso o elemento de tela desejado não esteja previsto entre os apresentados pelo Labview, a criação de um específico é limitada. No entanto, em virtude da grande variabilidade de representações possíveis, pode-se classificá-lo como uma ferramenta com alto poderio gráfico.

(**): o Labview, apesar de gerar arquivos executáveis, apresenta uma série de limitações para essa operação.

2.4.3 Desenvolvimento de recursos de simulação em Labview e Visual Basic

Devido à abrangência dos problemas a serem resolvidos pelo simulador de instrumentos, decidiu-se separar sua implementação em duas frentes: uma desenvolvida em Visual Basic 6.0 e outra em Labview 6i.

Essa ação teve como princípio possibilitar um melhor ajuste no que diz respeito ao tipo de problema a ser enfrentado e a linguagem a ser utilizada para resolvê-lo.

Adicionalmente, a limitação e a inflexibilidade causada pela geração dos programas executáveis que o Labview é capaz de produzir, levaram à necessidade do desenvolvimento de um sistema equivalente em Visual Basic, conforme pode ser percebido pela análise da Figura 2.2. Entretanto, essa falta de flexibilidade é compensada pela velocidade de desenvolvimento dos modelos dos instrumentos na fase de “Elaboração de modelos gerais”.

Esse alto desempenho reduz o esforço no sentido da implementação e teste dos modelos matemáticos de simulação, e canaliza-o para a sua elaboração e definição. Essa otimização é então aproveitada, no momento da implementação do mesmo modelo em Visual Basic, transformando esse trabalho em apenas uma etapa de tradução do código, de Labview para Visual Basic.

Conseqüentemente, pôde-se extrair de cada linguagem, um maior rendimento quando na fase de implementação do sistema, resultando em mais rapidez e facilidade para seu desenvolvimento. Além disso, a velocidade de implementação proporcionada pelo Labview o coloca em um papel adicional de ferramenta prototipadora, uma vez que gera e testa modelos antes dos mesmos serem implementados e incorporados ao sistema desenvolvido em Visual Basic, incentivando o intercâmbio de informações entre os módulos do sistema de simulação.

De forma a obter uma maior flexibilidade no que tange à aplicabilidade e facilidade de distribuição do sistema, a concepção física do sistema de simulação de instrumentos foi dividida conforme a Figura 2.2[19].

Esses módulos tiveram seu desenvolvimento classificado de acordo com o nível de utilização do simulador a ser implementado, conforme indicado pela Figura 2.2. O esclarecimento dessa classificação é apresentado nos tópicos 2.4.4 e 2.4.5, quando é definida a terminologia adotada para o termo “experimento” e os tipos de modelo (geral e particularizado).

Cabe ressaltar que as informações fornecidas dizem respeito principalmente aos módulos com referência à linguagem Visual Basic. Os

módulos realizados com o auxílio do Labview não fazem parte do foco deste trabalho. Entretanto, muitas das informações aqui transmitidas podem ser estendidas sem prejuízo no entendimento.

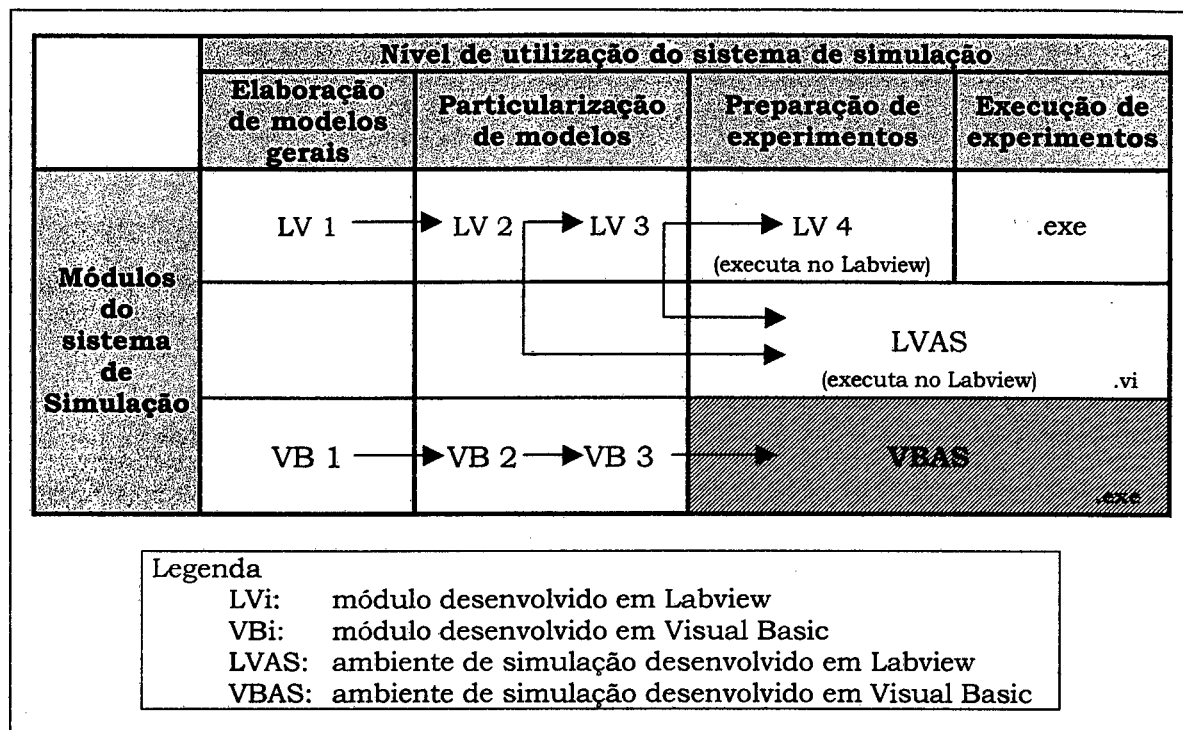


Figura 2.2. Concepção física do Sistema de Simulação de Instrumentos

2.4.4 Abordagem metrológica do ambiente de simulação

De maneira geral, uma cadeia de medição pode ser dividida em três módulos funcionais: o transdutor, o condicionador de sinais e o dispositivo mostrador[19][22][23]. Os transdutores ainda podem ser subdivididos em duas categorias[19]:

- os que necessitam de fonte externa de alimentação;
- os autogeradores, que não necessitam de fonte externa de energia.

Cada módulo pode constituir uma unidade independente ou pode estar fisicamente integrado ao sistema. A Figura 2.3 mostra essa configuração.

O simulador de instrumentos utiliza essa abordagem. De maneira mais específica, tem-se um simulador de cadeias de medição. Ele apresenta em sua concepção uma biblioteca de instrumentos destinada especialmente à

realização dessa atividade, contendo transdutores, condicionadores de sinal e dispositivos mostradores.

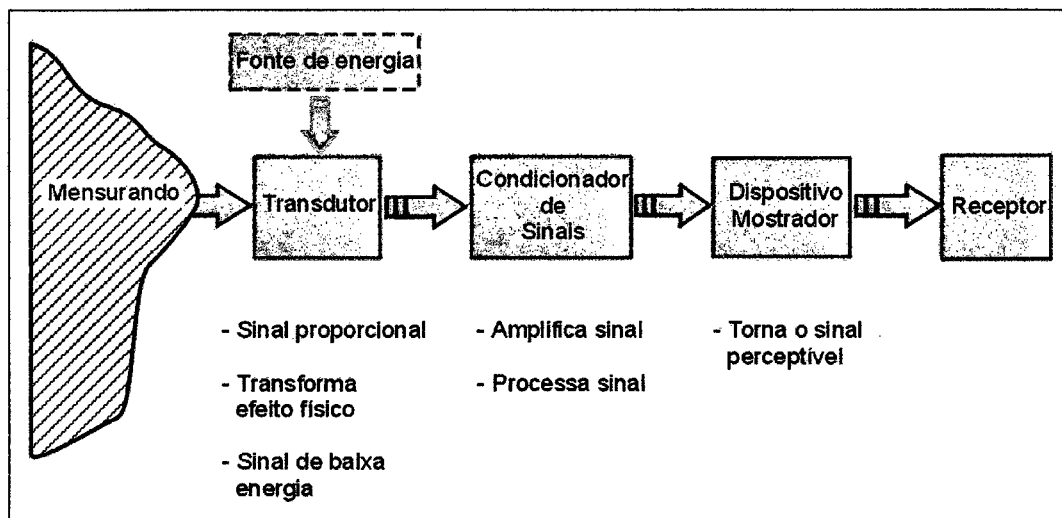


Figura 2.3. Cadeia de medição genérica

Seguindo essa abordagem, o simulador pode trabalhar com o conceito de “experimentos”, que são simulações de montagens com estrutura similar à mostrada na Figura 2.3. Isso é feito através da ligação dos elementos constitutivos do experimento através de ferramentas de edição do programa.

O software deve ser capaz de simular não somente os instrumentos constituintes dos módulos, mas também suas interações, de forma a contemplá-las na análise do resultado da medição ao final da cadeia. Essa abordagem é diferenciada, por considerar os elementos simulados imersos em um sistema que apresenta comportamentos distintos de acordo com sua configuração, não os considerando de forma estanque e isolada.

Os modelos matemáticos dos instrumentos presentes no simulador levam em consideração características particulares de erro representadas por efeitos sistemáticos e aleatórios [19][23][24][25][26][27][28][60]. Essas componentes são internas à modelagem, e são geradas através de distribuições estatísticas adequadas a cada parcela de erro. Dessa forma consegue-se um comportamento específico para cada instrumento utilizado em um experimento, representando assim a real variabilidade encontrada entre distintos equipamentos, mesmo quando oriundos de lotes equivalentes de fabricação.

2.4.5 Estrutura básica do ambiente de simulação de instrumentos

A construção do ambiente de simulação de instrumentos pode ser dividida em dois grandes blocos:

- ferramentas de edição: utilizadas para a construção da cadeia de medição a ser simulada. São as ferramentas responsáveis pela conexão entre os módulos. É através de seu intermédio que o usuário do software monta sua cadeia de medição ligando, movendo, removendo, inserindo módulos, entre outras operações.
- instrumentos: são os modelos matemáticos que procuram reproduzir o comportamento dos instrumentos simulados.

Na modelagem matemática dos instrumentos é necessário focar as características metrológicas mais significativas para o ensino da metrologia. Sendo assim, além da divisão anterior, foi estabelecida uma hierarquia para a particularização dos modelos dos instrumentos que compõe o sistema. Optou-se por separá-los em duas classes:

- Modelo geral: representa uma caracterização genérica do instrumento a ser simulado. Nele são disponibilizadas todas as entradas (efeitos sistemáticos e aleatórios, ajustes, ganhos, entre outros) que o modelo permite que o usuário especifique. É utilizado para criar as classes de instrumentos, como: voltímetros, amperímetros, termistores, entre outros.
- Modelo particularizado: corresponde ao modelo geral, com suas entradas definidas para um determinado tipo de instrumento. Vem como uma particularização do modelo genérico, feita através da especificação de parâmetros de fabricação do equipamento em questão. É voltado para instrumentos caracterizados por um tipo em particular ou marca, como: termopar tipo J do fabricante A, voltímetro modelo XX do fabricante B, e assim por diante.

Capítulo 3

Análise do Problema na Visão da Orientação a Objetos

Este capítulo tem como foco principal enunciar e detalhar os recursos necessários para o atendimento dos requisitos de software do sistema de simulação apresentados no item 2.4.1, com ênfase para o desenvolvimento utilizando a linguagem Visual Basic.

Neste sentido, inicialmente é apresentado o problema a ser resolvido, passando por peculiaridades que levam em consideração diversos aspectos inerentes ao desenvolvimento de um sistema computacional, como: finalidade do sistema, público alvo, usuários e benefícios.

Após, é discutida a Programação Orientada a Objetos, sendo a mesma proposta como metodologia de solução para a implementação do problema, enfatizando os aspectos positivos referentes à sua utilização.

Finalmente, é dada uma visão detalhada relativa à abordagem sugerida para a implementação dos recursos necessários ao desenvolvimento do software, contemplando quatro grupos de recursos definidos como principais no desenvolvimento do sistema: a Interface, as Ferramentas de simulação, os Instrumentos e os Recursos de análise.

3.1 Reconhecimento do problema de software

O reconhecimento do problema de software é uma investigação inicial com o intuito de estabelecer uma ligação entre as limitações de um projeto de software e a realidade do problema a ser solucionado [21].

Esse estudo possibilita um melhor conhecimento do problema a ser resolvido através do detalhamento da função que o software deve desempenhar. Assim sendo, é necessário entendê-lo em um contexto de sistema através do reconhecimento das peculiaridades referentes à sua utilização, seu público alvo e os benefícios esperados com a implementação do mesmo.

Com o auxílio da caracterização geral exposta no item 2.4, o detalhamento do problema de software pode ser direcionado pelo levantamento de informações através de questionamentos. São eles:

(a) Qual a finalidade/utilização do software?

O software é destinado ao ensino e treinamento em metrologia e instrumentação. Isso é feito através de técnicas de simulação de cadeias de medição com topologia igual à ilustrada na Figura 2.3. O sistema possibilita ainda a comparação entre diferentes montagens de instrumentos, permitindo o julgamento técnico e de viabilidade de uma configuração específica de um sistema de medição.

(b) Qual o público-alvo do sistema?

O sistema é direcionado tanto para instituições de ensino de nível técnico e superior, que ministram conteúdos que envolvem a metrologia ou a instrumentação, quanto para a Indústria que trabalha em setores envolvidos direta ou indiretamente com a metrologia, seja dimensionando, montando, testando ou analisando cadeias de medição.

(c) Quais são os seus usuários?

Os usuários do sistema são alunos (para instituições de ensino), pessoas sob treinamento (para empresas), bem como professores, engenheiros e instrutores de metrologia e instrumentação.

(d) Quais os benefícios, em relação aos laboratórios convencionais, que uma solução bem implementada traria?

Além das características advindas da técnica de simulação (conforme citado no item 2.1.1), o sistema proporciona os seguintes benefícios a serem destacados:

- atualização dos laboratórios de forma mais dinâmica e sinérgica com a evolução tecnológica;
- redução do custo de manutenção dos laboratórios;
- acesso da comunidade acadêmica a um número maior de instrumentos comparáveis ao estado da arte;
- possibilidade de realização de uma gama mais variada de experimentos, sem grandes acréscimos de custo;
- facilidade na execução dos experimentos, dispensando a necessidade de programação ou montagem física de equipamentos;
- possibilidade de implementação de módulos específicos para a situação de treinamento;
- melhor fixação do conteúdo-alvo do treinamento;
- capacidade para treinar um maior número de pessoas, sendo o limitador apenas a disponibilidade de computadores, incorrendo em menores custos com equipamentos.

De forma a facilitar a compreensão do resultado dessa análise, procurou-se ordenar as idéias captadas pelos questionamentos de uma forma sintética.

Sendo assim, assumindo as propostas referentes à finalidade e ao tipo de usuário do sistema, infere-se que um dos pontos mais críticos está localizado no projeto da interface do software. O caráter educacional exige que a intuitividade e a ergonomia sejam colocados em posição de destaque, de forma a envolver o usuário e convidá-lo ao seu autodesenvolvimento.

O ponto de análise referente ao público-alvo fornece indicativos que levam à necessidade de grande flexibilidade no que diz respeito à aplicabilidade do software, permitindo assim sua utilização nas mais diversas áreas de ensino e treinamento.

Com relação aos benefícios, pode-se notar a concordância dos itens listados com os tópicos já analisados. No entanto é necessário reforçar dois pontos fundamentais:

- um mecanismo de software que seja capaz de importar módulos adicionais, voltados para aplicações específicas da instituição de ensino ou empresa utilizadora do sistema;
- a necessidade de uma interface rica em recursos gráficos.

Com base no conjunto de informações obtido, é possível definir os recursos necessários para o desenvolvimento do sistema. No entanto, neste tópico do trabalho optou-se apenas por organizá-los em grupos, deixando seu detalhamento para o item 3.3.

O estabelecimento dessa divisão veio com o intuito de facilitar a abstração do problema, bem como classificar seus elementos por similaridade. Assim, os grupos formados foram:

- Interface: envolve os itens que possibilitam a interação direta do usuário com o simulador;
- Ferramentas de simulação: está relacionado diretamente com a recuperação, o andamento e a criação de simulações;
- Instrumentos: compreende os modelos referentes aos instrumentos do simulador;
- Recursos de análise: contempla ferramentas estatísticas e matemáticas com o objetivo de auxiliar na interpretação dos resultados simulados.

3.2 Programação orientada a objetos com foco no ambiente de simulação

Em programas de simulação freqüentemente são necessárias alterações no que diz respeito a inclusões ou melhorias de modelos [35][39].

As linguagens de programação puramente estruturadas apresentam dificuldades quanto à manutenção e à atualização desses programas. Adicionalmente, seus códigos tendem a se tornar confusos para programas de médio e grande porte. Essa situação se agrava após diversas atualizações,

sobretudo se são feitas por diferentes programadores. Dessa forma, fazem-se necessárias técnicas de programação que permitam eficiente implementação e atualização de programas de médio e grande porte [39][33].

Para solucionar esses problemas e suprir a necessidade de se produzir softwares cada vez mais atraentes, dinâmicos e com alto poder de troca de informações, a metodologia orientada a objetos passou a ser adotada por grande parte dos profissionais da área de desenvolvimento de sistemas [29][34].

Apesar de não ser um conceito totalmente novo no meio acadêmico, somente nos últimos anos a orientação a objetos vem ganhando força no mercado de software. Para constatar esse fato, basta citar que as grandes empresas da área de desenvolvimento de sistemas oferecem ferramentas se não totalmente fundamentadas, pelo menos baseadas no conceito de objeto [30][45].

A programação orientada a objetos segue uma estrutura fundamentada nos conceitos de classe e objeto. O termo objeto é utilizado para representar um determinado elemento do mundo real. Ele apresenta, em uma estrutura única, características estruturais e comportamentais, conhecidas respectivamente como propriedades e métodos.

As propriedades podem ser fixas ou variáveis. Elas determinam os possíveis estados de um objeto, ou seja, seu conjunto de valores em um determinado instante. Analogamente, os métodos determinam os possíveis comportamentos de um objeto, explicando como ele age e reage em termos de suas mudanças de estado e troca de mensagens com outros elementos.

Os objetos têm identidade e são instâncias de classes. Fundamentalmente, eles são entidades abstratas que encapsulam estado e comportamento. Já as classes são descrições de objetos com características comuns de implementação [30]. Elas têm seu foco direcionado para a implementação de características estruturais e comportamentais. Na Figura 3.1 é apresentado um exemplo de um conjunto classe/objeto contemplando os atributos e métodos de uma classe hipotética.

Alguns requisitos gerais presentes no ambiente de simulação fizeram com que a técnica de orientação a objetos fosse considerada como a opção de desenvolvimento mais adequada frente à programação puramente estruturada.

Os requisitos que direcionaram fortemente para essa decisão foram:

- necessidade permanente de alteração e expansão;
- necessidade de se trabalhar com ferramentas desenvolvidas por terceiros através da programação orientada a objetos;
- maior padronização de elementos de software em razão de trabalho em equipe;
- facilidade para compreensão e documentação do software, em vista de possíveis e freqüentes mudanças na equipe de implementação.

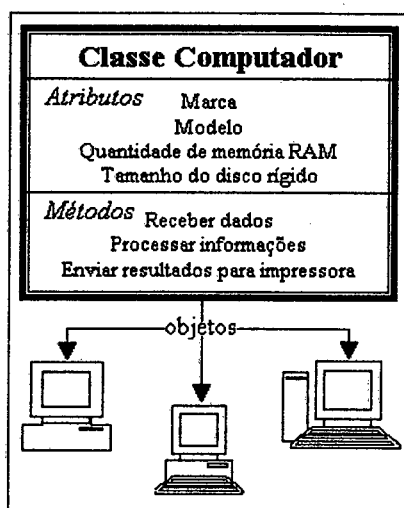


Figura 3.1. Exemplo de classe e objeto

A tecnologia orientada a objetos é fundamentada no “modelo de objetos”, que engloba, de forma sinérgica, os princípios pertinentes a essa metodologia [30].

Para uma melhor compreensão, é apresentada uma breve conceituação de quatro importantes princípios envolvidos, seguido pelos benefícios de suas respectivas aplicações ao problema em questão. São eles [30][31][39]:

- **Abstração:** diz respeito à formulação de classes de dados, onde são definidas apenas as características gerais dos objetos que a ela pertencem, sem implementação de código. A classe abstrata define apenas “o que fazer”, deixando as classes descendentes implementarem o “como fazer”. O processo de abstração procura criar representações focadas nas similaridades e diferenças entre um conjunto de entidades, a fim de extrair características relevantes à solução de um problema e evitar aspectos considerados irrelevantes.

- **Encapsulamento:** a maioria das linguagens de programação “procedurais” utiliza uma metodologia composta por uma base de dados passiva e várias sub-rotinas ou procedimentos que atuam sobre ela. Ou seja, nesse conjunto, os elementos ativos são as sub-rotinas. Na programação orientada a objetos, essa filosofia é invertida, sendo os dados os elementos ativos, uma vez que possuem sub-rotinas que os manipulam em sua própria estrutura. Essa peculiaridade de agrupamento denomina-se encapsulamento de dados, sendo uma das principais características da programação orientada a objetos.
- **Herança:** é o mecanismo pelo qual uma classe obtém as características e métodos de outra para expandi-la ou especializá-la de alguma forma.
- **Polimorfismo:** facilidade pela qual vários objetos diferentes podem possuir métodos com mesmo nome, geralmente associados a uma tarefa semelhante, porém com implementações distintas.

Os benefícios mais relevantes proporcionados pelos quatro princípios citados podem ser vistos na Tabela 3.1.

Tabela 3.1. Princípios e benefícios da programação orientada a objetos

Princípio	Benefícios
Abstração	<ul style="list-style-type: none"> – Proporciona a modelagem do mundo real, contribuindo para a redução da complexidade e para o melhor entendimento da estrutura do software; – Define uma padronização para as interfaces de objetos correlacionados, permitindo que um programa seja escrito sem o conhecimento detalhado das partes que o compõe; – Separa a implementação da abstração em si, impedindo que uma modificação interna afete o comportamento abstrato do objeto; – Facilita o processo de modificação, atualização ou otimização, pois os detalhes de implementação são concentrados no procedimento local; – Impede que a modificação na implementação de um procedimento afete outro componente do programa; – Agrupa informações correlatas e simplifica o tratamento de dados complexos.

Princípio	Benefícios
Encapsulamento	<ul style="list-style-type: none"> - Projeto pode ser dividido em componentes com características similares (classes); - a escrita e a manutenção do código-fonte de um objeto pode ser feita de maneira independente em relação a outros objetos; - os objetos podem ser distribuídos mais facilmente, podendo ser utilizados em outros programas, refletindo em ganhos de produtividade e tempo; - os detalhes de implementação do objeto podem ser ocultados, com a comunicação estabelecida através de métodos, sendo necessário para tal apenas o conhecimento de sua interface.
Herança	<ul style="list-style-type: none"> - os objetos que compõem o sistema podem ser aproveitados em outros sistemas, contribuindo para a redução do retrabalho; - desenvolvimento das classes pode ser feito de maneira incremental, indo do geral ao específico; - quando modificações são feitas na classe hierarquicamente superior, todas as descendentes automaticamente herdam essas alterações.
Polimorfismo	<ul style="list-style-type: none"> - aumento da flexibilidade e da reutilização de código através da criação de componentes genéricos de software; - permite uma padronização dos métodos com a mesma finalidade para objetos distintos; - facilita a documentação, legibilidade e organização do software.

A Tabela 3.1 traz características que são próprias da orientação a objetos, mas focadas no desenvolvimento do sistema em questão. Essa caracterização também vem reforçar o conceito de que a orientação a objetos foi a escolha mais coerente e prática para o desenvolvimento do simulador de instrumentos.

3.3 Proposta de implementação da solução do problema de software

Levando-se em consideração a classificação estabelecida no item 3.1, foi definido um conjunto de elementos constitutivos pertinentes a cada grupo de recursos. Para auxiliar nessa definição e também na solução de como os requisitos seriam atendidos, foram analisados dois populares simuladores de circuitos elétricos presentes no mercado: Pspice [36][37][38] e Electronics

Workbench [40]. Dessa análise foi possível retirar características referentes a aspectos como funcionalidades, interface e ferramentas de análise. Com a adaptação das características coletadas à realidade de um software de simulação de sistemas de medição, chegou-se na configuração detalhada nos tópicos seguintes. Para cada elemento é destacada sua função em relação ao sistema de simulação, bem como é estabelecido um paralelo com os requisitos enunciados em 2.4.1.

3.3.1 Interface

Os recursos de interface têm como principal foco atender aos requisitos de intuitividade e facilidade de uso. Isso obrigou o dimensionamento de ferramentas que possibilitassem a interação do usuário com o sistema de forma totalmente gráfica. É baseado nessa premissa que surgiu o conceito de janela de simulação.

A janela de simulação é o local onde são montados e executados os experimentos, sem a necessidade de qualquer programação. Ela é composta por três elementos básicos: os instrumentos, os fios e os ambientes de simulação. Esses elementos têm a função de fornecer suporte na construção, montagem e execução das simulações.

O conjunto de elementos integrantes do grupo definido como Interface é representado por:

1. **Ferramentas de edição (parte gráfica):** auxiliam na montagem do experimento e na disposição dos componentes na janela de simulação.
 - 1.1. **Fio:** responsável pela ligação entre os componentes integrantes de uma simulação.
 - 1.2. **Seleção:** recurso responsável pela seleção de objetos presentes em uma mesma janela de simulação. Pode selecionar várias estruturas através da definição de uma região de interesse, ou individualmente, pelo simples clique sobre o objeto.
 - 1.3. **Edição:** ferramentas que possibilitam editar a montagem de experimentos na janela de simulação.

- 1.3.1. **Mover:** movimenta os objetos na janela de simulação, possibilitando melhorar a disposição e a organização dos módulos presentes.
 - 1.3.2. **Excluir:** possibilita a eliminação de objetos presentes na tela de simulação.
 - 1.3.3. **Recortar/Colar:** exclui e insere objetos na tela de simulação.
- 2. **Menu:** menu horizontal do tipo “pull-down” contendo funções auxiliares do sistema de simulação, como ajuda, salvar, abrir, entre outras.
- 3. **Barras de ferramentas:** barras configuráveis e flutuantes dispostas na direção horizontal e vertical.
 - 3.1. **Horizontal:** contém as ferramentas de edição com possibilidade de adição de novas funcionalidades.
 - 3.2. **Vertical:** contém os instrumentos, com possibilidade de adição de novos elementos de simulação. É através dessa barra que os instrumentos serão inseridos na janela de simulação através do mecanismo de arrastar e soltar.
- 4. **Ambientes:** uma janela de simulação pode conter em seu interior vários ambientes. O ambiente é a estrutura que define o meio no qual um experimento ou instrumento está imerso. Ele é configurável de acordo com o tipo de instrumento ou experimento que estiver sendo simulado, podendo considerar inicialmente grandezas tais como temperatura (ambiente), pressão e umidade. A utilização dessa estrutura possibilita a comparação entre diferentes condições de simulação para uma mesma cadeia de medição, enriquecendo a análise e aumentando a aplicabilidade do sistema.
- 5. **Ajuda:** auxílio a respeito da utilização do sistema.

3.3.2 Ferramentas de simulação

O grupo Ferramentas de simulação contém elementos que possuem papel fundamental na realização das simulações. Ele procura suprir as necessidades de uma interface amigável e os requisitos de uma possível expansão do elenco de instrumentos. Para isso, esse grupo contempla itens que dão suporte à tarefa de simulação, possibilitando a recuperação e o

arquivamento de experimentos em andamento, a entrada e saída de dados dos experimentos, bem como a importação de novos módulos de programa.

O grupo Ferramentas de simulação é composto pelos seguintes elementos:

1. **Salvar/carregar:** operações que permitem a recuperação futura de experimentos já finalizados ou em processo de montagem.
2. **Importar instrumentos:** possibilita a importação de instrumentos adicionais segundo uma estrutura definida de procedimentos de implementação em Visual Basic.
3. **Painel de entrada de dados:** é através desse painel que são fornecidos os dados de origem necessários para a execução da simulação. No caso da simulação de uma cadeia de medição com um termopar, por exemplo, a entrada é a temperatura; no caso de medição com extensômetros, é a deformação; e assim por diante.
4. **Indicadores gráficos:** são gráficos que possibilitam melhor interpretação, análise e entendimento da cadeia de medição que estiver sendo simulada.

3.3.3 Instrumentos

O grupo Instrumentos contém estruturas que apresentam terminais de entrada e saída de informações, caracterizados por pontos onde a ligação de um fio é possível. Seus elementos são considerados fundamentais no processo de simulação, pois contêm os modelos matemáticos referentes a cada instrumento a simular.

O termo modelo pode ser definido (relativo à modelagem e simulação) como sendo uma “aproximação, representação ou idealização de aspectos de uma estrutura, comportamento, operação, ou outra característica de um processo real, conceito ou sistema” [41].

A análise dessa definição leva à conclusão de que a construção de modelos destinados à simulação exige como pré-requisito um conhecimento minucioso do princípio de funcionamento e das características metrológicas e operacionais do instrumento a ser modelado.

A grande diversidade de instrumentos presentes no mercado, e principalmente, a não padronização das especificações técnicas, dificultam a

tarefa da modelagem. Esses aspectos acabam por exigir profundo conhecimento das áreas de instrumentação e metrologia, bem como da análise de circuitos eletro-eletrônicos.

Paralelamente a este trabalho, foi desenvolvido um método para modelagem de instrumentos, destinado especificamente para simulação. Esse método está detalhado com rigor nas referências [42] e [43] e é apresentado de forma sucinta a seguir. Ele é baseado em especificações de fabricantes e na análise criteriosa de características representativas do comportamento metrológico e operacional de instrumentos.

É importante ressaltar que esse método atende à classificação hierárquica dos modelos proposta pelo item 2.4.5, na qual os mesmos são classificados em:

1. **Modelos gerais:** representa uma caracterização genérica do instrumento a ser simulado.
2. **Modelos particularizados:** corresponde ao modelo geral, com suas entradas definidas para um determinado tipo de instrumento.

O método sugere que cada modelo contenha as características estáticas e dinâmicas, visando representar o instrumento real e suas não-idealidades. Nesse sentido, todas as especificações da folha de dados técnicos de instrumentos devem ser analisadas e associadas a parâmetros característicos, de forma a possibilitar a implementação de um modelo para cada parâmetro.

A Figura 3.2 apresenta o formato e o procedimento de parametrização de um modelo para simulação. Esse procedimento está sistematizado em quatro módulos: “Experimento”, “Especificação do Fabricante”, “Cria” e “Modelo”.

A coluna denominada “Especificação do Fabricante” contém as especificações apresentadas na folha de dados do instrumento. Entretanto, existem entradas obrigatórias do modelo, que são atribuídas apenas quando se utiliza o instrumento para a realização de uma determinada medida, ou seja, quando se configura um experimento. Esses dados são apresentados na coluna “Experimento”, e referem-se a parâmetros como o mensurando e o meio em que o instrumento será utilizado, a tensão ou a corrente de alimentação, a faixa de frequência que se deseja medir, entre outros.

Dados desse tipo são identificados no modelo pela sigla “EXP”, indicando que tais informações são fornecidas pelo usuário ao configurar um experimento. Portanto essa coluna deve conter todos os dados pertinentes à configuração em que se deseja utilizar o instrumento para uma dada aplicação ou experimento.

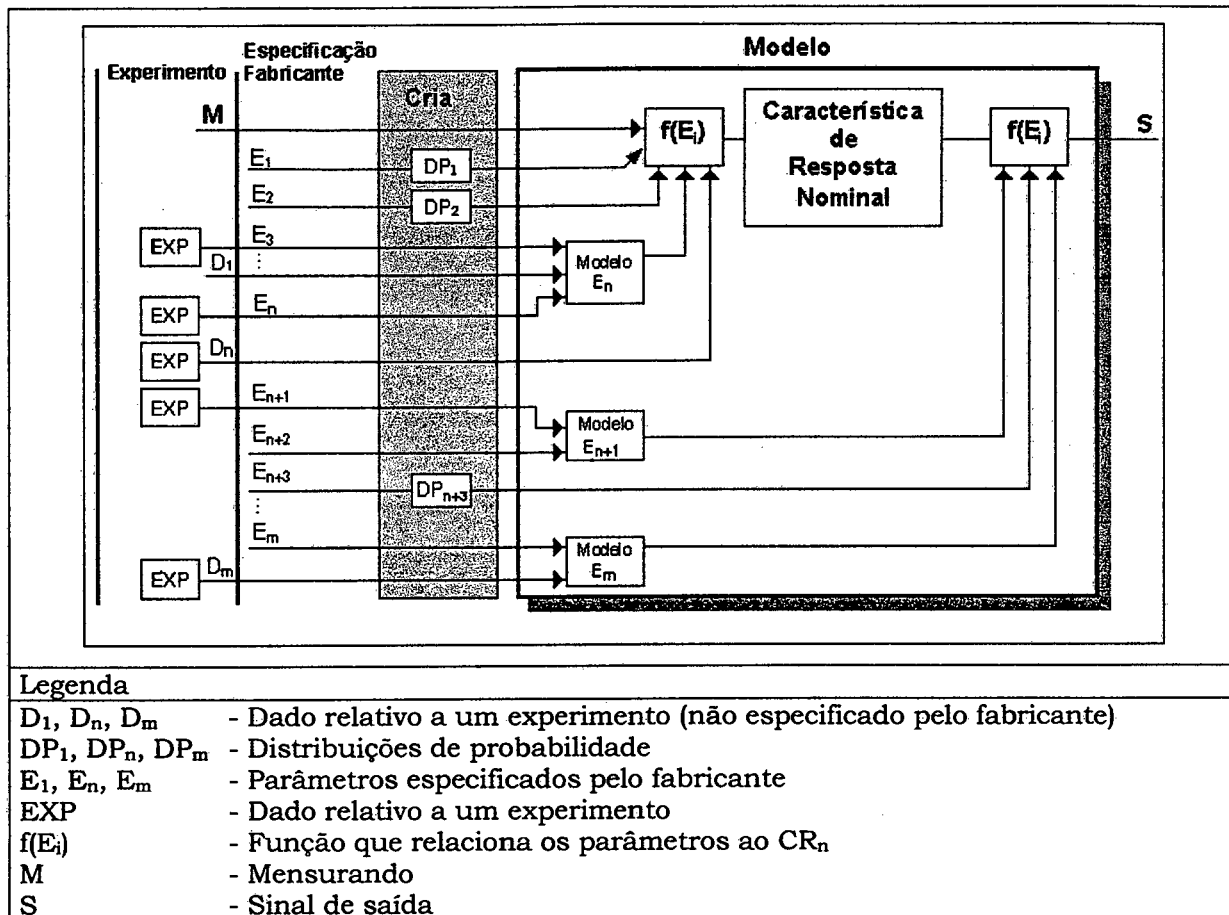


Figura 3.2. Padrão para parametrização de modelos de simulação

Da experiência prática em atividades laboratoriais, sabe-se que quando se analisa um lote de instrumentos ou componentes eletrônicos do mesmo modelo e do mesmo fabricante, encontram-se variações em seus parâmetros. No entanto, essas variações devem estar dentro de limites definidos em suas especificações. Para simular tal condição, são gerados valores aleatórios de distribuições de probabilidade que melhor representam esses dados. O tipo de distribuição atribuído ao dado é identificado na coluna “Cria”, através de uma sigla (DP₁, DP_n, DP_m). Dessas distribuições são escolhidos os valores específicos para cada instrumento de um lote de instrumentos gerado pelo modelo.

O bloco denominado “Modelo” é implementado a partir da característica de resposta nominal do instrumento em questão e de seus parâmetros. Esses parâmetros influenciam o comportamento do instrumento se manifestando antes ou depois da aplicação do modelo de resposta nominal. Esse detalhe pode ser visualizado através da representação da estrutura $f(E_i)$ na Figura 3.2.

A forma modular estabelecida para a implementação dos modelos facilita a criação e configuração de novos, pois muitos dos parâmetros já modelados para um determinado instrumento são características também presentes em vários outros.

De forma a ilustrar a utilização do método apresentado, é exemplificado a seguir o modelo de um termistor do tipo 44007, do fabricante Omega [44].

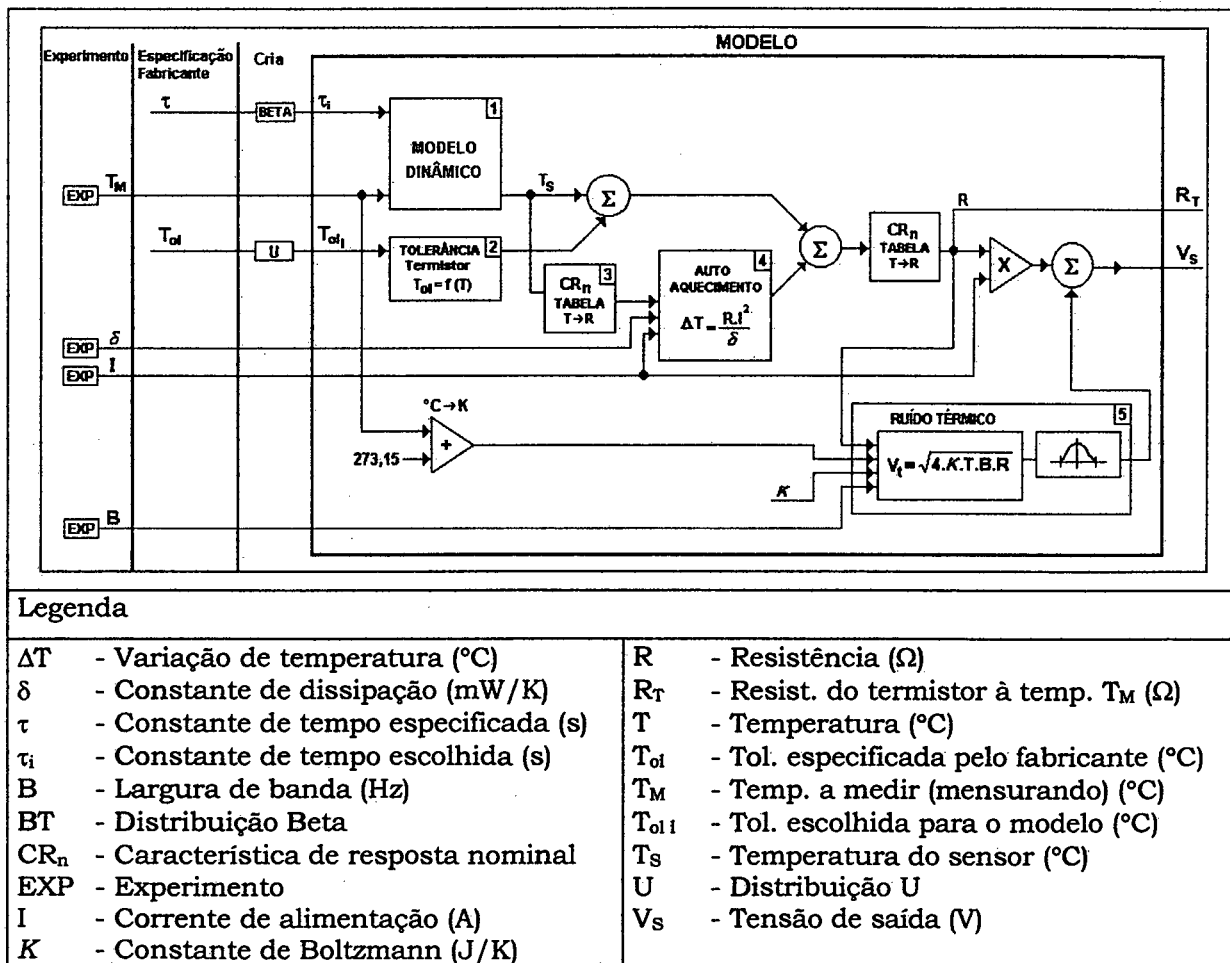


Figura 3.3. Modelo para simulação de um termistor

O modelo ilustrado na Figura 3.3 tem como entrada o mensurando temperatura (T), e como saída a resistência equivalente a tal temperatura (R_T).

Para facilitar sua aplicação na simulação de experimentos reais, o modelo do termistor incorpora uma entrada para a corrente de alimentação (I) e uma saída em tensão (V_s).

Tem-se ainda como entrada, parâmetros sistemáticos como a constante de tempo e a tolerância do termistor, além de dados que deverão ser fornecidos no momento da utilização do instrumento em um experimento, como a constante de dissipação (δ) e a largura de banda (B), ambos identificados no modelo com a sigla "EXP".

Conforme a Figura 3.3, o modelo é implementado através da característica de resposta nominal do termistor (módulo 3), do modelo da tolerância (módulo 2), do modelo dinâmico (módulo 1) e das fontes de erro relativas ao auto-aquecimento (módulo 4) e ruído térmico (módulo 5). Maiores detalhes podem ser encontrados nas referências [42] e [43].

3.3.4 Recursos de análise

Esse grupo contém ferramentas básicas de análise estatística e matemática auxiliares do sistema. Ele vem com a intenção de aumentar a interatividade e facilitar a interpretação dos resultados, transformando dados em informação.

Os recursos de análise são formados por ferramentas de utilização freqüente na metrologia, como calculadores de média, desvio padrão e variância. Pretende-se, gradativamente, com o lançamento de novas versões do simulador, incorporar funções mais sofisticadas de análise, proporcionando um leque mais variado de opções para o usuário.

Para a implementação dessas funções, pretende-se utilizar um grupo de componentes ActiveX denominado Component Works [47][49], da National Instruments, de forma a reduzir o tempo de implementação desses recursos.

Capítulo 4

Implementação do Ambiente de Simulação

Este capítulo visa apresentar os detalhes envolvidos no processo de implementação do ambiente de simulação, constatando na prática, a utilização do Visual Basic para a realização dessa tarefa.

Inicialmente é apresentada a interface gráfica do sistema de simulação, com a indicação das principais estruturas definidas para o software em questão.

Após essa apresentação, é feito o detalhamento técnico da implementação das classes desenvolvidas neste trabalho para o ambiente de simulação: `clsElemento`, `ctlInstrumento`, `ctlLinhaReta`, `clsFerramenta`, `clsInstrumento`, `clsTerminal`, `clsLigação` e `frmSimul`.

Em seguida são apresentadas as principais dificuldades enfrentadas no processo de desenvolvimento do ambiente, passando por tópicos que levam em consideração a linguagem Visual Basic, o relacionamento bidirecional entre objetos e a interligação entre elementos da janela de simulação.

Finalmente, é feita a análise e a avaliação das soluções mais críticas assumidas no desenvolvimento do sistema, onde são fornecidos detalhes e informações que dizem respeito às possíveis imperfeições das soluções implementadas.

4.1 A interface do ambiente de simulação

O ambiente de simulação de instrumentos apresenta uma interface implementada de forma similar à grande maioria dos simuladores e aplicativos baseados no sistema operacional Windows.

Essa interface encontra-se em estágio de desenvolvimento, e foi implementada de forma simplificada. No entanto, ela caracteriza com bastante propriedade o objetivo final de se obter uma interface amigável e funcional para o usuário do simulador.

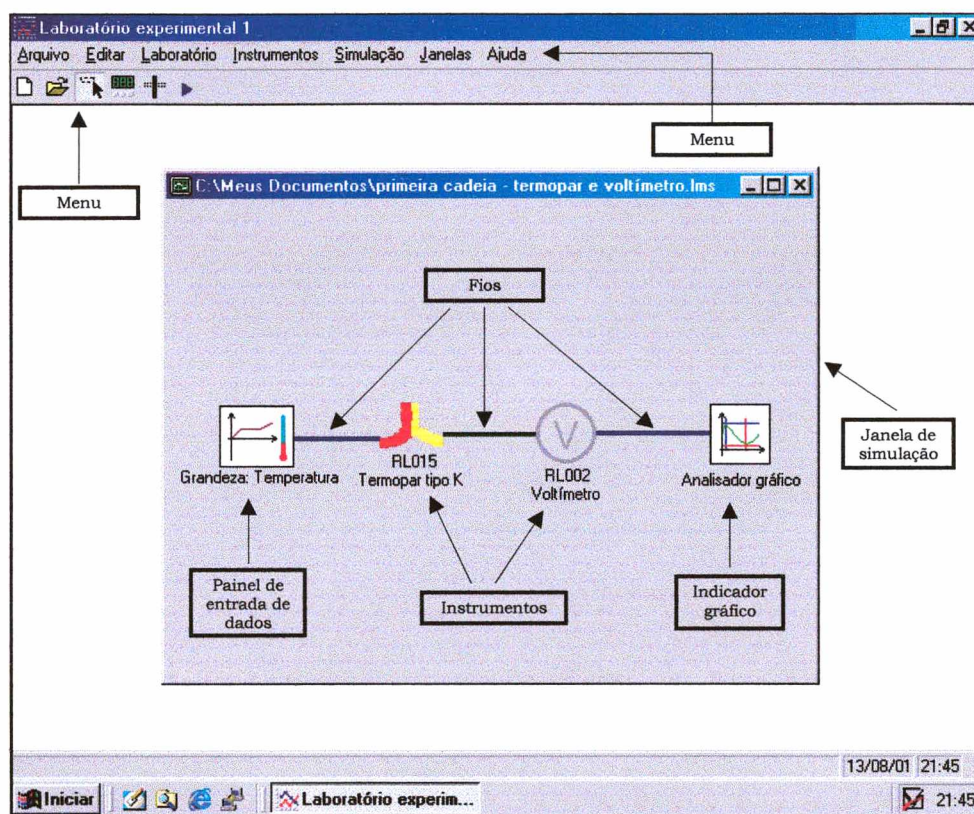


Figura 4.1. O ambiente de simulação

Sendo assim, de acordo com a proposta definida no item 3.3, a interface gráfica apresenta as seguintes estruturas:

- Elementos de tela
 - Instrumentos
 - Painel de entrada de dados
 - Indicador gráfico
- Menus

- Ferramentas de edição
 - Fio
 - Seleção
 - Edição: mover e excluir.

No exemplo da Figura 4.1, pode ser visualizada uma janela de simulação com os principais elementos de tela do simulador. Eles fazem parte do que foi chamado de núcleo operacional do sistema e têm sua implementação detalhada no item 4.2.

Além dos elementos mostrados pela Figura 4.1, para a montagem das cadeias de medição é necessária a existência de ferramentas de edição. Os fios já foram representados na Figura 4.1, mas há ainda as ferramentas que realizam as operações de seleção, movimentação e exclusão dos elementos da tela.

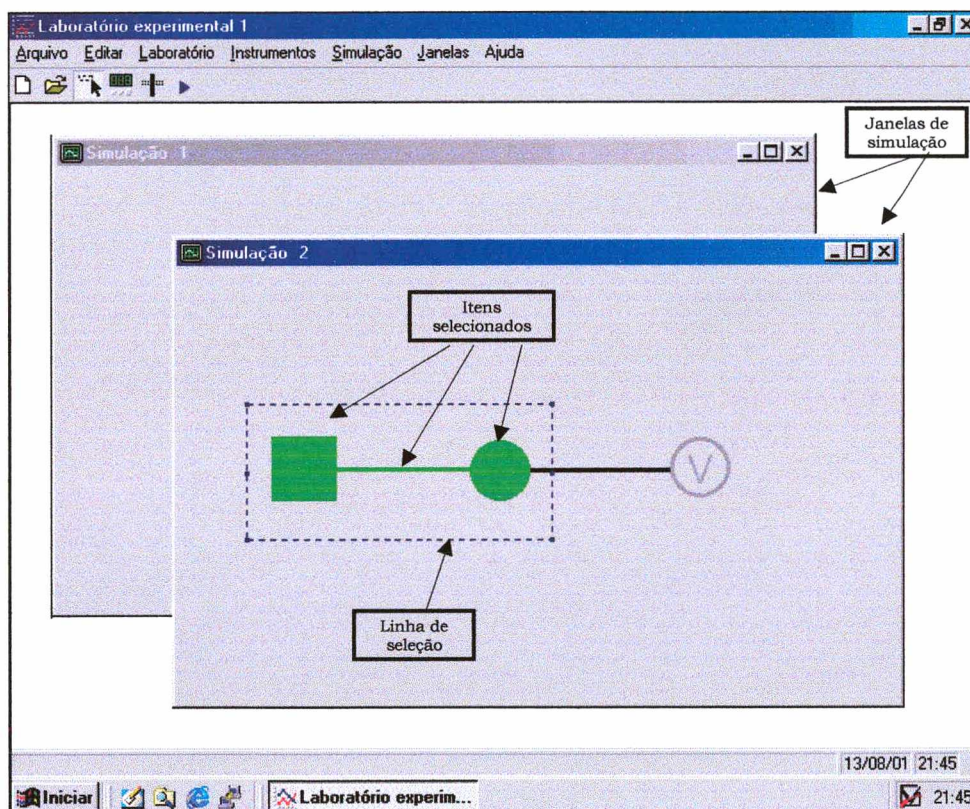


Figura 4.2. A edição no ambiente de simulação

Na Figura 4.2 são mostradas duas janelas de simulação. Na janela intitulada “Simulação 2” pode ser vista a edição de um experimento, com a

ferramenta de seleção sendo utilizada. Isso resulta na modificação da representação gráfica dos instrumentos (“Highlight”), como forma de indicar quais elementos estão selecionados para a posterior realização de uma operação de movimentação ou exclusão.

Com o gradativo desenvolvimento do sistema, novos componentes serão adicionados, tornando a interface mais funcional e intuitiva para o usuário. Isso inclui uma variedade maior de instrumentos, de indicadores, bem como de recursos de interface, tais como menus flutuantes e ferramentas de edição mais versáteis.

4.2 O núcleo operacional do sistema

O núcleo operacional do sistema é formado por um conjunto de classes responsáveis por viabilizar a execução das principais atribuições do software de simulação. Sua estrutura foi totalmente baseada na abordagem de orientação a objetos e inclui classes correspondentes à interface, às ferramentas de simulação e as que são diretamente relacionadas aos instrumentos. Com isso, conseguiu-se cobrir parte dos requisitos de software, bem como dos recursos enunciados no Capítulo 3.

Neste trabalho optou-se pela utilização de classes abstratas com o intuito de generalizar o tratamento a ser oferecido aos diversos tipos de objeto presentes no sistema. Com isso, grupos de componentes foram definidos com base na similaridade de ação no software, bem como de programação. Assim sendo, as características inerentes aos componentes desses grupos auxiliaram na definição dos métodos e propriedades relativas às classes desenvolvidas.

Essa estruturação culminou na definição de cinco grupos:

- Elementos de tela;
- Ferramentas de edição;
- Instrumentos;
- Terminais;
- Ligações.

Dessa forma, estruturou-se o núcleo do sistema com as seguintes classes:

- clsElemento (abstrata)
 - ctlLinhaReta
 - ctlInstrumento
 - eleJanela
- clsFerramenta (abstrata)
 - ferFio
 - ferSeleção
 - ferInstrumento
- clsInstrumento (abstrata)
 - insTermopar
 - insVoltmetro
- clsTerminal
- clsLigação
- frmSimul

Estabelecendo uma relação entre as classes desenvolvidas e os recursos do sistema enunciados no item 3.3, resulta a Tabela 4.1

Tabela 4.1. Relação entre os recursos do sistema e as classes do núcleo operacional

Grupo de recursos do sistema	Classes do núcleo operacional
Interface	Classe clsElemento - ctlLinhaReta Classe clsFerramenta - ferFio - ferSeleção - ferInstrumento Classe clsLigação Classe frmSimul
Ferramentas de simulação	Classe clsElemento - ctlInstrumento Classe frmSimul
Instrumentos	Classe clsElemento - ctlInstrumento Classe clsInstrumento Classe clsTerminal Classe frmSimul

As classes componentes do núcleo operacional apresentam inter-relacionamentos necessários para o estabelecimento da dinâmica de

funcionamento do sistema. Essas relações são mostradas de forma sucinta na Figura 4.3.

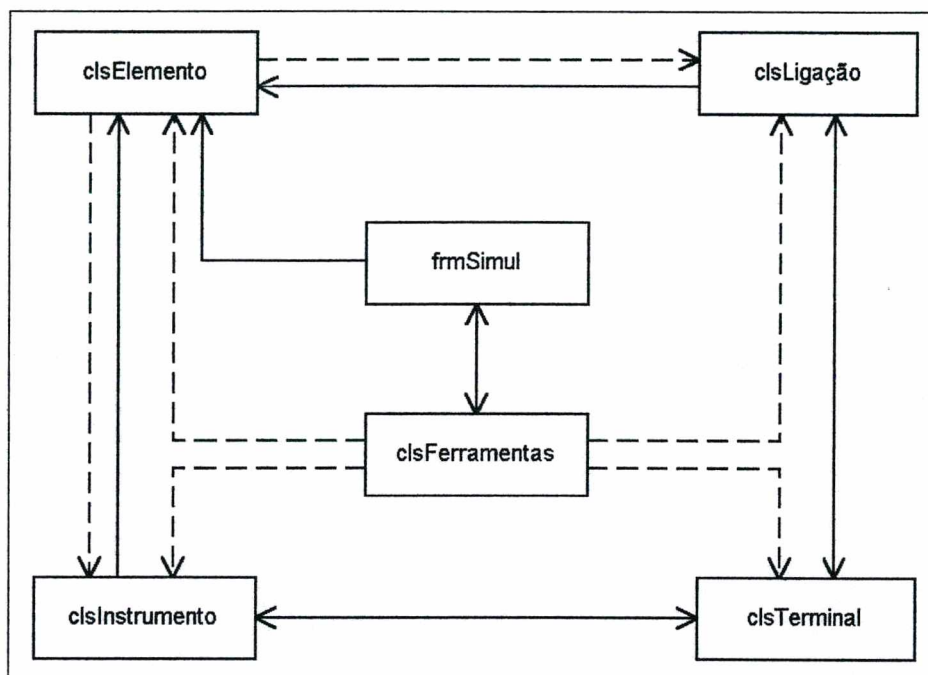


Figura 4.3. Inter-relacionamento entre classes do sistema

Para simplificar essa representação, optou-se por considerar no gráfico apenas as classes abstratas `clsElemento`, `clsInstrumento` e `clsFerramenta`, e as classes `clsLigação`, `clsTerminal` e `frmSimul`. No diagrama, as linhas contínuas representam relação direta entre as classes, enquanto as tracejadas indicam relacionamento parcial, com dependência das classes que implementam os métodos e propriedades da classe abstrata em questão.

O detalhamento das classes é apresentado nos itens 4.2.1 até 4.2.8. Isso é feito com o auxílio de diagramas que indicam métodos, propriedades e a hierarquia envolvida na sua estruturação. Seu escopo funcional é apresentado com detalhes contendo, para cada classe, a explicação dos principais métodos e propriedades, bem como o seu contexto no ambiente de simulação em desenvolvimento.

4.2.1 Classe `clsElemento`

A classe `clsElemento` foi criada para generalizar e simplificar o tratamento dado aos objetos apresentados na janela de simulação, bem como

permitir a adição de possíveis elementos novos ao sistema, sem a necessidade de complexas alterações no software. É uma classe abstrata (interface), ou seja, não pode criar objetos. Os “user controls” “CtlLinhareta” e “CtlInstrumento”, e a classe “EleJanela” implementam seus métodos e propriedades, e por isso, podem ser tratados de forma similar no sistema.

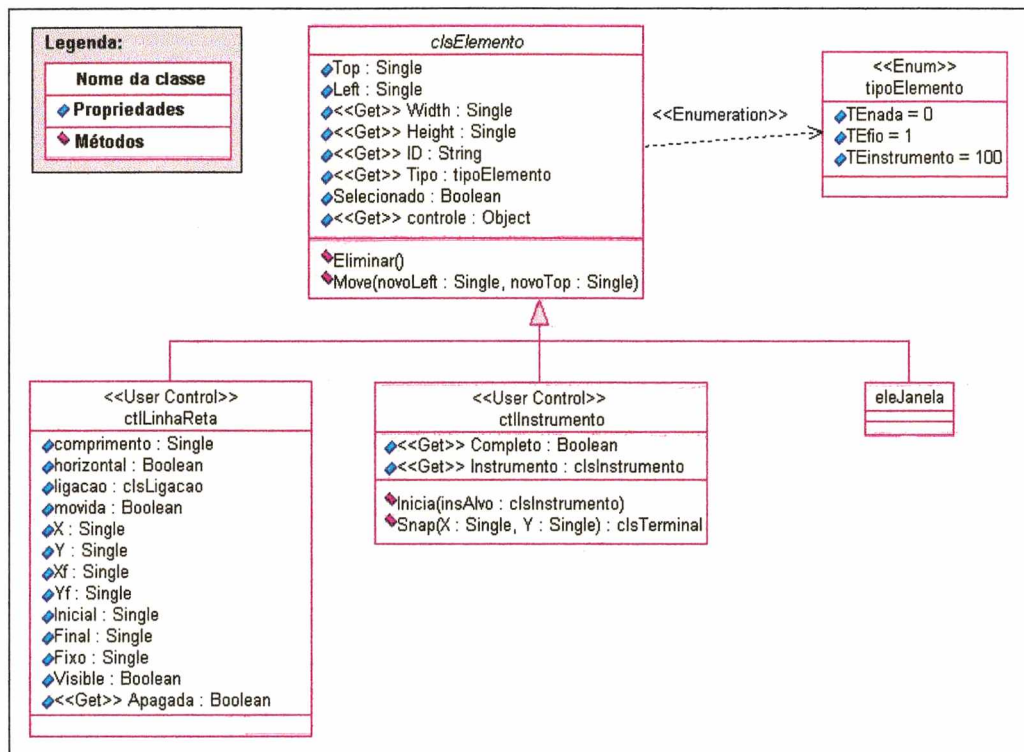


Figura 4.4. As classes clsElemento, ctLinhaReta, ctInstrumento e eleJanela

As principais propriedades desta classe foram definidas por comparação com os controles disponibilizados pelo Visual Basic. As propriedades “Top”, “Left”, “Width” e “Height” são exemplos disso, caracterizando-se como simples cópias de propriedades homônimas dos controles. O método “Move” se diferencia do ajuste direto de “Top” e “Left” por ajustar ambos ao mesmo tempo. Isso evita uma indesejável posição transitória resultante da alteração independente e seqüencial desses parâmetros.

Para que se possa ter um tratamento específico sobre cada tipo possível de objeto, foi criada a propriedade “Tipo”. De forma a simplificar sua utilização, essa propriedade foi implementada através de dados enumerados. Com esse

artifício o programador pode optar pela utilização da numeração ou pelo nome dado ao tipo em específico.

Para simplificar a identificação de cada objeto, foi criada a propriedade “ID”. Ela é composta por uma “string” que foi implementada para ser única, diferente entre classes diferentes e entre objetos diferentes. Sendo assim, evitamos que em um mesmo “Form” de simulação, dois elementos tenham o mesmo “ID”.

A propriedade booliana “Selecionado” é responsável pela tarefa da indicação visual do estado do objeto em questão – selecionado ou não. Esta indicação se dá através da mudança de cor, tanto das linhas quanto dos instrumentos, para a cor de sistema correspondente a “item selecionado” (“Highlight”).

A propriedade “Controle” aponta para a classe (CtlInstrumento, CtlLinhaReta ou EleJanela) que implementa a interface ClsElemento. Essa foi a solução utilizada para conseguir tratamento específico a certos objetos.

O método “Eliminar” é responsável pela tarefa de excluir o elemento em questão. No momento em que é eliminado, o objeto deve antes se desconectar de outras classes que estejam se referenciando a ele. Somente assim o objeto, sem nenhuma referência, será efetivamente eliminado. No caso de controles, deve-se também pedir seu “descarregamento” (“Unload”) da janela de simulação.

Um processo de eliminação incompleto pode fazer com que a memória ocupada pelo elemento não seja liberada, fazendo assim com que a edição da simulação exija cada vez mais memória, até extrapolar o limite da máquina ou, pelo menos, reduzir o seu desempenho.

4.2.2 Classe ctlInstrumento – Controle

A classe controle ctlInstrumento é uma das classes que implementa os métodos e propriedades da classe clsElemento. Ela é responsável pela representação visual do instrumento na janela de simulação, sendo então implementada na forma de um “user control”.

A classe `ctlInstrumento` foi implementada de forma a possibilitar que um mesmo instrumento possa estar presente em várias simulações, permitindo assim a comparação de resultados entre diferentes configurações de montagem.

Além das propriedades e métodos definidos em `clsElemento`, `ctlInstrumento` implementa métodos e propriedades próprias.

A propriedade booliana “Completo”, indica o estado de um instrumento no que se refere às suas conexões com outros elementos, ou seja, indica se o instrumento tem as conexões mínimas necessárias para a execução de uma simulação. É uma verificação local com a função de garantir consistência na montagem da cadeia de medição, impedindo a realização de sua simulação seja por falta de conexão de algum terminal ou mesmo pela conexão de terminais auto-excludentes.

A propriedade “Instrumento” faz referência ao instrumento que está sendo apresentado na tela pelo controle.

Em relação aos métodos, “Snap” solicita como parâmetros a posição (horizontal, vertical) de um ponto. Com esses dados, o método é capaz de retornar o terminal livre mais próximo de tal ponto. Dessa forma, o procedimento de ligação entre elementos de simulação fica facilitado.

4.2.3 Classe `ctlLinhaReta` – Controle

A classe controle `ctlLinhaReta` implementa métodos e propriedades da classe `clsElemento`, bem como propriedades próprias.

Implementada como um “user control”, a classe `ctlLinhaReta` se apresenta visualmente na tela como um segmento de reta (parte de uma ligação entre dois elementos de simulação), com duas orientações possíveis: horizontal ou vertical.

A propriedade booliana “Horizontal” é responsável pela armazenagem desse estado.

As propriedades “X” e “Y” armazenam, respectivamente, as coordenadas do “primeiro nodo”, ou seja, do início do segmento de reta. “Xf” e “Yf” podem ser acessados para posições recíprocas do “segundo nodo”, ou fim do segmento.

Para facilitar o tratamento dado aos segmentos de reta em certos pontos do programa, foram criadas as propriedades “Inicial”, “Final” e “Fixo”. Elas

apontam, em um segmento horizontal, para “X”, “Xf” e “Y”, respectivamente. Para o caso vertical, apontam para “Y”, “Yf” e “X”. Assim, certas partes do código não necessitam tratar de ambos os casos separadamente.

O comprimento do segmento (diferença entre “Xf” e “X” em uma horizontal, ou entre “Yf” e “Y” em uma vertical) pode ser acessado pela propriedade de mesmo nome.

A propriedade “Apagada” informa para outras classes (principalmente ClsLigação) o fato de que o segmento de reta em questão (parte de uma ligação) não possui mais uma representação gráfica, faltando apenas que a ligação ao qual o mesmo pertence pare de referenciá-la, possibilitando assim sua eliminação da memória.

Outra propriedade booliana, “Movida”, tem a tarefa de avisar o fato da linha ter sido movida (pela ferramenta seleção). Somente com esta informação disponível se consegue atualizar corretamente sua representação gráfica.

“Visible” é uma propriedade característica de controles do Visual Basic e indica se o mesmo se encontra visível ou não. Essa propriedade foi disponibilizada nesta classe com o intuito de contornar problemas durante a edição das ligações.

Finalmente, a propriedade “Ligação” mantém uma referência para a ligação da qual esta linha faz parte.

4.2.4 Classe clsFerramenta

A classe abstrata clsFerramenta vem com o intuito de uniformizar o tratamento às ferramentas de edição do simulador, ou seja, as que permitem ao usuário alterar uma simulação. Essa alteração pode ser feita através da inserção de novos instrumentos (ferInstrumento), da ligação dos terminais desses instrumentos (ferFio) e da movimentação, exclusão, recorte ou colagem de elementos em geral (ferSeleção).

Como na classe clsElemento, muito dos métodos desta classe são homônimos a métodos já existentes no Visual Basic. Dentre estes métodos, estão: “Click”, “DbClick”, “Down” e “KeyDown”. Uma diferença significativa quanto a seus equivalentes originais é a presença do argumento da classe clsElemento. Esse argumento é utilizado em quase todos os métodos definidos

nesta classe, e tem por objetivo indicar à ferramenta a qual elemento a ação realizada pelo usuário se refere.

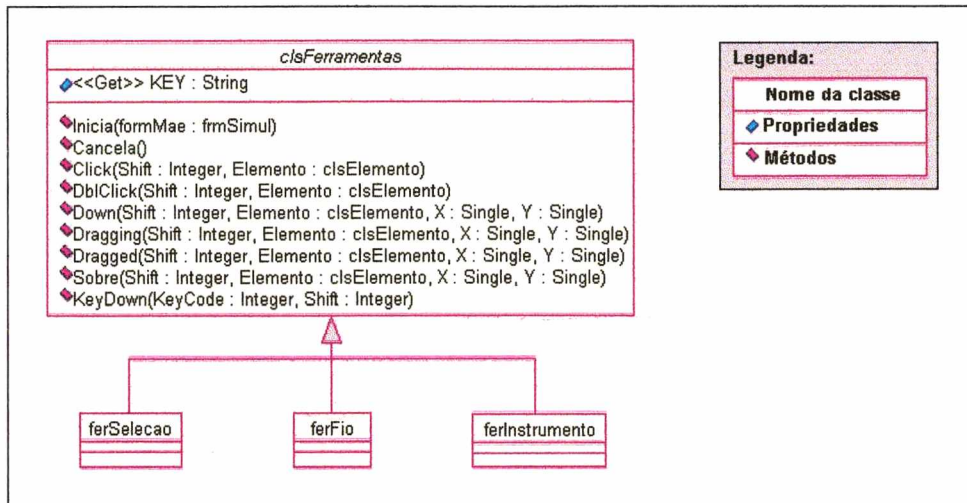


Figura 4.5. As classes clsFerramenta, ferSeleção, ferFio e ferInstrumento

Apesar de serem dois métodos diferentes, “Dragging” e “Sobre” desempenham o mesmo papel que o evento “Over” do Visual Basic: são disparados ao movimento do mouse. A diferença está no fato de que, em situação normal (botões do mouse não pressionados), é chamado o método “Sobre”. O método “Dragging” só ocorre quando o usuário “clica e arrasta” (“click-and-drag”, ou seja, move o mouse com o botão pressionado) sobre algum elemento. Sendo assim, fica mais fácil para as ferramentas tratarem o movimento do mouse, já que o próprio padrão da interface Windows é o tratamento diferenciado destes dois casos.

Quando é terminada a operação de “clique e arrasto” do mouse, o método “Dragged” é chamado, e não “Click”, de forma que a ferramenta consiga tratar esse evento com mais facilidade. Esse método é implementado para operação conjunta com “Dragging”. Normalmente, “Dragging” deve ser utilizado para previsão das alterações causadas pela ação do usuário, enquanto “Dragged” deve realmente efetivar essas alterações.

A única propriedade definida na classe clsFerramenta é “KEY”. Ela pode ser descrita da mesma forma que a propriedade “ID” da classe clsElemento. A diferença, no entanto, fica no fato de que KEY é distinta somente entre classes

que implementam as propriedades e métodos de `clsFerramenta` (`ferFio`, `ferInstrumento` e `ferSeleção`). Isto porque supõe-se que somente uma ferramenta vai estar em uso por vez em uma mesma janela de simulação.

Dado os métodos até aqui apresentados, pode-se mostrar casos onde o uso desta classe pode implicar problemas inesperados. Por exemplo, se uma ferramenta necessitar de vários cliques do mouse para terminar uma alteração válida (a ferramenta fio, onde cada clique do mouse cria um novo “nodo”). Neste caso, seria necessário um modo de avisar a ferramenta que o usuário cancelou o comando, seja pedindo sua troca, iniciando a simulação ou por outro comando ou ação qualquer que impossibilite sua continuidade. Para isso, foi implementado o método “Cancela”. Um detalhe importante de sua implementação é a necessidade de se assegurar que esse método possa ser chamado a qualquer instante, tratando de eliminar quaisquer pendências criadas pela ferramenta, como por exemplo, desenhos temporários na janela de simulação.

4.2.5 Classe `clsInstrumento`

`ClsInstrumento` é uma classe abstrata que tem por objetivo a generalização do tratamento dos instrumentos para a operação de simulação.

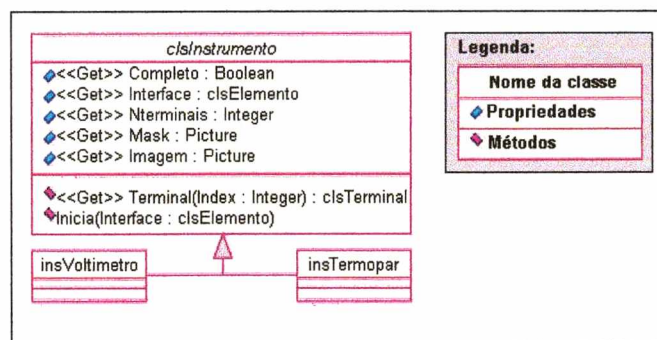


Figura 4.6. As classes `clsInstrumento`, `insVoltmetro` e `insTermopar`

As classes `insVoltmetro` e `insTermopar` implementam os métodos e propriedades definidos em `clsInstrumento`, e estão representando aqui um pequeno subconjunto dos instrumentos a serem desenvolvidos para o simulador. São essas classes que contêm os respectivos modelos matemáticos

dos instrumentos, seguindo a estruturação e metodologia apresentada no item 3.3.3.

As duas próximas classes (`clsTerminal` e `clsLigação`) a serem exploradas nos itens 4.2.6 e 4.2.7 são consideradas auxiliares da classe `clsInstrumento`. Isso levando em consideração a premissa de que a simulação em si cabe aos elementos de `clsInstrumento`, deixando para as classes `clsTerminal` e `clsLigação` a tarefa de conexão entre eles.

A classe `clsInstrumento` possui um estreito relacionamento com a classe controle `ctlInstrumento`. Isso é traduzido em forma de métodos e propriedades definidos especificamente para essa interligação. Esse é o caso das propriedades “Nterminais”, “Mask” e “Imagem”, bem como do método “Terminal”.

A propriedade “Nterminais” informa o número de terminais de conexão presentes no instrumento em questão. “Imagem” e “Mask” dizem respeito à apresentação do instrumento na tela. Ambas as propriedades foram definidas como tipo “Picture” do Visual Basic, sendo que “Imagem” contém a informação de qual figura deve ser utilizada para representar o instrumento na tela, e “Mask” é auxiliar, funcionando como uma máscara que define limites para a sensibilidade da imagem a eventos do mouse, possibilitando assim a representação de instrumentos de forma variada.

O método “Terminal” vem como um complemento à propriedade “Nterminais”, pois retorna uma matriz contendo os terminais do instrumento mediante a entrada de um índice como argumento.

A propriedade “Completo” é homônima da propriedade de mesmo nome presente na classe `ctlInstrumento`, apresentando a mesma funcionalidade.

4.2.6 Classe `clsTerminal`

A classe `clsTerminal` é responsável pelo tratamento dos terminais presentes nos instrumentos de simulação. Conforme citado em `clsInstrumento`, esta classe, juntamente com `clsLigação`, trabalham em conjunto para conferir a capacidade de conexão aos instrumentos.

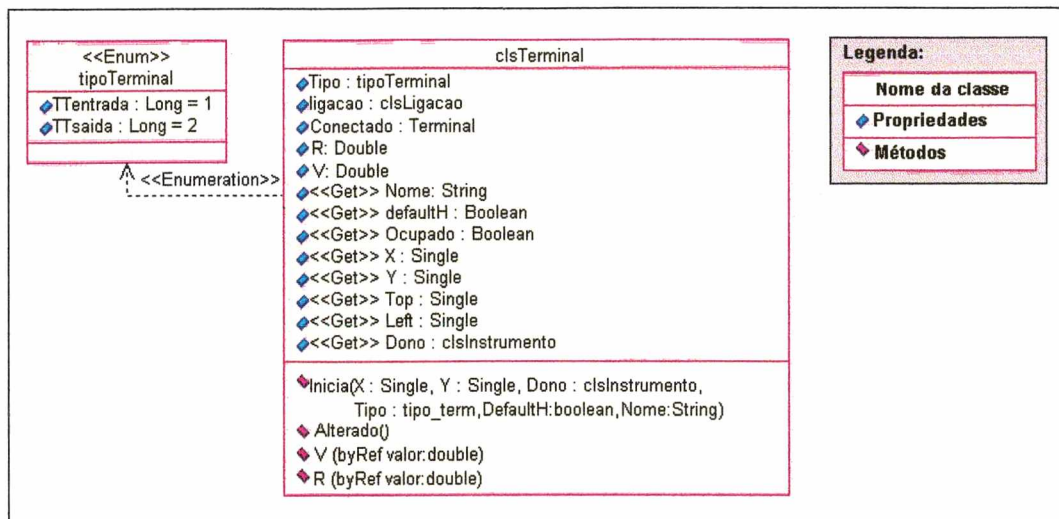


Figura 4.7. A classe clsTerminal

A classe clsTerminal possui quatro propriedades que contêm valores referentes ao posicionamento de um terminal do instrumento. Entre elas temos “X” e “Y”, que indicam a posição (horizontal e vertical, respectivamente) do terminal, com relação ao canto superior esquerdo da figura que representa o instrumento. Essa posição é utilizada por ctlInstrumento para definir onde conectar os fios e para determinar qual o terminal mais próximo do mouse para se fazer uma conexão. As propriedades “Left” e “Top” são comparáveis às homônimas do Visual Basic, com a ressalva de serem somente leitura. São calculadas através da soma da posição do canto superior esquerdo da figura que representa o instrumento com os valores de “X” e “Y”. Dessa forma, temos como localizar os terminais tanto em relação à figura fornecida por ctlInstrumento, quanto em relação à tela (“form”) no qual a simulação está sendo realizada.

A propriedade “Dono” indica qual o instrumento que detém a utilização do terminal em questão, bem como “Nome” contém a sua identificação.

“DefaultH” define o direcionamento vertical ou horizontal que um fio deve tomar quando se origina em um determinado terminal, facilitando assim o procedimento de conexão entre elementos de simulação no momento de sua edição.

“Ocupado” é uma propriedade responsável por informar para as classes ctlInstrumento e clsInstrumento que o terminal encontra-se conectado a

alguma ligação. Essa propriedade atualiza-se automaticamente quando ocorre alguma alteração na propriedade “Ligação”. Assim, o instrumento pode atualizar-se, mudando o estado de sua propriedade “Completo”.

A classe `clsTerminal` conta ainda com o evento “Alterado”, que é disparado no momento da ação de conexão ou desconexão de um terminal. Essa informação é ainda complementada pela propriedade “Conectado”, que informa qual o terminal localizado na outra ponta da ligação entre dois elementos de simulação.

A propriedade “Tipo” serve para diferenciar os terminais entre saída ou entrada de dados ou sinais.

As propriedades “R” e “V” são responsáveis pela comunicação dos instrumentos seguindo a modelagem matemática definida no item 3.3.3. Estas propriedades são utilizadas em conjunto com os eventos “R” e “V”. Tais eventos têm como parâmetro único o valor de cada uma das grandezas. Assim, um terminal do tipo “entrada de tensão” dispara o evento “R” quando o instrumento conectado a ele pedir o seu valor de resistência. Da mesma forma, ele dispara o evento “V” quando o instrumento informar a diferença de potencial aplicada. O próprio terminal deve ser capaz de evitar um pedido inválido, checando seu tipo, de forma a detectar erros em tempo de execução.

O método “Inicia” é o único método que altera os valores das propriedades “X”, “Y”, “Dono”, “DefaultH” e “Nome”. É esperado que somente o instrumento que contém o Terminal acesse esse método.

4.2.7 Classe `clsLigação`

A classe `clsLigação` é responsável pela conexão entre dois terminais e pela manutenção da representação gráfica deste fato. A essa conexão é dado o nome de ligação.

Uma ligação nada mais é que um conjunto de controles do tipo `ctlLinhaReta`, acompanhados por dois terminais da classe `clsTerminal`, um de início e outro de fim.

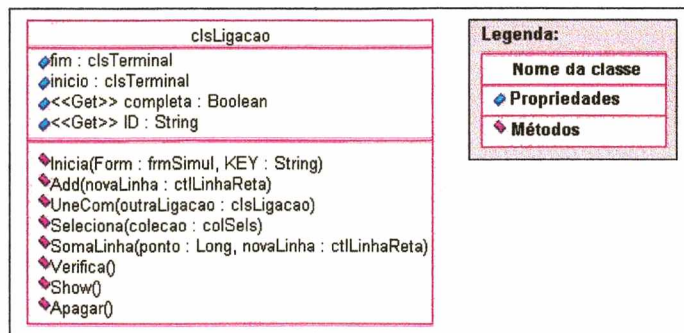


Figura 4.8. Classe clsLigação

Os métodos “Add” e “SomaLinha” tem essencialmente a mesma função: a adição de um controle `ctlLinhaReta` a um conjunto que forma a representação gráfica dessa conexão. Eles se diferenciam pelo fato de que “Add” adiciona uma linha ao final de uma lista de linhas. Por outro lado, “SomaLinha” pede como parâmetro qual o ponto onde a linha deve ser inserida. A primeira tem a vantagem de ser mais rápida e exigir menos parâmetros, porém certos momentos da edição das ligações exigem a utilização da segunda. “Add” tem utilidade principalmente no momento da criação de uma ligação nova, quando todas as linhas estarão sendo adicionadas ordenadamente a esta ligação.

Como um complemento aos métodos “Add” e “SomaLinha”, temos o método “UneCom”, utilizado para unir duas ligações ainda incompletas.

Quando ocorre um duplo clique da ferramenta de seleção sobre uma ligação, ela deve ser completamente selecionada. Para isso, foi criado o método “Selecciona”. Ele utiliza como parâmetro uma coleção de itens selecionados, e ordenadamente adiciona todos os elementos que compõem a ligação, a essa coleção.

“Verifica” é um método que tem como objetivo manter a integridade gráfica de uma ligação, bem como a certificação de sua conexão com os terminais.

As propriedades “Início” e “Fim” são as “pontas” da ligação, assim, indicam os terminais aos quais a mesma está conectada.

“Show” é um método utilizado pela ferramenta fio. Durante a criação de uma nova ligação, os “fios” que vão sendo adicionados têm sua propriedade “visível” (visible) desativada, para evitar problemas relativos quanto ao movimento do mouse sobre os mesmos. Completada a ligação, como a

ferramenta não tem mais acesso direto aos controles `ctlLinhaReta` que foram criados, para tornar os mesmos visíveis, é acessado esse método da nova ligação, que então irá torná-los visíveis um a um.

“Apagar” é o método responsável por eliminar as ligações, excluindo todas as referências a ela. Trata da desconexão dos terminais aos quais está ligada, das linhas que contêm e da eliminação de sua referência na lista de ligações da janela a qual está atrelada.

“ID” identifica isoladamente cada uma das ligações existentes em uma mesma janela de simulação. Funciona como as já mencionadas propriedades “ID” de `clsElemento` e “KEY” de `clsFerramenta`.

4.2.8 Classe `frmSimul`

Pode-se afirmar que a classe `frmSimul` “hospeda” uma simulação através do controle e armazenamento de suas características. Ela apresenta todos os métodos e propriedades necessárias para desenvolver uma simulação desde a sua criação até o seu encerramento.

A classe `frmSimul` realiza o que foi chamado de “mapeamento” dos cliques de mouse para as ferramentas de edição. Com isso conseguiu-se segregar as funções de controle e execução das tarefas de edição, deixando assim as classes dessas ferramentas somente com suas atribuições principais, facilitando o aprimoramento e o desenvolvimento de novas funções.

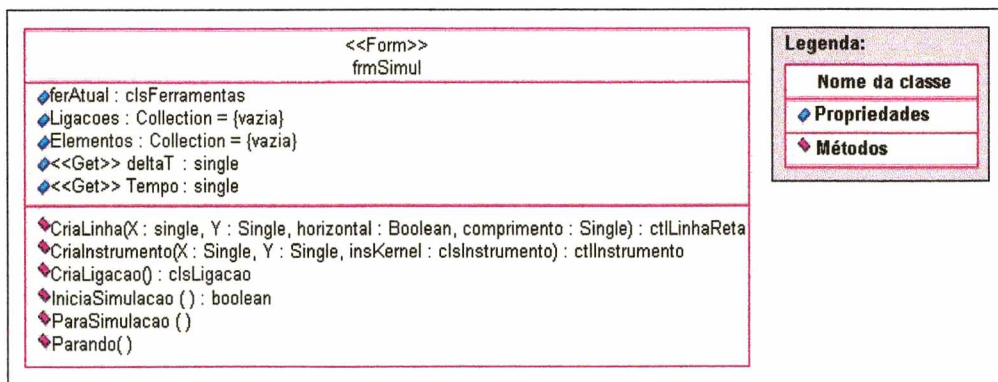


Figura 4.9. Classe `frmSimul`

As propriedades e métodos dessa classe apresentam forte ligação com as classes `clsFerramenta` e `clsElemento`. Isso pode ser constatado pela visualização dos respectivos tipos dos parâmetros associados a esses

procedimentos. Vale ressaltar que existe relacionamento direto tanto com as classes abstratas quanto com as classes hierarquicamente inferiores, refletindo o apresentado na Figura 4.3, Figura 4.4 e Figura 4.5.

As propriedades de frmSimul apresentam basicamente dados de controle da simulação em andamento. A propriedade “ferAtual” indica qual a ferramenta de edição está sendo utilizada no momento, possibilitando que os métodos de cada ferramenta sejam executados de forma correta.

A propriedade “Ligações” é do tipo “collection”, e contém todas as ligações (da classe clsLigação) presentes na simulação em andamento. Da mesma forma, “Elementos” contém uma lista de todos os elementos gráficos da classe clsElemento (instrumentos e fios) presentes na janela de simulação. Esses dados são importantes pois viabilizam o mapeamento dos cliques e o redirecionamento da aplicação para os métodos correspondentes no momento da simulação.

Existem ainda propriedades que têm a função de dar informações no que diz respeito aos tempos de simulação. A propriedade “Tempo” retorna a informação do instante atual da simulação. “DeltaT” informa o tempo entre amostras.

Alguns dos métodos da classe frmSimul estão vinculados à criação dos objetos necessários no momento da simulação. Em vista disso, temos três casos distintos de criação de elementos. O método “CriaLinha” é utilizado pela ferramenta fio (ferFio) e pela classe clsLigação quando precisam estabelecer a conexão entre dois elementos. “CriaInstrumento” é responsável pela inserção de novos instrumentos na janela de simulação através da criação de objetos do tipo ctlInstrumento e seus equivalentes da classe clsInstrumento. Finalmente temos o “CriaLigação” que, em conjunto com “CriaLinha” define a comunicação entre o par de instrumentos que foram interligados.

Os outros métodos que a classe frmSimul possui fazem o controle do início e do fim do processo de simulação. Assim, temos “IniciaSimulação” e “ParaSimulação”. Adicionalmente, foi criado um evento denominado “Parando”, que dispara uma rotina interna de reinicialização quando uma simulação foi interrompida de forma irregular, seja por problemas de erros ou inconsistência de dados.

4.3 Dificuldades enfrentadas na implementação do núcleo operacional

Neste tópico são apresentados o que foi considerado como os principais problemas enfrentados durante o processo de implementação do núcleo operacional do ambiente de simulação de instrumentos.

Notadamente, muitas dessas dificuldades estão relacionadas diretamente ao uso da linguagem de programação Visual Basic e à implementação de um sistema orientado a objetos. Isso porque tal linguagem apresenta particularidades no que tange a essa metodologia, falhando em certos aspectos conceituais e se omitindo em características imprescindíveis a uma linguagem dita orientada a objetos.

Apesar disso, neste tópico é dada ênfase às soluções utilizadas para contornar ou enfrentar tais problemas, além de explicações sobre os mesmos.

4.3.1 A linguagem Visual Basic e a orientação a objetos

Durante a implementação do sistema, foram notados alguns problemas na linguagem Visual Basic que foram considerados de importância e impacto no desenvolvimento do software. Entre eles, pode-se citar uma nítida falha no tocante à programação orientada a objetos: a ausência do princípio da herança.

O impacto da falta desse princípio causou basicamente a necessidade de transferência de tarefas entre classes e um aumento no trabalho de implementação. Um exemplo disso está na hipótese de se agrupar instrumentos com características semelhantes dentro de uma mesma classe. Isso pode parecer lógico, no entanto, a falta da herança acarreta um acréscimo no esforço de implementação de novos instrumentos, devido ao modo como a hierarquização entre classes é determinada do Visual Basic. Essa particularidade traz complicações no acesso às propriedades e métodos de classes hierarquicamente inferiores, exigindo muitas vezes o uso de variáveis auxiliares, o que dificulta a programação e o entendimento do software.

Ligações cíclicas entre classes e comunicação bidirecional também são problemas que, apesar de não serem específicos de Visual Basic, são agravados por ele. A inexistência de funções construtoras específicas para cada classe cria

a necessidade de uma aproximação destas com métodos do tipo “Inicia”. Tais métodos exigem do programador o conhecimento prévio da necessidade de se executá-los para a criação de novos objetos. Em se tratando de desenvolvimento de software, onde várias pessoas podem estar envolvidas e a rotatividade é grande, esse fato já pode ser considerado complicador.

Outro problema a ser destacado está relacionado à eliminação dos “user controls”. O processo de “descarregamento” do controle, com o objetivo de sua retirada da janela de simulação, segue um processo complicado. Isso acarreta ao programa o risco do recebimento de mensagens de erro do tipo “client side not available”, o que significa que, por exemplo, um fio não existe mais na tela, mas ainda existe de fato no programa.

Apesar desses problemas, poucas linguagens comerciais podem se dizer realmente orientadas a objeto, no sentido mais completo da metodologia. Sendo assim, seria muito difícil eliminá-los simplesmente alterando a linguagem de programação.

É de importância frisar que é típico de linguagens de programação, a solução para esses problemas, mesmo que para isso seja necessária a modificação de detalhes de projeto. A grande preocupação fica sendo, então, sobre a velocidade final do software. No entanto, dada à forma de implementação, mesmo que o processo de edição das cadeias de medição não seja muito eficiente, pode-se sempre melhorar a implementação dos instrumentos. E isto deve ser o suficiente para a melhoria do desempenho na parte crítica do programa: a simulação.

4.3.2 Objetos com relacionamento bidirecional

A maioria das classes que estão sendo implementadas neste software contém uma relação que exige uma comunicação bidirecional, ou seja, que ambos os objetos saibam da existência um do outro. Porém, o “coletor de lixo” (garbage collector) do Visual Basic somente elimina uma instância de uma classe, quando não existem mais referências a ela. Por isso, fica-se sujeito a referências cíclicas, como no caso em que a ligação referencia as linhas que a formam, e vice-versa. Assim, mesmo que se elimine do restante do programa todas as referências às linhas, impossibilitando o seu uso, ainda assim a

memória ocupada não será liberada. Para eliminar problemas dessa natureza, foram criados vários métodos de eliminação, como `Elemento.Eliminar` e `Ligação.Apagar`.

As relações de bidirecionalidade entre os objetos não são criadas instantaneamente, apesar de ser muito comum que alguns deles tenham uma relação de hierarquia ou posse, como o exemplo de terminais que pertencem a um instrumento ou ligações que estão em uma determinada janela. O fato de a janela de simulação ter criado a ligação não liga uma a outra. Sendo assim, foram criados métodos “Inicia”, que fazem um contraponto aos supracitados métodos de eliminação: criam a ligação entre o objeto criador ou “pai”, e o objeto criado ou “filho”.

4.3.3 Interfaceamento e representação gráfica

As linguagens classificadas como “Visual” apresentam em sua concepção uma certa facilidade para o desenvolvimento de interfaces gráficas. Com o Visual Basic isso não é diferente. Mesmo assim, problemas surgiram na criação da interface do simulador, especialmente na edição da janela de simulação, mais especificamente na criação de ligações e na sua representação gráfica.

As ligações são uma das partes principais da interface do sistema. Elas têm uma característica que as diferencia de todas as outras: não apresentam nenhuma similaridade com os controles normais do Visual Basic ou do Windows. Na verdade, uma ligação nem ao menos apresenta características geométricas normais, pois todos os controles são retângulos ou áreas de formatos variados que na verdade estão contidas em retângulos. Porém, este “controle”, ligação, estava mais próximo de um conjunto de controles.

Sendo assim, sua implementação se baseou na criação do controle `ctlLinhaReta`, e `Ligação` passou a ser basicamente um conjunto desses controles. Porém, isso causou problemas quanto ao posicionamento e edição das ligações, pois o Visual Basic não consegue tratar de eventos em “arrays” de classes. Sendo assim, não seria possível avisar diretamente à ligação quando ocorresse uma alteração em uma de suas linhas.

Para solucionar esse problema, foi criado o método “Verifica”. Esse método trata de situações indesejáveis e temporárias criadas pelo processo de

edição, o que se traduz basicamente na movimentação ou eliminação de uma linha. O fato é que esse tratamento tornou-se criticamente difícil, dado que o mesmo método é responsável por várias verificações:

- verificação da eliminação de uma ou mais linhas;
- verificação da movimentação de uma ou mais linhas, não necessariamente iguais;
- criação de novas ligações (separação), caso em que uma ligação foi “rompida”, possivelmente em vários pontos;
- tratamento das pontas desconectadas de uma ligação;
- movimentação das linhas adjacentes às movidas, e também possivelmente das adjacentes a estas últimas;
- criação de novas linhas para os casos de movimentações em linhas ligadas diretamente a terminais.

Apesar do relativo sucesso na obtenção desse método, ele foi de difícil implementação e apresenta um código bastante extenso, podendo parecer confuso para futuros programadores do ambiente de simulação.

4.4 Análise e avaliação das soluções implementadas

O desenvolvimento de sistemas computacionais está sempre sujeito ao aparecimento de imperfeições.

Diante desse fato, este tópico retrata as possíveis imperfeições das soluções implementadas para os problemas surgidos durante o desenvolvimento do ambiente de simulação. Cabe aqui lembrar que, durante a concepção deste trabalho, muito se fez para que futuras alterações e melhorias não incorressem em excessivo retrabalho de porções muito grandes do programa, e também que possíveis modificações não culminassem em incompatibilidades de versões de software.

A solução dada para a ligação é provavelmente a mais crítica utilizada no software. Por mais que o método “Verifica” seja melhorado, ainda assim o fato de uma ligação ser um conjunto de controles parece bastante deselegante, talvez até ineficiente. O próprio fato de que o programador deve lembrar de colocar chamadas a esse método, toda vez que efetuar alterações nas linhas, é uma falta a ser considerada, em se tratando de um projeto que pode vir a ser

desenvolvido paralelamente por vários usuários. Possivelmente, para futuras implementações do software, este seja um ponto passível de melhoria, talvez até mesmo de remodelagem completa. Esse assunto só não foi melhor desenvolvido, por sua futura alteração depender de modificações na porção efetivamente simuladora, e por ter se apresentado como um problema de implementação somente, sendo que sua modificação pode ser transparente para o usuário.

O método utilizado para enfrentar o problema das ligações cíclicas (métodos “Inicia” e métodos de eliminação) é eficaz, porém não é infalível. Qualquer que seja o motivo que faça com que todos os outros elementos deixem de se referenciar a um par de objetos que se relacionam de forma recíproca, pode fazer com que ocorra perda de memória, e possivelmente de eficiência computacional. Apesar da atenção dada para evitar tal evento, ainda assim é muito difícil garantir total certeza. Somente um “garbage collector” mais avançado no Visual Basic poderia solucionar o problema completamente. Mas a princípio, com o estado atual de implementação, pode-se esperar um funcionamento condizente com as necessidades do software.

No que diz respeito à estruturação do núcleo operacional do sistema, acredita-se que foi escolhido um conjunto abrangente de classes que representa de forma suficiente as necessidades de desenvolvimento do simulador. Essa estrutura foi criada de forma a possibilitar a expansão e a continuidade do desenvolvimento do ambiente de simulação, bem como a inserção de novos instrumentos, recursos de edição e análise de dados.

Considerando-se todos os aspectos enunciados, e avaliando o atual estado de desenvolvimento do sistema, de uma forma macro, pode-se vislumbrar o desenvolvimento do simulador com a manutenção da estrutura de classes definida até o momento. Isso indica que a composição atual se mostra adequada e voltada para a continuidade do desenvolvimento do ambiente de simulação, facilitando assim o trabalho a ser enfrentado na realização dos novos desafios de implementação que estão por vir.

Capítulo 5

Conclusões e Sugestões para Trabalhos Futuros

5.1 Conclusões

Este trabalho teve como objetivos duas frentes que fazem parte do rol de atividades relacionadas ao desenvolvimento de um ambiente de simulação de cadeias de medição. São elas:

- o projeto de um ambiente de simulação utilizando a linguagem Visual Basic, estabelecendo requisitos e definindo sua estruturação;
- a idealização e implementação de classes de objetos pertinentes à interface gráfica do simulador, com vistas à possibilidade da montagem de experimentos sem a necessidade de programação, bem como a expansão futura através da adição de novos módulos de instrumentos.

Seguindo esse escopo, no que tange ao projeto do ambiente de simulação, pôde-se constatar que os pontos mais importantes estão localizados na definição da estruturação inicial do sistema e na escolha da linguagem Visual Basic. Isso devido a este trabalho se configurar como uma base a ser seguida durante a continuidade do desenvolvimento do software.

Uma análise de necessidades das implementações futuras a serem incorporadas, indicam a adequação da estrutura vislumbrada neste trabalho, que procurou limitar as barreiras e deixar os recursos necessários para que novas idéias fossem anexadas ao projeto original.

No tocante à implementação da interface do ambiente de simulação, observou-se uma grande carga de trabalho que se localizou no desenvolvimento de estruturas que são auxiliares ao sistema de simulação. São os fios de ligação, as ferramentas de edição, entre outras. Estruturas às quais não valorizamos ou nem notamos ao utilizá-las no nosso dia-a-dia em softwares de simulação.

Durante o processo de implementação, procurou-se abreviar o desenvolvimento dessas ferramentas através da busca de bibliotecas prontas. No entanto, a consulta a desenvolvedores e uma abrangente busca na Internet não resultou em retorno algum, o que levou ao direcionamento de boa parte dos esforços para a concretização dessas tarefas.

Antes de se iniciar o processo de criação das classes pertinentes ao núcleo operacional do sistema, foi necessário um estudo preliminar para garantir a máxima reutilização do código implementado. Isto em vista da certeza da continuidade do processo de desenvolvimento do sistema e da consciência de que este trabalho é parte integrante de um projeto maior.

Assim, essa pode ser considerada como uma das maiores dificuldades deste trabalho, e também uma das grandes justificativas para a utilização da abordagem de "Orientação a objetos". Nesse sentido, além da documentação detalhada do código do software, entre as precauções tomadas de forma a suavizar a transição entre desenvolvedores, está a generalização do tratamento aos objetos através da criação de classes abstratas. Essa ação veio como uma forma de garantir maior flexibilidade na criação de itens novos para o sistema, bem como na implementação de melhorias, proporcionando uma estrutura profícua para a continuidade do desenvolvimento.

Durante a pesquisa bibliográfica, verificou-se na literatura, nos laboratórios virtuais já implantados [1][52][53][54][55][56], e nos softwares simuladores investigados, que a criação de um ambiente de simulação que contemple as características inerentes à montagem de uma cadeia de medição

possui caráter, se não de ineditismo, no mínimo inovador se comparado ao que se tem disponível no mercado.

A grande massa de informações que se encontra sobre simuladores computacionais é referida principalmente à área da eletrônica. É o caso dos renomados Pspice [36][37][38] e Electronics Workbench [40]. Encontram-se também simuladores para a área de controle de processos, como Simulink [51], Visual Simulator [50] e o próprio Labview [46][47][48], através de “add ons” específicas. E muitas outras áreas.

Entre toda essa diversidade de aplicações, o interesse maior na pesquisa desses simuladores, estava na busca de características de interface que pudessem agregar algo de diferente na implementação do sistema de simulação de cadeias de medição. No entanto, foi encontrada uma grande similaridade no modo de tratamento dos objetos na tela e na interface como um todo. Isso possibilitou, sem prejuízo algum, que o universo de pesquisa de softwares que serviriam de arcabouço de informações, fosse reduzido apenas ao PSpice e ao Electronics Workbench.

Um ponto detectado durante a pesquisa realizada e que merece ser mencionado, se refere à diversidade de informações desconexas quanto à utilização do termo “virtual”. Com o advento da INTERNET e os avanços tecnológicos nas áreas de telecomunicações e de informática, a palavra “virtual” passou a ser largamente utilizada como qualificador para uma série de aplicações.

Uma verificação nos dicionários de língua portuguesa e inglesa apresentou diversos significados para o termo “virtual”. Dentre as fontes consultadas, acredita-se este ser o mais adequado [4]: *“... algo que não existe como realidade, mas sim como potência ou faculdade. Que equivale a outro, podendo fazer as vezes deste, em virtude ou atividade ...”*.

Entretanto, essa definição não contempla a grande diversidade de utilizações para a qual o termo “virtual” está sendo empregado. A utilização da expressão como uma forma de vinculação à “World Wide Web”, causa banalização e confusão à medida que surgem “novas aplicações” diariamente. Constatação para o fato ocorre cada vez que é realizada pesquisa em uma ferramenta de busca da INTERNET. Entretanto, isso não significa que sua

utilização esteja incorreta, mas indica a necessidade de atualização de sua definição no vernáculo.

Enfim, analisando de uma forma geral, pode-se dizer que este trabalho atingiu os objetivos inicialmente propostos. Ele deixa como legado:

- a estruturação de um ambiente de simulação de cadeias de medição em Visual Basic;
- a documentação necessária para a compreensão e continuidade da implementação do sistema;
- as classes-base necessárias para a continuidade deste trabalho, definidas e implementadas.

5.2 Melhorias e desenvolvimentos a serem implementados para o ambiente de simulação

Conforme inicialmente proposto, este trabalho é parte integrante do projeto de desenvolvimento de um ambiente de simulação de cadeias de medição.

Durante o projeto desse ambiente, constatou-se a necessidade de uma série de componentes a serem desenvolvidos, de forma a dar condições para a realização de simulações. Dentro desse conjunto, foram desenvolvidas algumas classes que contemplavam os recursos de sistema previamente estabelecidos.

Existe ainda um elenco de implementações a serem executadas na continuidade do desenvolvimento do sistema. São elas:

- Implementação inicial
 - implementação da classe “Ambiente”, com o intuito de definir o meio no qual um experimento ou instrumento está imerso;
 - desenvolvimento de indicadores gráficos sofisticados e de recursos de análise de dados com o auxílio da biblioteca de componentes ActiveX “Component Works” da National Instruments;
 - desenvolvimento do módulo de salvamento e carregamento de simulações, juntamente com a criação de uma estrutura de arquivos (de instrumentos e de simulações);
 - desenvolvimento das funções de edição Cut/Copy/Paste;

- desenvolvimento do módulo de importação de instrumentos via ActiveX DLL;
- inserção de barras de ferramentas flutuantes;
- desenvolvimento da Ajuda ao usuário.
- Implementação parcial ou melhorias
 - aprimoramento do painel de entrada de dados;
 - aprimoramento da classe `clsLigação`, alterando suas linhas atualmente inseridas como controles (`ctlLinhaReta`) para linhas desenhadas na tela;
 - melhoria das rotinas e estabelecimento de novas propriedades e métodos para possibilitar a utilização de um mesmo instrumento em janelas de simulação diferentes;
 - implementação de uma extensa biblioteca de instrumentos para simulação.

Conforme pode ser verificado, a lista de atividades a serem desenvolvidas é bastante extensa. Vale ressaltar que a intenção do sistema é retratar o estado da arte em termos de instrumentação de medição, o que leva a um fluxo contínuo de criação e melhoria dos instrumentos de simulação.

Não se pode esquecer do constante aperfeiçoamento, no tocante à necessidade do desenvolvimento contínuo de interfaces gráficas, que primem pela didática e intuitividade. O surgimento de novos recursos que proporcionam essa maior interatividade deve ser incorporado ao sistema gradativamente, acompanhando as tendências apresentadas por outros simuladores do mercado e de softwares baseados na plataforma Windows.

Referências Bibliográficas

- [1] STEGAWSKI, Marcin A., SCHAUMANN, Rolf. A New Virtual-Instrumentation-Based Experimenting Environment for Undergraduate Laboratories with Application in Research and Manufacturing. **IEEE Transactions on Instrumentation and Measurement**, v. 47, n. 6, December, 1998.
- [2] PEZZOTTA, Carlos A., FLESCHE, Carlos A., NARDIN, Augusto De. Desenvolvimento de um Ambiente para Aplicação a um Simulador de Instrumentos. **Anais do II Congresso Brasileiro de Metrologia**, v. 1, p. 455-463. Dez, 2000.
- [3] EADY, Fred. A New View – Part 1: Virtual Instrumentation. **Circuit Cellar Ink (USA)**. v. 94, p. 54-59, May, 1998.
- [4] FERREIRA, Aurélio Buarque de Holanda. **Novo Aurélio – O Dicionário da Língua Portuguesa – SEC XXI**. 3ª. ed., Rio de Janeiro : Editora Nova Fronteira LTDA., 1999.
- [5] SHIMIZU, Tamio. **Simulação em Computador Digital**. São Paulo, Ed. Edgard Blücher, 1975.
- [6] HARRO, Stamm. **Simulação Industrial: Uma Avaliação de sua Utilização no Sudeste e Sul do Brasil**. Florianópolis/SC, fevereiro de 1998. 75p. Dissertação de mestrado. Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina.

- [7] POLLATSCHEK, M. A. **Introduction to Simulation**, <<http://iew3.technion.ac.il:8080/~bani/simint.html>> (30/08/2000).
- [8] APPLIED SIMULATION CONSULTANTS CORPORATION - ASC Corp. "Teaching Skills Enhancement System - TSES™." **The Power of Simulation**, <<http://www.appliedsim.com/applied/power.html>> (30/08/2000).
- [9] FISHWICK, Paul. A. Computer Simulation: The Art and Science of Digital World Construction. **IEEE Potentials**, February/March 1996, 24-27.
- [10] FISHWICK, Paul, A. Web-Based Simulation: some Personal Observations. **Proceedings of the 1996 Winter Simulation Conference - Society for Computer Simulation Int'l**. San Diego, p. 772-779, Dec. 1996.
- [11] PAGE, Ernest et al. Web-Based Simulation: Revolution or Evolution?. **ACM Transactions on Modeling and Computer Simulation**. Feb. 1999.
- [12] KRAUS JR., Werner. **Aspectos do Planejamento Curricular e da Atividade de Ensino em Engenharia de Controle e Automação**. Florianópolis/SC, 1991. 69p. Dissertação de mestrado. Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina.
- [13] GOKHALE, Anu. A. Effectiveness of computer simulation for enhancing higher order thinking. **Journal of Industrial Teacher Education**, v.33, n. 4, p. 36-46, 1996.
- [14] HELGESON, Stanley L. Microcomputers in the Science Classroom. **ERIC/SMEAC Science Education Digest**, n. 3, 1998
- [15] PORTLAND STATE UNIVERSITY. **Virtual-Instrumentation-Based Laboratory**, <<http://www.ee.pdx.edu/~eelab>> (23/11/2000).

- [16] STANFORD UNIVERSITY. **Cyberlab: A Remote Laboratory Environment For Teaching and Learning**, <<http://cyberlab.stanford.edu/About/about.html>> (28/08/2000).
- [17] HESSELINK, Lambertus, et al. Cyberlab, A New Paradigm in Distance Learning. **Instrumentation Newsletter – Technical News from National Instruments**, Third quarter, 1999.
- [18] GERHANI, N., MCGETTRICK, A. D. **Software Specifications Techniques**. Califórnia, USA: Addison Wesley Publishing Company, 1986.
- [19] FLESCH, Carlos A. **Estruturação de um sistema de simulação de Instrumentos de medição**. Florianópolis/SC, novembro de 2000. 115p. Proposta de tese de doutoramento. Programa de Pós-graduação em Engenharia Mecânica, Universidade Federal de Santa Catarina.
- [20] MEEK, B., HEATH, P. **Guide to Good Programming**. USA, Halsted Press (Willey), 1980.
- [21] PRESSMAN, Roger, S. **Engenharia de Software**. São Paulo, Brasil: Makron Books do Brasil Editora LTDA., 1995.
- [22] GONÇALVES JR., Armando A. **Metrologia Parte 1**. Florianópolis/SC. Apostila, 1998 – Departamento de Engenharia Mecânica – Laboratório de Metrologia e Automação, Universidade Federal de Santa Catarina (UFSC).
- [23] DOEBELIN, E. O. **Measurement Systems: Application and Design**. Fourth edition. USA: McGraw-Hill Publishing Company, 1990.
- [24] BLANK, Martin. **Sistematização das especificações metrológicas em sistemas automatizados de aquisição de sinais**. Florianópolis/SC, 1996. 120p. Dissertação de Mestrado. Programa de Pós-graduação em Metrologia Científica e Industrial, Universidade Federal de Santa Catarina.

- [25] FLESCHE, Carlos A., BARP, Alexandre M. Avaliação a priori da incerteza em sistemas modulares de medição de temperatura. **Anais do IV Congresso Iberoamericano de Ingeniería Mecânica - CIDIM/99**. Santiago de Chile, 23 a 26 Nov. 1999.
- [26] BLANK, Martin, FLESCHE, Carlos A. Sistematização da análise de erros na multiplexação automatizada de transdutores piezoelétricos convencionais. **Anais do XV Congresso Brasileiro de Engenharia Mecânica**. Bauru, SP, Dez. 1997.
- [27] FLESCHE, Carlos A., CAMARANO, Denise M. Proposta de metodologia de seleção dos módulos físicos e lógicos de um processo de medição automatizado de temperatura com transdutores dos tipos termopar, termorresistor e termistor. **Anais do XII Congresso Brasileiro de Engenharia Mecânica**, v.3, p. 269-274. Brasília, Dez. 1993.
- [28] FLESCHE, Carlos A., TRONCOSO, Luiz S. Sistematização da análise das fontes de erros na multiplexação automatizada de extensômetros resistivos. **Anais do V Congresso Nacional de Ingeniería Mecânica**. Chile, Dez. 1992.
- [29] MONTEIRO, Odlaniger L. D. **Programação de sistemas e a orientação a objetos**. <<http://www.odlaniger.pro.br/apostilas/artigoOO.htm>> (27/03/2001).
- [30] BOOCH, Grady. **Objected-Oriented Analysis and Design with applications**. USA: Adison Wesley, 1998.
- [31] ALHIR, Sinan, S. **UML in a Nutshell**. USA: O'Reilly & Associates, 1998.
- [32] NIKOUKARAN, Jalal, PAUL, Ray J. Software selection for simulation in manufacturing: a review. **Simulation Practice and Theory**, v. 7, n.1, p. 1-14, March 1999.

- [33] PRAEHOFER, Herbert. Object Oriented Modular Hierarchical Simulation Modeling: Towards Reuse of Simulation Code. **Simulation Practice and Theory**, v. 4, n. 4, p. 5-8, July 1996.
- [34] KIM, Y, FISHWICK, Paul A. The Design of a Human Computer Interface for a Multimodeling Object Oriented Simulation Environment. **SPIE Aerosense Conference**, Orlando, April 1998.
- [35] CUBERT, R. M et al. MOOSE: architecture of an object-oriented multimodeling simulation system. Proceedings of Enabling Technology for Simulation Science – Part of SPIE AeroSense '97 Conference. Orlando, April 1997.
- [36] MICROSIM. **MicroSim Pspice A/D & Basics – Circuit Analysis Software – User Guide (version 7.1)**. Irvine, USA: MicroSim Corporation, 1996.
- [37] MICROSIM. **MicroSim Pspice A/D & Basics – Circuit Analysis Software – Reference Manual (version 7.1)**. Irvine, USA: MicroSim Corporation, 1996.
- [38] CADENCE. **Pspice products – Analog and mixed signal simulation, optimization, simulation, analog, digital**; <<http://pcb.cadence.com/Product/Analog/ContentPate/pspicedsr9.asp>> (25/05/2001).
- [39] MANZONI, Alessandro. **Desenvolvimento de um módulo dinâmico para simuladores de ensino e treinamento em sistemas de energia elétrica usando programação orientada a objetos**. Florianópolis/SC, 1996. 215p. Dissertação de mestrado. Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina.
- [40] ELECTRONICS WORKBENCH. **Electronics Workbench – Design solutions for every desktop**, <<http://www.electronicworkbench.com>> (22/06/2001).

- [41] IEEE Std 100-1996. **The IEEE Standard Dictionary of Electrical and Electronics Terms**. Sixth edition. Institute of Electrical and Electronics Engineers, Inc., 1997.
- [42] DARRIGO, Silvia R. **Desenvolvimento de modelos do comportamento metrológico de instrumentos de medição**. Florianópolis/SC, 2001. 125p. Dissertação de Mestrado. Programa de Pós-graduação em Metrologia Científica e Industrial, Universidade Federal de Santa Catarina.
- [43] OLIVEIRA, Antônio C. X. de. **Modelagem de grandezas dinâmicas de instrumentos de medição**. Florianópolis/SC, 2001. 130p. Dissertação de Mestrado. Programa de Pós-graduação em Metrologia Científica e Industrial, Universidade Federal de Santa Catarina.
- [44] OMEGA ENGINEERING, Inc. **The Temperature Handbook**. Stanford, USA: Omega Engineering, Inc., vol. 28, 1995.
- [45] OBJECT MANAGEMENT GROUP INC. **OMG Unified Modeling Language (UML) Specification**. Volume 1.3, Junho de 1999. <<http://www.omg.org>> (20/07/2001).
- [46] NATIONAL INSTRUMENTS. **LabVIEW Reference Manual**. USA, 1999.
- [47] NATIONAL INSTRUMENTS. **Instrumentation Catalogue: 2001**.
- [48] NATIONAL INSTRUMENTS. **LabVIEW Basics Course** (Course Software Version 5.1). USA, 2000.
- [49] WHITE, Jason, CONE, Evan. **Using ComponentWorks GPIB ActiveX Controls to Accelerate Development with Visual Basic and Visual C/C++**. National Instruments – Application note 119. USA, 1998.
- [50] VISUAL SOLUTIONS INC. **Visual Simulator - VISSIM**, <<http://www.vissim.com>> (14/05/2001).

- [51] THE MATH WORKS INC. **MATLAB: User's guide for Microsoft Windows: High-Performance Numeric Computation and Visualization Software**. Englewood Cliffs, NJ: Prentice Hall, 1992.
- [52] TELEMAR UNIVERSITY COLLEGE. **SYSLAB - Dynamic Systems Virtual Lab**, <http://www.techteach.no/syslab/index_eng.htm> (21/11/2000).
- [53] SHIN, Dongil et al. Web-Based interactive virtual laboratory system for unit operations and process systems engineering education. **Computers and Chemical Engineering**, v. 24, p. 1381-1385, Dec., 2000.
- [54] STANCIL, Daniel D. "The Virtual Lab: Engineering the Future." **Carnegie Mellon's Virtual Lab: Application for the Smithsonian Computerworld Leadership Award**, <<http://www.ece.cmu.edu/~stancil/virtual-lab/application.html>> (27/11/2000).
- [55] PALOP, Jose M. G., TERUEL, Jose M. A. Virtual Work Bench for Electronic Instrumentation Teaching. **IEEE Transactions on Education**, v. 43, n. 1, p. 15-18, Feb. 2000.
- [56] BENETAZZO, Luigino et al. A Web-Based Distributed Virtual Educational Laboratory. **IEEE Transactions on Instrumentation and Measurement**, v. 49, n. 2, p. 349-356, April 2000.
- [57] MICROSOFT. **Microsoft Visual Basic 5.0 Programmer's Guide**. Microsoft Press, 1997.
- [58] GUREWICH, Nathan, GUREWICH, Ori. **Teach Yourself Visual Basic 5 in 21 Days - Professional Reference Edition**. Indianápolis, USA: Sams Publishing, 1995.
- [59] SMILEY, John. **Learn to Program Objects with Visual Basic 6**. USA: Active Path, 1999.

- [60] BIPM;IEC;IFCC;ISO;IUPAC;IUPAP;OIML. **Guia para a Expressão da Incerteza de Medição.** Segunda Edição Brasileira do “Guide to the Expression of Uncertainty in Measurement”. Rio de Janeiro, RJ: Programa RH–Metrologia, 1998.
- [61] QUATRANI, Terry. **Modelagem Visual com Rational Rose 2000 e UML.** Rio de Janeiro, Editora Ciência Moderna Ltda., 2001.