UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Uma nova abordagem para modelagem e simulação interativa visual

por

PAULO RECH WAGNER

Tese submetida à avaliação, como requisito parcial
para obtenção do título de
Doutor em Ciência da Computação

Profa. Dra. Carla Maria Dal Sasso Freitas
Orientadora

Porto Alegre, março de 1999.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Wagner, Paulo Rech
Uma nova abordagem para modelagem e simulação interativa visual / Paulo Rech Wagner.- Porto Alegre: PPGC da UFRGS, 1999.
136f.: il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 1999, Orientador: Freitas, Carla Maria Dal Sasso.

1. Simulação. 2. Modelagem. 3. Simulação Interativa Visual. 4. Modelagem Interativa Visual. I. Freitas, Carla Maria Dal Sasso. II. Título.

APLICAÇÃO DOS COMPUTADORES. SEU
SIMULAÇÃO
SIMULAÇÃO INTERATIVA VISUAL
CDU 519.1.03.03.00.6

UNIVERSIDADE FEDREAL DO RIO GRANDE DO SUL
Reitora: Profa. Wrana Panizzi
Pró-Reitor de Pós-Graduação: Prof. Franz Rainer Semmelmann
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenador do PPGC: Profa. Carla Maria Dal Sasso Freitas
Bibliotecária-Chefe do Instituto de Informática: Beatriz Haro
Bibliotecária responsável pela normalização de documentos: Ida Rossi

Para Vera Regina, Pedro e Gabriel

Agradecimentos

Este trabalho teve a contribuição de inúmeras pessoas, que de uma forma ou de outra, direta ou indiretamente, ajudaram para que ele chegasse no estágio atual. A todas estas pessoas agradeço. Mas, não posso deixar de externar um agradecimento especial àquelas pessoas que de uma forma mais direta participaram intensamente deste trabalho.

Em especial à professora e orientadora Carla Maria Dal Sasso Freitas e ao professor Flávio Rech Wagner que contribuíram de maneira efetiva no desenvolvimento e aprimoramento do trabalho. Também ajudaram quase que diariamente no meu crescimento e amadurecimento na área de informática desde que ingressei no curso de mestrado da UFRGS, mostrando muita paciência e compreensão ao longo deste período.

Ao Instituto de Informática e ao PPGC da UFRGS pelos recursos disponibilizados durante todo o doutorado. Também ao pessoal administrativo e dos laboratórios pela amizade durante todos estes anos, sempre se colocando à disposição para ajudar nas tarefas que estivessem a seu alcance.

À CAPES pelo apoio financeiro durante o período de desenvolvimento do trabalho. Também não posso deixar de destacar o apoio da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), sempre disposta a investir no aprimoramento de seu corpo docente.

Aos colegas e amigos do PPGC e da PUCRS pelas conversas e incentivos, principalmente durante o "cafezinho", que sempre me ajudaram a manter as forças nos momentos mais difíceis para conseguir concluir este trabalho. Em especial ao colega e prof. Bernardo Copstein pelo interesse constante e pelas inúmeras e valiosas contribuições ao logo do trabalho.

Aos alunos de graduação Daniel Wildt, Luis Felipe Gheller e João Jornada, sem os quais a implementação do protótipo não seria possível. Também aos alunos da PUCRS por mim orientados que fizeram seus trabalhos relacionados com a implementação do protótipo.

A toda minha família, em especial aos meus pais, que me apoiaram e incentivaram em todos os momentos e decisões tomadas ao longo deste período. À minha esposa Vera Regina pela compreensão com um doutorando quase sempre a beira de um ataque de nervos e pela dedicação aos nossos dois filhos que nasceram e cresceram durante este período no qual estive ausente por diversas vezes.

Finalmente, um agradecimento especial ao meu pai que nunca deixou de medir esforços para me ajudar e foi um verdadeiro anjo da guarda durante praticamente todo esta minha caminhada. Este trabalho também é dedicado a ele que também está se sentindo vitorioso, assim como toda minha família, mesmo não estando mais entre nós.

Sumário

Lista de Figuras 8
Lista de Tabelas 10
Resumo 11
Abstract 12

1 Introdução 13
1.1 Simulação 13
1.2 Modelos de simulação 14
1.3 Simulação de modelos discretos 16
1.4 Fases de um processo de simulação 17
1.5 Orientação a objetos na simulação 19
1.6 Simulação Interativa Visual 22
1.7 Modelagem Interativa Visual 22
1.8 Objetivo do trabalho 23
1.9 Organização do texto 24

2 Objetivos do trabalho e metodologia 25
2.1 Objetivos específicos 25
2.2 Métodos adotados 25
2.3 Aplicação exemplo 27

3 Simulação interativa visual 29
3.1 Introdução 29
3.2 Conceitos 30
3.3 O Uso da animação 31
3.4 Sistemas e linguagens de simulação com recursos gráficos 33
3.5 Análise dos ambientes de VIS 35
3.5.1 Ambientes com apresentação dos dados em pós-processamento 35
3.5.1.1 Proof 36
3.5.1.2 CINEMA 37
3.5.2 Ambientes que realizam a análise com monitoramento 38
3.5.2.1 Performance Analysis Workstation 39
3.5.2.2 Ambiente de simulação do sistema AMPLO 39
3.5.3 Ambientes que utilizam a técnica de controle 40
3.5.3.1 SEE-WHY 41
3.5.3.2 Ambiente de simulação centrado no usuário 41
3.5.3.3 VISE 42
3.6 Características e requerimentos para ambientes de VIS 43

4 Modelagem interativa visual 49
4.1 Introdução 49
4.2 Ambiente de modelagem 50

4.3 Linguagens visuais de programação **52**
4.3.1 Programação Gráfica 53
4.4 Modelagem gráfica hierárquica **54**
4.5 Detecção de erros na modelagem gráfica **56**
4.6 Ambientes gráficos de construção de modelos **57**
4.6.1 IMDE 57
4.6.2 GPVSS 59
4.6.3 ISI 60
4.6.4 SIMFLEX 60
4.6.5 SimView 62
4.7 Características para um ambiente de VIM **63**

5 Uma abordagem integradora para VIS e VIM **66**
5.1 Introdução **66**
5.2 A abordagem VISM **66**
5.3 Modelagem diagramática **68**

6 VISME -Um ambiente de simulação segundo a abordagem VISM **71**
6.1 Introdução **71**
6.2 Ferramentas de experimentação e análise **72**
6.2.1 Recursos de monitoramento 73
6.2.2 Recursos de controle 73
6.2.3 Recursos de análise 73
6.3 Ferramentas de modelagem **74**
6.3.1 Diagrama de classes 74
6.3.2 Diagrama de estados 78
6.3.3 Editor de cenário 80
6.3.4 Tradutor 83
6.3.5 Editor de ícones 83
6.4 Animação **84**
6.5 Alternativas de modelagem **85**
6.6 Recursos de controle disponíveis **89**

7 Implementação **94**
7.1 Introdução **94**
7.2 Ferramenta de edição da estrutura do modelo **95**
7.3 Ferramenta de descrição do comportamento **98**
7.4 Biblioteca de classes para simulação **100**
7.5. Ferramenta de edição do cenário **102**
7.6 Ferramenta de análise de resultados **105**
7.7 Geração do modelo executável **109**
7.5.1 Diagrama de classes 109
7.5.1.1 Estrutura de dados 109
7.5.1.2 Código gerado 111
7.5.2 Diagrama de estados 112
7.5.2.1 Estrutura de dados 112

7.5.2.2 Código gerado 113
7.5.3 Cenário 115
7.5.3.1 Estrutura de dados 115
7.5.3.2 Código gerado 116

8 Comparação com outros ambientes de simulação 117
8.1 VMSS 117
8.2 VSE 118
8.3 ARENA 118
8.4 MICRO SAINT 119
8.5 WITNESS 120
8.6 TAYLOR II 121
8.7 PROMODEL 122
8.8 SIMOBJECT 122
8.9 Análise comparativa 123

9 Conclusões 126
9.1 Resumo e contribuição do trabalho 126
9.2 Perspectivas e continuidade do trabalho 128

Bibliografia 129

Lista de Figuras

FIGURA 1.1 - Tipos de modelos. 15
FIGURA 1.2 - Fases de um processo de simulação. 18
FIGURA 2.1 - Sistema de filas com um único servidor. 25
FIGURA 2.2 - Agência bancária. 27
FIGURA 3.1 - Abordagem hierárquica para animação na simulação. 33
FIGURA 3.2 - Ambiente de simulação com a técnica de pós-processamento 36
FIGURA 3.3 - Ambiente de simulação com a técnica de monitoramento. 38
FIGURA 3.4 - Ambiente de simulação com a técnica de controle. 40
FIGURA 3.5 - Arquitetura de um ambiente de VIS 45
FIGURA 4.1 - Arquitetura de um ambiente de modelagem. 51
FIGURA 4.2 - Arquitetura de um ambiente de VIM. 64
FIGURA 5.1 - Ambiente de VISM 67
FIGURA 6.1 - Arquitetura do ambiente VISME. 71
FIGURA 6.2 - Composição do modelo em VISME. 74
FIGURA 6.3 - Representação gráfica de uma classe. 75
FIGURA 6.4 - Classes padrões dos modelos em VISME. 76
FIGURA 6.5 - Diagrama de classes da agência bancária 77
FIGURA 6.6 - Notação para diagrama de estados. 78
FIGURA 6.7 - Notações possíveis para transições. 79
FIGURA 6.8 - Diagrama de estados da classe gerente. 80
FIGURA 6.9 - Cenário de uma agência bancária 81
FIGURA 6.10 - Alternativa 1: todos clientes conhecem todos os recursos. 86
FIGURA 6.11 - Alternativa 2: clientes distintos podem conhecer recursos distintos. 87
FIGURA 6.12 - Alternativa 3: cliente específico conhece recurso específico 89
FIGURA 7.1 - Interface principal do ambiente VISME. 94
FIGURA 7.2 - Padrão de interface das ferramentas do ambiente VISME 95
FIGURA 7.3 - Ferramenta para o diagrama de classes. 96
FIGURA 7.4 - Barra de ícones da ferramenta do diagrama de classes. 96
FIGURA 7.5 - Janela de criação de classes 97
FIGURA 7.6 - Cárdápio de opções para uma classe selecionada. 98
FIGURA 7.7 - Ferramenta para o diagrama de estados. 99
FIGURA 7.8 - Barra de ícones para a ferramenta de diagrama de estados 99
FIGURA 7.9 - Janela de propriedades de uma transição. 100
FIGURA 7.10 - Ferramenta de edição do cenário 103
FIGURA 7.11 - Barra de ícones da ferramenta de edição do cenário. 103
FIGURA 7.12 - Janela de criação de instâncias. 104
FIGURA 7.13 - Janela de especificação de um recurso "storage" 105
FIGURA 7.14 - Interface da ferramenta de análise de resultados 106
FIGURA 7.15 - Janela com o arquivo de resultados 106
FIGURA 7.16 - Barra de ícones da ferramenta de análise de resultados. 107
FIGURA 7.17 - Janela do recurso filtro. 107
FIGURA 7.18 - Gráfico de barras para análise de resultados. 108
FIGURA 7.19 - Estatísticas da análise de resultados. 109
FIGURA 7.20 - Estrutura de dados para o diagrama de classes. 110

FIGURA 7.21 - Estrutura de dados para o diagrama de estados..... 113
FIGURA 7.22 - Rotinas geradas relativas a um diagrama de estados. 114
FIGURA 7.23 - Lista de instâncias do cenário. 115
FIGURA 7.24 - Lista de relacionamentos do cenário. 116

Lista de Tabelas

TABELA 8.1 - Análise comparativa. 124

Resumo

Nos últimos anos, vários esforços têm sido realizados para desenvolver ambientes de simulação mais amigáveis para seus usuários (modeladores e tomadores de decisões). Técnicas de computação gráfica têm sido largamente utilizadas na interface com o usuário, tanto para a apresentação dos resultados da simulação em pós-processamento ou durante a experimentação, como para a construção de modelos de simulação através de editores gráficos. Desta forma, vários estudos na área de simulação têm sido direcionados para ambientes que ofereçam facilidades para a modelagem e experimentação, trazendo novos conceitos como simulação interativa visual (VIS - Visual Interactive Simulation) e modelagem interativa visual (VIM - Visual Interactive Modeling).

A partir de estudos de diversos sistemas e ambientes de simulação, através de uma análise de seus recursos e facilidades disponíveis, são definidos de forma bem clara os conceitos de VIS e VIM. VIM se refere à construção do modelo de simulação com auxílio de recursos visuais e interativos na fase de modelagem, enquanto que VIS se refere à execução do modelo de simulação (usa a descrição gerada em VIM) através de recursos oferecidos na fase de experimentação. Desta forma, são estabelecidas as características desejáveis para ambientes de VIS e VIM.

Com base nestes estudos, foi identificada a ausência de um ambiente de simulação que integre os recursos de VIS e VIM, ao mesmo tempo, durante a experimentação do modelo. Deste modo, este trabalho propõe uma nova abordagem para ambientes de simulação que integra propriedades e recursos encontrados em VIS e VIM. Esta nova abordagem, chamada VISM (Visual Interactive Simulation and Modeling), tem como objetivo o controle completo de todo o processo de simulação permitindo que o usuário tenha, ao mesmo tempo, controle sobre as tarefas de experimentação e possibilidade de modificação do modelo de simulação durante ela.

Para permitir a validação desta nova abordagem, foi desenvolvido o ambiente VISME (Visual Interactive Simulation and Modeling Environment) que tem como característica principal o uso de recursos visuais tanto na construção como experimentação do modelo. O ambiente VISME possui três módulos principais, totalmente integrados através de uma interface única: um diagrama de classes hierárquico para a construção da estrutura estática do modelo de simulação; um diagrama de estados para descrever o comportamento das entidades do modelo; e, um editor gráfico específico que permite construir o cenário (apresentação visual) do modelo utilizado na experimentação. Além destes módulos que dão suporte à fase de modelagem, o ambiente VISME possui todos os recursos necessários para experimentar com o modelo. Tais recursos, caracterizados por sua natureza gráfica e interativa, estão totalmente integrados com as ferramentas de modelagem.

Palavras-chave: simulação, modelagem, simulação interativa visual, modelagem interativa visual.

TITLE: "A new approach to visual interactive modeling and simulation."

Abstract

In recent years, many efforts have been made to develop simulation environments that are more user friendly. Computer graphics techniques have been widely applied in user interface development, both for the presentation of simulation results in post-processing or during the experimentation and, for building models by means of graphic editors. On simulation several studies have aimed to develop environments that offer facilities for both modeling and experimentation, introducing concepts like Visual Interactive Simulation (VIS) and Visual Interactive Modeling (VIM).

Based on studies of many systems and simulation environments, by analysis of their resources and facilities, the concepts of VIS and VIM are clearly defined. VIM refers to the construction of a simulation model by means of visual and interactive resources in the modeling phase, while VIS refers to the use of similar resources in the experimentation phase. This work presents the characteristics expected from VIS and VIM environments.

Such study, allowed us to identify the lack of a simulation environment which integrates VIS and VIM resources at the same time during the model executation. This work proposes a new approach for simulation environments that integrates the properties and resources found in VIS and VIM. This new approach, called VISM (Visual Interactive Simulation and Modeling), aim to control the simulation process allowing the user to have at the same time control of the experimentation tasks and possibility of modifying the simulation model during the experimentation.

To allow the validation of this new approach was developed the VISME environment (Visual Interactive Simulation and Modeling Environment). The VISME environment has as mainly characteristic the use of visual resources for construction and experimentation of the simulation model. The environment has three modules, fully integrated by a single interface: a hierarchical class diagram to construct the static structure of the simulation model; a state diagram to describe the behavior of the model entities; and, a specific graphical editor used to built the scenario of the model used in the experimentation. Besides these three modules which have support to the modeling phase, the VISME environment has all the resources to experiment with the model. These resources, of the visual and interactive nature, are totally integrated with the modeling tools.

Key words: simulation, modeling, visual interactive simulation, visual interactive modeling.

1 Introdução

1.1 Simulação

O planejamento e a análise de diversas estratégias e procedimentos são essenciais em qualquer área da atividade humana. Para tal, técnicas de simulação são utilizadas na resolução de problemas e na ajuda a tomadas de decisão. Simulação, segundo Shannon [SHA75], é o processo de projetar um modelo de um sistema real ou imaginário e conduzir experimentos com este modelo com o propósito de compreender ou prever o comportamento do sistema e avaliar várias estratégias para a sua operação. Na realidade, simulação é o uso de um modelo para "imitar" um sistema real e, através dele, obter um entendimento melhor do sistema sob uma variedade de circunstâncias [THE90]. As simulações são geralmente utilizadas para determinar como alguns aspectos do sistema deveriam ser gerenciados, de acordo com determinadas situações a ele impostas.

Sendo uma metodologia utilizada para a análise de sistemas reais, uma das questões a respeito da simulação é verificar quando o seu emprego é útil. Os principais motivos para se utilizar a simulação são [GOR69, SHA75, KEL98]:

a) no projeto de sistemas ainda não existentes quando a construção de um protótipo tem custo muito elevado tornando-se praticamente proibitivo (por exemplo, a construção de circuitos integrados);

b) quando a experimentação com o sistema real é impossível, seja pela falta de acesso direto ao sistema para observação (por exemplo, física nuclear) ou pela impossibilidade de variação nos parâmetros do sistema (por exemplo, sistemas biológicos);

c) quando a experimentação com o sistema real é indesejável pois pode afetar o desempenho e o comportamento do mesmo (por exemplo, ecossistema);

d) para a compressão ou expansão da escala de tempo, podendo-se fazer observações em períodos muito grandes ou pequenos (por exemplo, fenômenos econômicos, demográficos e fenômenos nucleares);

e) para treinamento e instrução, podendo-se através da simulação se obter uma melhor compreensão e operação do sistema (por exemplo, simuladores de vôo).

Uma das áreas de aplicações onde mais cresce o uso de simulação é a de manufatura; atualmente quase todos os projetos se beneficiam de algum tipo de análise via simulação. Outros exemplos de aplicações da simulação são [MAR68]: estudo de problemas em projetos de sistemas eletrônicos e circuitos; estudo e análise de problemas urbanos, como tráfego; análise e avaliação de sistemas elétricos e mecânicos que estão em fase de desenvolvimento; análise dos efeitos de mudanças metereológicas e fenômenos atmosféricos; estudo de problemas em biologia, como mutações e hereditariedade; estudo de problemas educacionais, como administrativos e horários escolares; estudo dos efeitos de falhas em redes de computadores; estudo do espaço aéreo, traçado de planos de vôos de acordo com as condições metereológias e rotas de outras aeronaves; estudos sobre as consequências do crescimento populacional em uma região; teste de estratégias e táticas na política,

esportes, negócios e relações humanas; estudo das consequências da ação humana no ecossistema, como desmatamento e queimadas; estudo de problemas que envolvam atendimento de clientes, como bancos e supermercados; projeto de jogos de computadores.

1.2 Modelos de simulação

Para se estudar um sistema real de forma a compreendê-lo detalhadamente ou prever como ele se comportará em diversas circunstâncias, poderiam ser construídos protótipos para este sistema, mas isto pode apresentar uma relação custo/benefício muito alta. Conseqüentemente, os estudos sobre os sistemas são geralmente conduzidos através de ***modelos*** do sistema que agregam os dados significativos a respeito do mesmo. Gordon [GOR69] define modelo como um corpo de informações sobre um sistema coletado com o propósito de entender este sistema. Um modelo deve ser definido como uma abstração de um sistema real, com uma estrutura mais simples, onde a própria construção do modelo auxilia a organizar, avaliar e examinar a validade de idéias, ou seja, ajuda na compreensão do sistema. O modelo deve ser uma aproximação bem razoável do sistema real e conter a maior parte dos aspectos importantes do mesmo. Por outro lado, o modelo não deve ser tão complexo que se torne impossível compreendê-lo e manipulá-lo. Dados, parâmetros e relações devem ter representação adequada ao problema sob investigação.

A utilização dos modelos é muito ampla e se dá em todos os ramos da ciência [EMS70]. Os modelos ajudam na comunicação evitando a ambigüidade de entendimento por parte das pessoas quando da observação de sistemas complexos. Também na área de instrução e treinamento os modelos são muito úteis, reduzindo os riscos e custos. O conhecimento prévio sobre certas situações que podem acontecer na realidade é muito importante. Isto pode ser conseguido através do estudo do modelo do sistema. Talvez o mais importante uso dos modelos seja na previsão do comportamento do sistema, quer para o projeto, para a análise ou diante de situações anormais de funcionamento, de modo que se possa determinar, a partir de uma ou mais mudanças em determinados aspectos do sistema, os reflexos que poderão surgir em outros aspectos do sistema ou no próprio sistema como um todo. Além destas funções, os modelos devem apresentar outras características próprias, relativas a custos de desenvolvimento, utilização e obtenção de resultados. Tais características podem ser identificadas como critérios para um bom modelo: simplicidade de compreensão pelo usuário; facilidade de controle, manipulação e comunicação dos resultados; facilidade de modificação e atualização; completo, no nível de abrangência requerida e consistente, no sentido de não fornecer respostas absurdas.

Os modelos podem ser classificados de diferentes formas, conforme apresentado na figura 1.1:

FIGURA 1.1 - Tipos de modelos.

Os modelos físicos são modelos que são representados por atributos físicos semelhantes ao sistema em estudo. Os modelos físicos em escala natural, também chamados de protótipos, são aqueles que tem uma equivalência um-para-um com o sistema real. Estes modelos são utilizados para minimizar riscos e incertezas e tem grande capacidade de fornecer informações com precisão. Em contrapartida, a sua grande desvantagem é o grande custo para a sua construção. Um exemplo típico deste tipo de modelo são os simuladores de vôo. Já os modelos físicos em escala reduzida são assim denominados pois suas dimensões são menores que o sistema real devido à impossibilidade de construção de um protótipo. Como exemplo deste tipo de modelo podem-se citar as barragens e túneis de vento.

Os modelos matemáticos se caracterizam pelo uso de símbolos algébricos para representar os componentes do sistema e suas relações. São modelos abstratos onde aproximações matemáticas são buscadas para os atributos físicos. Suas vantagens são a flexibilidade, pouco tempo e custo de desenvolvimento e resposta mais rápida. Os modelos analíticos resultam em expressões matemáticas bem definidas e sua solução é obtida através da resolução da expressão. No modelo numérico não é necessário o conhecimento da equação matemática que rege o sistema, sua solução é obtida por iterações onde os valores das variáveis são fornecidos. Um exemplo de um modelo matemático é o estudo da economia de um país, onde através de um conjunto de fórmulas matemáticas pode-se calcular a renda nacional, tendo-se como variáveis o consumo, os investimentos, impostos e gastos do governo.

Nos modelos de simulação, a obtenção dos resultados se dá através da execução do modelo. O sistema é representado pela construção de um modelo computacional, de modo que a técnica de simulação possa ser realizada no computador para a compreensão do sistema. Conforme a variação do estado do sistema, o sistema pode ser classificado como contínuo ou discreto, sendo representado por modelos contínuos ou modelos discretos. A simulação contínua modela sistemas onde as variáveis tem valores que variam continuamente ao longo do tempo (existe um conjunto de equações que descreve o sistema, onde os valores das variáveis são fornecidos em todos os instantes de tempo). Exemplos de modelos contínuos são as reações químicas, circuitos eletrônicos, crescimento populacional, etc. Já na simulação discreta as variáveis tem seus valores alterados apenas em certos

instantes de tempo. A mudança de estado se dá pela ocorrência de eventos que indicam o início ou fim de uma atividade. O comportamento é dado por um conjunto de regras que determinam o tempo do próximo evento e as alterações nos valores das variáveis. Como exemplo de modelos discretos pode-se citar: controle de tráfego, controle de produção de fábricas, sistema telefônico, bancos, etc. Ainda tratando de modelos discretos, eles podem ser divididos em estocásticos e determinísticos. Os modelos estocásticos, também chamados de probabilísticos, tem associadas probabilidades condicionadas às funções de distribuição e variáveis randômicas. Os modelos determinísticos apresentam valores bem determinados para eventos específicos. Os modelos estocásticos são mais largamente utilizados do que os determinísticos.

1.3 Simulação de modelos discretos

A simulação de modelos discretos (simulação discreta, simulação de eventos discretos) é utilizada em sistemas em que a mudança de estado se dá de forma descontínua em tempos bem determinados. Um sistema é um conjunto de objetos reunidos segundo uma interdependência ou interação regular. Qualquer objeto de interesse no sistema é chamado de ***entidade*** e possui propriedades e características próprias (***atributos***). As entidades são classificadas em permanentes e temporárias. As ***entidades permanentes*** estão presentes durante toda a simulação, enquanto que as ***entidades temporárias*** são conhecidas porém aparecem e desaparecem durante a simulação. Os valores dos atributos de todas as entidades existentes em um determinado instante de tempo constituem o ***estado*** do sistema. Na simulação discreta, o estado do sistema só pode alterar nos tempos de ***eventos***. Um ponto importante da simulação discreta é a técnica de avanço de ***tempo***. O mecanismo mais utilizado na maioria das linguagens de simulação é o avanço de tempo por eventos, onde a mudança de tempo de simulação ocorre somente nas unidades de tempo em que ocorrem eventos, ou seja, o tempo de simulação varia aos saltos. Isto significa que entre um evento e outro não ocorre nenhuma mudança no estado do sistema. Qualquer processo que cause uma alteração no estado do sistema é definido como uma ***atividade***; atividades são realizadas quando ocorrem eventos. Os eventos representam qualquer alteração no estado do sistema e são de duas espécies: endógenos e exógenos. Os eventos endógenos são os gerados internamente pelo próprio sistema e os eventos exógenos são os impostos ao sistema pelo mundo exterior. Outro componente muito importante em um modelo discreto são as ***filas***. Quando uma determinada entidade servidora não consegue atender a demanda imposta pelas entidades clientes, formam-se as filas de espera por atendimento que são constituídas pelas entidades clientes.

Como exemplo de um modelo discreto pode-se considerar uma caixa bancária 24 horas. Este modelo é composto pela caixa 24 horas e pelas pessoas que irão realizar determinada operação bancária. As pessoas chegam à caixa de acordo com uma freqüência que varia segundo uma distribuição de probabilidade. Caso a caixa já esteja ocupada por uma pessoa, as demais pessoas que chegarem entram em uma fila de espera. O tempo de permanência de uma pessoa dentro do caixa 24 horas também é regido por uma distribuição de probabilidade. Quando uma pessoa termina

sua operação no caixa, ela abandona o sistema e a primeira que estiver na fila é atendida. Neste exemplo podem ser observadas dois tipos de entidades: a caixa 24 horas e as pessoas. A primeira é dita entidade permanente pois é uma entidade que existe enquanto durar a simulação; a segunda é uma entidade temporária, uma vez que ela é criada e destruída durante a simulação. Também pode-se identificar a formação de uma fila e a presença de eventos, tais como a chegada e a saída de uma pessoa no sistema.

Outra característica a ser analisada na simulação discreta é a abordagem empregada para descrever o sistema e as mudanças de estado [PID92]. São três as abordagens: orientada a evento, orientada a processo e orientada a atividade. Na ***simulação orientada a eventos***, o sistema é modelado pela definição das mudanças que ocorrem no tempo de evento. Deve-se determinar os eventos que podem causar uma mudança no estado do sistema e, então, desenvolver a lógica associada a cada tipo de evento. As linguagens de simulação já possuem algoritmos específicos de simulação para escalonamento de eventos. Algumas linguagens que utilizam esta abordagem são GASP, Simscript e SLAM II. Na ***simulação orientada a processos***, o sistema é modelado por uma seqüência de ações nas quais se envolve uma entidade. Na realidade, cada ação é uma seqüência de eventos que ocorrem em padrões definidos, de modo que esta seqüência de eventos pode ser generalizada e definida como uma única afirmação. GPSS, Simula e SLAM II são alguns exemplos de linguagens orientadas a processo. Finalmente, na ***simulação orientada a atividades***, são descritas as atividades nas quais as entidades do sistema estão engajadas e as condições que causam o seu início e o seu fim. Para cada avanço do tempo de simulação, todas as atividades são examinadas e suas condições são testadas. São executadas aquelas cujas condições forem verdadeiras. Poucas linguagens de simulação utilizam esta abordagem devido a esta necessidade de escalonar todas as atividades a cada avanço do tempo. Uma linguagem que adota esta abordagem é SLAM II.

1.4 Fases de um processo de simulação

Podem ser identificados vários passos quando da aplicação da simulação para o estudo e investigação de um sistema real [SHA75, BAL90, LAW91, SAD91, KEL98]. Embora existam várias descrições de quais seriam os passos de um processo de simulação, os principais passos são descritos a seguir e podem ser visualizados conforme a figura 1.2.

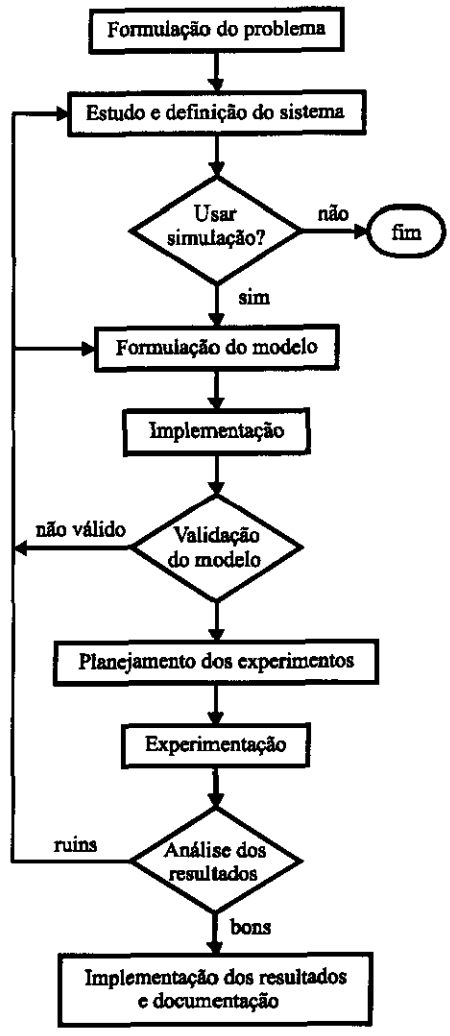


FIGURA 1.2 - Fases de um processo de simulação.

Normalmente o processo de simulação é descrito da seguinte forma. Existe um problema real que deve ser transformado em um problema bem definido, claro e bem organizado capaz de estabelecer direções para os estudos. O primeiro passo seria uma definição cuidadosa do problema, onde poderia ser determinado o nível de detalhe requerido no modelo. Dentro desse mesmo contexto, também poderiam ser descritas as possíveis técnicas que podem ser utilizadas para a solução do problema. Deve ser realizada um relação custo/benefício das diversas alternativas verificando se deve-se usar ou não a simulação. Como continuidade do processo, foi assumido que a simulação é a técnica mais apropriada para a solução do problema proposto. O próximo passo consiste na construção do modelo conceitual que representa o sistema em estudo. Neste momento deve-se reduzir a abstração do sistema real decidindo quais aspectos são realmente significantes. Dentro desta etapa também deve ser realizada a coleta e análise dos dados de entrada para provar a correção dos parâmetros do modelo e das distribuições de probabilidade de entrada. A seguir vem a fase de implementação que consiste na translação (codificação) do modelo conceitual em um modelo computacional. Esta codificação deve ser realizada

através da escolha de uma linguagem acessível ao ambiente disponível: uma linguagem de programação de propósitos gerais (FORTRAN, C, PASCAL, etc) ou uma linguagem de simulação (GPSS, SIMAN, Simscript, etc). Também é nesta fase que é realizada a verificação do modelo, ou seja, verificar se o programa realiza o que se espera do modelo. Diferentes formas de como verificar um modelo são relatadas em [BAL90, SAR91, HIL96].

Após a implementação do modelo é feita a validação do mesmo, ou seja, garantir que o modelo, dentro de seu domínio de aplicabilidade, se comporta com exatidão satisfatória, consistente com a aplicação pretendida do modelo [SAR91]. Validar propicia uma maior confiabilidade de que o modelo se comporta como o sistema real. As várias técnicas para a validação de um modelo são abordadas em [BAL90, LAW91, SAR91, HIL96]. Depois de validar o modelo, deve ser realizado o planejamento dos experimentos através da elaboração de um plano para obter a informação desejada com um custo mínimo e habilitar o usuário a realizar inferências com o modelo. Deve ser projetado um conjunto de experimentos para alcançar os objetivos desejados, onde o custo e o tempo de simulação são restrições que devem ser consideradas. A seguir, vem a experimentação que é a simulação propriamente dita com a finalidade de gerar os dados desejáveis para uma análise posterior. Através da análise da sensibilidade do modelo (sensibilidade dos resultados devido a uma variação dos valores fornecidos aos parâmetros), a observação dos resultados da simulação pode acarretar em novos trabalhos no processo de simulação, tal como a reformulação do próprio modelo de simulação. Terminada a experimentação, é hora de avaliar o desempenho do sistema simulado interpretando os resultados obtidos durante a simulação. Geralmente são empregadas a análise visual e estatística dos resultados. Pode-se concluir que os resultados apresentados não foram satisfatórios e que deve-se realizar novos experimentos. Finalmente, chega-se ao momento de colocar em uso o projeto de simulação, pois nenhum projeto de simulação pode ser considerado completo enquanto ele não for aceito, entendido e usado. Também uma documentação completa de cada uma das fases do processo é essencial, uma vez que um sistema não é estático, acarretando atualizações no modelo de simulação sempre que ocorrem mudanças no sistema.

1.5 Orientação a objetos na simulação

A necessidade de criar software de qualidade mais confiável, muito mais rapidamente, e a um custo mais baixo é um fator decisivo para a indústria de software. Devido ao aumento crescente da potência dos computadores, a evolução e a modificação do software deve ser feita rapidamente para que ele não fique defasado em relação aos demais. O ponto chave no projeto de um software é a obtenção de componentes que sejam reusáveis e flexíveis. Diversos autores [MEY88, BOO94, HIL96, RUM97] concordam que a abordagem modular é a solução para minimizar estes problemas e que a arquitetura do software deve ser estruturada a partir de objetos. O termo "orientação a objetos" superficialmente significa a organização de software como uma coleção de objetos discretos que incorporam tanto a estrutura de dados como o comportamento [RUM97]. A orientação a objetos está baseada em uma série de conceitos que serão vistos a seguir. O projeto orientado a objetos é um modo

de usar estes conceitos para estruturar e construir sistemas. Um modelo de simulação pode ser visto como um conjunto de entidades que interagem entre si. Desta forma, o paradigma de orientação a objetos permite que se faça uma analogia entre a produção de um modelo de simulação e o desenvolvimento de um software orientado a objetos, uma vez que pode-se fazer uma associação entre entidades e objetos.

Um **objeto** é qualquer coisa, real ou abstrata, que faz sentido no contexto de uma aplicação. Objetos são entidades que agrupam dados e um conjunto de operações que manipulam estes dados. Todo objeto possui uma identidade única, que o permite ser distingüido dos demais objetos, um estado e um comportamento. O **estado** de um objeto engloba todas as propriedades do objeto mais os valores de cada uma destas propriedades. O **comportamento**, como definido por Booch [BOO94], é o modo como um objeto age e reage, em termos de suas mudanças de estado e passagem de mensagens.

Uma **classe** é a descrição de um "template" que especifica as propriedades e o comportamento para um conjunto de objetos similares. Cada classe descreve um conjunto infinito de objetos e cada objeto é dito ser uma **instância** de uma classe. A estrutura de dados e o comportamento dos objetos são descritos na classe, respectivamente por seus atributos e operações. Os **atributos** armazenam o estado abstrato de cada objeto. Cada atributo tem um valor em cada instância de objeto. As **operações** são o único meio para acessar, manipular, e modificar os atributos de um objeto. Todos os objetos de uma mesma classe compartilham as mesmas operações. Um **método** é a implementação específica de uma operação em uma classe. Um objeto comunica-se com outro através de **mensagens** que identificam operações a serem realizadas no segundo objeto. O objeto responde a uma mensagem possivelmente mudando seus atributos ou retornando um resultado.

Durante a abordagem de orientação a objetos alguns aspectos devem ser levados em consideração [BOO94, HIL96]. Sem eles o modelo não pode ser considerado orientado a objetos (modelo de objetos).

A **abstração** é um princípio que consiste em ignorar os aspectos que não são significativos para os objetivos permitindo que se concentre completamente nos aspectos mais importantes. A abstração se concentra na visão externa do objeto separando o comportamento de sua implementação. Isto significa identificar o que um objeto é e o que ele faz sem decidir antecipadamente como será implementado. A abstração provê a definição da **interface** de uma classe, onde são definidos os atributos e métodos disponíveis para as outras classes.

O **encapsulamento** é definido como uma técnica para minimizar interdependências entre módulos pela definição de uma interface externa estrita. Encapsulamento é o processo de ocultar os detalhes de um objeto que não contribuem para o entendimento de suas características essenciais. Enquanto que a abstração se preocupa com o comportamento observável do objeto, o encapsulamento provê a implementação do objeto. Desta forma, a implementação de um objeto pode ser alterada sem que sejam afetadas as aplicações que o utilizam.

O particionamento de um sistema em componentes individuais (módulos) reduzindo sua complexidade é característica da **modularidade**. Cada módulo deve armazenar classes e objetos relacionados. Esta propriedade de decompor o sistema em um conjunto de módulos deve seguir as diretrizes do tradicional método do projeto estruturado de modularização de sistemas, onde os módulos devem ser coesos e fracamente acoplados. Isto permite que se obtenha um bom nível de reusabilidade e extensibilidade do sistema.

A classificação ou ordenação de abstrações resulta em uma **hierarquia**. As duas hierarquias mais importantes em um sistema são as hierarquias de herança (estrutura de classes que descrevem relações do tipo "é um tipo de") e as hierarquias de agregação (estrutura de objetos que descrevem relações como "é parte de"). A identificação destas hierarquias em um projeto auxilia no entendimento do problema.

Herança é um mecanismo para derivar novas classes a partir de classes existentes através de um processo de refinamento. Uma classe derivada (subclasse) herda todas as propriedades (representação de dados e operações) da sua classe pai (superclasse), mas pode seletivamente adicionar novas operações, estender a representação de dados ou redefinir a implementação de operações já existentes. Uma vez que as características de uma superclasse não precisam ser repetidas nas subclasses, a repetição de projeto e implementação pode ser reduzida. A reutilização provida pelo mecanismo de herança é uma das principais vantagens do paradigma de orientação a objetos. **Agregação** define o relacionamento entre uma classe agregada que é composta por diversas classes componentes. As classes componentes são parte da classe agregada.

Outro aspecto importante a ser ressaltado é a distinção entre tipo e classe. **Tipo** é a coleção de todos os objetos do sistema que respondem do mesmo modo ao mesmo conjunto de mensagens. Em outras palavras, tipo é uma coleção de objetos com a mesma interface pública. Tipo é a imposição de uma classe a um objeto, tal que objetos de diferentes tipos não podem ser intercambiados. Desta forma o uso de tipos é motivado pela necessidade de verificação de tipos.

Apesar da programação baseada em objetos ter sido inicialmente desenvolvida em SIMULA [BIR73] uma linguagem de programação com facilidades de simulação, apenas recentemente o paradigma de orientação a objetos chegou a linguagens comerciais mais difundidas, como MODSIM [BEL90] e SIM++ [LOM91]. SIMULA é uma linguagem de programação que possui todos os recursos de uma linguagem de programação de propósitos gerais acrescida de conceitos que a tornam uma linguagem voltada para a implementação de modelos de simulação. SIMULA introduziu alguns conceitos, originalmente projetados para serem utilizados em simulação, que a classificam como precursora das linguagens orientadas a objetos.

Bischack [BIS91] apresenta um histórico das linguagens de simulação que demonstra sua evolução até o surgimento das linguagens de simulação orientadas a objetos. As primeiras linguagens de simulação tais como GASP e Simscript focalizavam o controle do fluxo através da geração de eventos de simulação. A geração seguinte de linguagens de simulação, tais como GPSS, SLAM e SIMAN

enfatizam o fluxo e o processamento de entidades através de uma rede (diagramas de blocos). Já a idéia básica do paradigma de orientação a objetos é a modularização das tarefas de programação tendo como base os objetos físicos ou abstratos do sistema. A estrutura e os métodos associados com um objeto são encapsulados dentro do objeto de modo que a única maneira de acessar um método é através do envio de mensagens apropriadas para este objeto. Uma grande variedade de linguagens de programação orientadas a objetos tem sido utilizada para o desenvolvimento de ambientes de simulação, tais como: C++, MODSIM II, SIMULA e Smalltalk. Freitas [FRE91] apresenta um estudo sobre a aplicação de conceitos de orientação a objetos a linguagens e sistemas de simulação.

1.6 Simulação Interativa Visual

A simulação ao longo dos anos começou a ser utilizada nas mais diferentes áreas de aplicações, ocasionando um crescimento na complexidade das simulações bem como no volume de dados gerados para uma posterior análise. Desta forma, os estudos na área de simulação estão mais voltados para o desenvolvimentos de sistemas que ofereçam ao usuário facilidades para a simulação e modelagem. Recursos de computação gráfica passaram a ser parte integrante de todo o processo de simulação, desde a sua especificação até a implementação. Além da necessidade de uma melhor visualização dos dados, é muito importante que os ambientes de simulação ofereçam recursos para a interação do usuário com os experimentos.

Simulação interativa visual (VIS - *Visual Interactive Simulation*) é uma abordagem que requer a participação do usuário durante a fase de experimentação do processo de simulação. VIS caracteriza-se pelo controle da simulação [BEL87], ou seja, há a visualização de dados intermediários e capacidade de interação com o modelo durante o transcorrer da simulação, não simplesmente a apresentação gráfica dos resultados da simulação. Diversos aspectos do modelo podem ser alterados durante a execução da simulação fazendo com que as consequências no processo de simulação sejam imediatas. A animação também é uma característica essencial para sistemas de VIS, pois auxilia na depuração, interpretação e apresentação de resultados [JOH88].

O propósito principal de ambientes de VIS é prover ferramentas para um melhor entendimento do comportamento do sistema que se está simulando. A partir de uma análise de vários sistemas e requisitos propostos [VUJ90, ROO91], algumas características podem ser esperadas de ambientes de VIS: visualização gráfica do modelo de simulação; interação do usuário com a representação gráfica do modelo; interação do usuário durante a simulação; armazenagem de informação para uma futura análise e interação do usuário com estas informações após a simulação.

1.7 Modelagem Interativa Visual

Com a crescente utilização de ambientes de VIS, o uso de recursos gráficos interativos começaram a ser propagados com naturalidade para todas as fases

do processo de simulação. Como a análise de sistemas através de simulação está baseada na modelagem do sistema a ser simulado, o processo de modelagem do sistema começou a fazer uso destas características gráficas interativas.

Modelagem interativa visual (VIM - *Visual Interactive Modeling*) se refere à construção do modelo de simulação com o auxílio de um ambiente gráfico-interativo. Atualmente diversos ambientes de modelagem tem sido desenvolvidos, sendo constituídos por inúmeras ferramentas que auxiliam no desenvolvimento de modelos de simulação [BAL92, PAU94]. O principal objetivo destes ambientes é aumentar a produtividade do modelador, com ênfase na construção do modelo de simulação graficamente. Em muitos sistemas atuais, o modelo de simulação é construído através de uma linguagem visual, geralmente provida por editores gráficos diagramáticos. Por linguagem visual, entende-se toda a linguagem de programação baseada em recursos visuais [CHA87, MYE90].

Um ambiente de modelagem deve incluir pelo menos três diferentes ferramentas para suportar o desenvolvimento de modelos de simulação: um gerador de modelos, que permite a visualização e a construção gráfica do modelo; um analisador de modelos, que permite verificar a correção do modelo; e, um tradutor de modelos, que transforma o modelo especificado em uma representação simulável. Após esta representação simulável ter sido gerada, um ambiente de VIS poderia ser utilizado para a execução dos experimentos planejados. O processo de VIM, que dentro de um ambiente de modelagem está relacionado com o gerador de modelos, permite que com o auxílio de uma ferramenta visual interativa, o usuário desenvolva o modelo de simulação através da descrição de sua estrutura (estática e dinâmica) e de seu comportamento.

1.8 Objetivo do trabalho

O uso dos termos VIS e VIM, até certo ponto indiscriminado, tem causado uma certa confusão quanto à classificação de ambientes de simulação. Desta forma, a partir de uma análise e classificação dos ambientes de simulação, considerando os aspectos de modelagem e experimentação, tornaram-se mais claros os conceitos de VIS e VIM. Com base nesta classificação, o presente trabalho propõe uma nova abordagem para ambientes de simulação, chamada de VISM (Visual Interactive Simulation and Modeling), que integre os conceitos de VIS e VIM. O objetivo principal desta abordagem é o controle completo de todo o processo de simulação através de um ambiente integrado que ofereça as características e recursos de VIS e VIM.

A idéia desta nova abordagem vem da constatação da não existência de um ambiente integrado que possua recursos que permitam, ao mesmo tempo, a interação do usuário tanto com o modelo como com a experimentação. O ambiente *VISME* (*V*isual *I*nteractive *S*imulation and *M*odeling *E*nvironment) é um ambiente completo de modelagem e simulação baseado na abordagem VISM, onde o usuário tem à sua disposição simultaneamente ferramentas que oferecem tanto recursos para a experimentação quanto para a construção e modificação do modelo de simulação.

1.9 Organização do texto

O restante deste texto está organizado da seguinte maneira. No capítulo 2 são apresentados os objetivos específicos do trabalho e a metodologia empregada para atingir estes objetivos. No capítulo 3 são apresentados o conceito de simulação interativa visual juntamente com suas características principais e os recursos disponíveis nos ambientes comerciais que passaram a utilizar tais características. As pesquisas realizadas em VIS, através de uma análise de diversos ambientes servem para apresentar as características necessárias para que um ambiente de simulação possa ser enquadrado como ambiente de VIS. O capítulo 4 discute modelagem interativa visual, apresentando suas características e formas de construção do modelo de simulação através da análise de ambientes que permitem tal recurso. Já o capítulo 5 apresenta o tema central da tese, qual seja, uma nova abordagem para ambientes de simulação que contemplem tanto os recursos de VIS quanto os de VIM. Neste capítulo é definida esta nova abordagem mostrando suas características e suas vantagens e desvantagens através de uma comparação com ambientes de VIS e VIM. No capítulo 6 é apresentado um ambiente que suporte esta nova abordagem, mostrando sua arquitetura e descrevendo com detalhes cada uma de suas ferramentas e recursos. O capítulo 7 descreve o protótipo implementado com o objetivo de validar as idéias apresentadas no capítulo anterior. O capítulo 8 mostra uma análise comparativa do ambiente proposto com outros correlatos. Finalmente, o capítulo 9 resume os resultados do trabalho apresentando a sua contribuição, conclusões e futuras extensões.

2 Objetivos do trabalho e metodologia

2.1 Objetivos específicos

O objetivo da tese é propor e avaliar a integração de simulação interativa visual (VIS - Visual Interactive Simulation) e modelagem interativa visual (VIM - Visual Interactive Modeling) num único ambiente, já que estas duas classes de ferramentas são comumente separadas. Esta abordagem é chamada de VISM (Visual Interactive Simulation and Modeling).

Para avaliar e validar esta abordagem, foi desenvolvido um protótipo de um ambiente chamado *VISME* (Visual Interactive Simulation and Modeling Environment) aplicado a um problema representativo. *VISME* é um ambiente integrado de modelagem e simulação de sistemas discretos onde haja formação de filas. Como exemplo de tais sistemas podem ser citados: caixa bancária 24 horas, supermercados, sistema de tráfego, agência bancária, pedágio em auto-estrada, sistemas de linhas de montagem, etc. De uma forma geral, um sistema de filas tem o seguinte comportamento: os clientes chegam para serem atendidos, esperam pelo serviço se não forem atendidos imediatamente (servidor está ocupado) e saem do sistema após serem atendidos. A figura 2.1. apresenta um sistema de filas com um único servidor, que poderia ser exemplificado como uma barbearia sendo atendida por um único barbeiro.

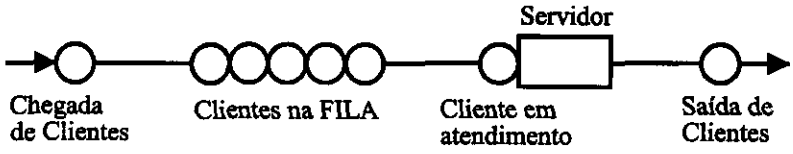


FIGURA 2.1 - Sistema de filas com um único servidor.

2.2 Métodos adotados

A metodologia empregada para atingir os objetivos seguiu os seguintes passos:

a) Definição dos termos VIS e VIM

Inicialmente foi realizado um levantamento bibliográfico genérico sobre simulação, ambientes de simulação e linguagens visuais. Com base nos estudos sobre os diversos temas selecionados, pode-se definir com maior clareza os conceitos dos termos VIS e VIM. Também foram determinados quais seriam os recursos e características desejáveis para ambientes de VIS e VIM.

b) Definição da abordagem VISM

A partir deste momento, a idéia de integrar em um único ambiente de simulação os recursos existentes em ambientes de VIS e VIM começou a amadurecer. Este fato levou a um novo estudo bibliográfico onde foram analisados diversos ambientes de simulação que tentassem integrar os conceitos de VIS e VIM. Através deste novo estudo foi criada uma nova abordagem integradora de VIS e VIM que foi chamada de VISM. Deste modo, pode-se concluir que um ambiente de VISM para ter o controle completo de todo o processo de simulação deveria ter: **a)** um conjunto de recursos para VIM; **b)** um conjunto de recursos para VIS; **c)** um conjunto de recursos adicionais que permitam auxiliar nas tarefas de experimentação e modelagem (modificação do modelo de simulação) durante a fase de experimentação. Estes resultados foram apresentados em [WAG96a] e [WAG96b].

c) Definição da aplicação-exemplo

A análise de diversos ambientes de simulação existentes juntamente com a idéia básica da abordagem VISM, que permite o controle completo do processo de simulação, pode-se verificar que seria muito complexa a definição de um ambiente de simulação de propósitos gerais. Deste modo, foi estabelecido que o domínio de problemas seria sistemas discretos com formação de filas. Com isto foi determinado que a aplicação exemplo utilizada seria a de uma agência bancária. Esta aplicação exemplo será mais detalhada na seção 2.3.

d) Definição do ambiente *VISME*

Neste momento foi definido um ambiente de simulação baseado na abordagem VISM, com suas características, ferramentas e recursos. Este ambiente foi chamado de *VISME.*

e) Avaliação do conjunto de recursos

A partir dos recursos identificados no item b), verificou-se quais são suficientes e/ou necessários para a aplicação definida no item c). Como forma de facilitar o projeto, alguns recursos foram incluídos, não sendo estritamente necessários.

f) Determinação de um subconjunto de recursos

Conforme a aplicação escolhida determinou-se um conjunto mínimo de recursos a serem desenvolvidos no ambiente *VISME* com o objetivo de validar a abordagem VISM.

g) Definição do protótipo do ambiente *VISME*

Neste momento foram definidas as estruturas de dados e funcionalidades do ambiente *VISME.*

h) Desenvolvimento do protótipo

Implementou-se o protótipo do ambiente *VISME* na linguagem de programação Delphi 3.0.

i) Implementação da aplicação exemplo no protótipo desenvolvido

Após o desenvolvimento do protótipo, foi construído o modelo da aplicação (agência bancária) no ambiente *VISME* e realizou-se os experimentos sobre ele.

j) Avaliação dos resultados

Com base nos resultados obtidos pode-se fazer uma análise do modelo construído, bem como uma avaliação do ambiente *VISME* e da abordagem VISM.

2.3 Aplicação exemplo

De acordo com o domínio de problemas escolhido (sistemas discretos com formação de filas), a aplicação exemplo utilizada será a de uma agência bancária. Esta agência será composta por seis caixas (cinco caixas normais e uma especial), três gerentes e três terminais eletrônicos, conforme mostrado na figura 2.2.

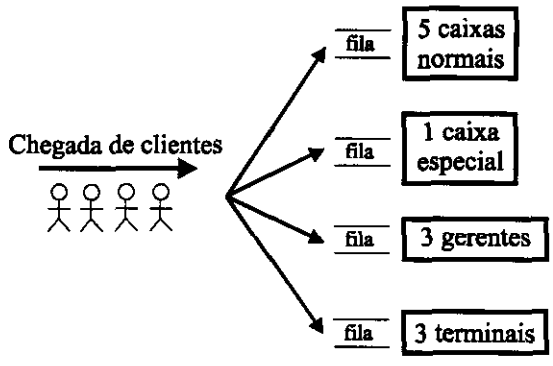


FIGURA 2.2 - Agência bancária.

Os clientes (entidades temporárias) chegam na agência a uma determinada taxa (distribuição de probabilidade) e utilizam os recursos oferecidos por ela. O objetivo deste problema é analisar o tempo de espera dos clientes na fila de um determinado recurso e o tempo de atendimento de um recurso. Na agência podem tanto existir filas específicas para um determinado recurso, como filas únicas para um tipo de recurso. Isto depende do modelo construído. Neste problema da agência esta característica será modelada no recurso caixa, onde existirá uma caixa específica para clientes especiais e as outras cinco caixas serão destinadas para os demais clientes (fila única).

À medida que os clientes chegam, eles são direcionados para os recursos existentes. Este direcionamento é regido por uma distribuição de probabilidade. Quando o cliente se dirige para um determinado recurso ele já tem a informação se este recurso possui fila específica ou uma fila única compartilhada com os demais recursos da mesma categoria. No caso de existir mais de um recurso disponível, a escolha será sempre pelo primeiro. Caso o recurso esteja ocupado ocorre a formação de fila. Após ser atendido, o cliente pode sair da agência ou utilizar qualquer outro recurso oferecido pela agência. As estatísticas são coletadas em dois pontos: após a saída de um cliente de cada uma das filas e após o atendimento em cada um dos recursos. Para ambos os casos deve ser registrado o momento de chegada do cliente.

3 Simulação interativa visual

3.1 Introdução

Ao longo dos anos o uso de simulação se difundiu para as mais diferentes áreas de aplicação, causando deste modo um crescimento considerável na complexidade das simulações bem como no volume de dados gerados e que devem ser analisados pelos usuários. Com isto, a interpretação dos resultados das simulações se tornou uma tarefa muito dispendiosa e muitas vezes cansativa e improdutiva. Este fato deu início às pesquisas visando o desenvolvimento de ferramentas que permitissem uma melhor compreensão dos dados, através de apresentações mais eficientes para a análise dos mesmos. A partir deste momento houve uma tendência de uso de representações gráficas e animações para os objetos em estudo, com o objetivo de prover uma melhor compreensão do experimento e facilidades de interação com os mesmos. Este é o objetivo da área conhecida como visualização de dados científicos.

Visualização científica é uma área relativamente nova dentro da Ciência da Computação que começou com a realização do "Workshop on Visualization in Scientific Computing - ViSC" em 1987 [MCC87, SPE87]. A visualização científica emprega técnicas de computação gráfica na apresentação de resultados de processamentos diversos. Recentemente novos métodos, técnicas e pacotes (softwares) tem sido desenvolvidos fazendo com que a visualização científica possa a ser aplicada em variadas áreas como: metereologia, medicina, geografia, oceanografia, astronomia, etc. Utilizando-se de representações gráficas para os objetos em estudo e recursos gráficos (sombras, texturas, realismo, etc) para converter dados complexos e idéias abstratas em figuras significativas, a visualização científica tem por finalidade uma apresentação mais eficiente dos dados para uma melhor compreensão por parte do usuário [BRO92, EAR92].

Os primeiros ambientes de simulação desenvolvidos que ofereciam recursos de visualização, na realidade utilizavam-na simplesmente como apresentação de dados, sem se preocupar com os aspectos interativos. Estes recursos oferecidos caracterizam o que chamamos de apresentação visual dos dados. A diferença entre visualização científica e apresentação visual dos dados é que, enquanto esta preocupa-se com uma apresentação mais sofisticada de resultados a título de comunicação, aquela preocupa-se em oferecer recursos para um melhor entendimento destes dados [EAR92], o que supõe interação. Assim, além de facilitar a comunicação, existe a necessidade de exploração dos mesmos para fins de compreensão dos fenômenos que eles representam. Este é o objetivo principal da análise exploratória dos dados [TUK77], aplicação principal da visualização científica. A análise exploratória visual dos dados reune um conjunto de técnicas e ferramentas com o propósito de "observar melhor os dados para ver o que eles dizem". Desta forma, a visualização científica tem por objetivo facilitar a análise exploratória dos dados e apresentar os dados de forma mais compreensível ao usuário leigo.

Atualmente os estudos na área de simulação estão mais voltados para o desenvolvimento de sistemas que ofereçam ao usuário facilidades para a simulação e modelagem. Além da necessidade de uma melhor visualização dos dados, é muito importante que os ambientes de simulação ofereçam recursos para a interação do usuário com os experimentos. Normalmente, o processo de simulação é bastante demorado. A não-apresentação de resultados intermediários e a falta de interatividade exigem que o usuário aguarde até o final do processo de simulação para analisar os resultados e faça suas alterações, caso julgue necessário. Muitas destas alterações podem exigir a retomada da simulação, fazendo com que o usuário perca muito tempo, pois poderia ter interrompido a simulação no meio, caso esta facilidade estivesse disponível. Deste modo, os termos interativo e visual estão cada vez mais relacionados com o processo de simulação [PID92]. Enquanto que o propósito da simulação é obter um nível mais aprofundado no entendimento dos dados, o propósito da interação é oferecer ao usuário a facilidade de poder trabalhar diretamente com o modelo e o processo de simulação, influenciando no seu comportamento e realizando experimentações com suas propriedades e relacionamentos.

3.2 Conceitos

Utilizando-se dos conceitos e características de simulação e computação gráfica, Hurrion [HUR76] apud [BEL87] apresentou pela primeira vez o termo Simulação Interativa Visual (VIS - *Visual Interactive Simulation*). VIS significa a condução de uma simulação pelo usuário com a visualização de dados intermediários e a possibilidade de interação com o modelo, para a modificação de parâmetros e variáveis, permitindo deste modo um controle completo do modelo durante este processo. Até a introdução do termo VIS, a visualização de resultados se fazia por meio de gráficos 2D e 3D e a simulação de sistemas discretos empregava linguagens de programação numéricas como FORTRAN e ALGOL. Diversos pacotes começaram a ser desenvolvidos nas décadas de 70 e 80 utilizando-se dos conceitos de VIS: SEE-WHY, WITNESS, FORSSIGHT, OPTIK (todos referidos em [BEL87]). Linguagens de simulação como SIMULA, GPSS, Simscript e SIMAN aos poucos passaram a conter aspectos de visualização. Um número de ferramentas com capacidade de animação também começaram a aparecer: CINEMA, Xcell, Tess, etc. Estas ferramentas permitem que se criem representações gráficas para os objetos utilizados no modelo durante o processo de simulação.

Atualmente, simulação interativa visual alcançou um estágio onde seu uso está se tornando comum na indústria e comércio em um grande número de aplicações. Em particular, VIS está sendo aplicado no projeto e análise de sistemas de manufatura, transportes (tais como aeroportos, linhas férreas e sistemas de tráfego) e em serviços organizacionais (tais como bancos, hospitais e restaurantes).

Conforme o grau de interação entre o usuário e o modelo durante o processo de simulação, utilizando recursos de visualização, Marshall [MAR90] estabelece três categorias para interação e visualização na simulação: pós-processamento ("post-processing"), monitoramento ("tracking") e controle

("steering"). A interação com os dados durante a simulação é inexistente com pós-processamento e máxima com controle.

A técnica mais comum e utilizada é a de ***pós-processamento***. As imagens são geradas depois da coleta e cálculo dos dados, não havendo interação com a fonte dos dados. Um ambiente de visualização acessa a base de dados criada pela simulação e gera as imagens. Uma das vantagens desta técnica é que os dados podem ser examinados várias vezes até um completo entendimento dos resultados. Em contrapartida, somente após o final da simulação, que poderá levar horas, é que os erros podem ser detectados e corrigidos.

Na técnica de ***monitoramento***, as imagens são visualizadas à medida que o processo de simulação avança. O usuário não tem controle sobre os parâmetros da simulação, somente sobre o encerramento do processo. A vantagem desta técnica é a observação imediata dos resultados. Deste modo, os erros podem ser detectados e corrigidos mais cedo e o processo pode ser encerrado se necessário. Os recursos computacionais também são melhor utilizados, uma vez que quando um procedimento incorreto for detectado, o processo de simulação pode ser interrompido evitando assim um maior tempo de processamento e de acesso à disco. Uma desvantagem desta técnica é que os resultados ficam disponíveis apenas no tempo de simulação onde foram calculados, ou seja, eles não podem ser analisados mais adiante caso o usuário necessite. Para contornar este problema, deve-se ter a possibilidade de armazenar os dados.

O ***controle*** é a técnica onde o usuário detém o controle direto do modelo computacional durante o processo de simulação podendo realizar a modificação de parâmetros da simulação durante a visualização. Diversas características de um modelo podem variar durante a execução, incluindo parâmetros internos, variáveis monitoradas, apresentação das saídas, intervalos de tempo, etc. Esta técnica é referenciada como uma meta na visualização científica, correspondendo à simulação interativa visual [BEL87].

3.3 O Uso da animação[1]

A animação de modelos constitui atualmente uma considerável parte no processo de simulação. Os principais pontos do processo de simulação onde a animação fornece um valioso auxílio são: desenvolvimento e verificação do modelo, validação do modelo, análise e projeto dos experimentos e, apresentação dos resultados. Observar uma animação para descobrir uma operação inválida é uma tarefa muitas vezes complexa. O usuário deve compreender os elementos gráficos apresentados, as relações entre estes elementos e detectar quando as assertivas iniciais do modelo forem violadas, caso elas sejam. Devido a isto, Swider [SWI94] apresenta 6 princípios básicos para uma efetiva animação: 1) usar um cenário gráfico com

[1]No contexto da área de simulação, animação é empregada como sinônimo de apresentação dinâmica de resultados, sendo a visualização do comportamento das entidades representadas graficamente ao longo do tempo.

movimentos de ícones para as entidades do modelo; 2) projetar o cenário tendo em mente a tarefa de validação; 3) utilizar contrastes visuais para indicar a existência de problemas; 4) ajustar a velocidade de apresentação dos resultados de forma que os eventos discretos fiquem evidenciados; 5) evitar a sobrecarga do usuário com muita informação visual; 6) treinar o usuário na procura de potenciais problemas na animação.

A animação é uma característica essencial para sistemas de VIS, auxiliando na depuração, interpretação e apresentação de resultados. Quando usada durante o processo de simulação, na fase de experimentação, ela permite uma compreensão visual mais detalhada do processo de simulação. As representações gráficas dos diversos componentes do sistema são utilizadas para animar as atividades da simulação, ou seja, sempre que ocorre uma alteração no estado da simulação, isto irá refletir na representação gráfica do sistema modelado. A animação é um método para apresentação da simulação de fenômenos contínuos através da exibição de uma coleção discreta de imagens. Através de rápidas atualizações na apresentação da imagem, tem-se a impressão de se observar uma atividade contínua. Esta capacidade é muito valiosa para a interação com o usuário, uma vez que problemas no modelo podem ser melhor visualizados e corrigidos, reduzindo o tempo de depuração. A animação aumenta a credibilidade do modelo de simulação, especialmente para os usuários menos experientes. A animação em um sistema de VIS, também pode ser utilizada após o término da simulação, em pós-processamento, permitindo uma apresentação mais eficaz dos resultados da simulação e auxiliando ao tomador de decisões verificar se a simulação realmente reflete o sistema real.

Os recursos de visualização em ambientes de simulação devem suportar as diversas fases do processo de simulação. Johnson e Poorte [JOH88] identificam 4 fases onde a animação pode ser empregada durante o processo de simulação: depuração e verificação do modelo, validação do modelo, análise e apresentação dos resultados. De acordo com esta identificação, foi proposta uma abordagem hierárquica com três níveis de utilização da animação e representação gráfica, em função da abstração do processo de simulação conforme apresentado na figura 3.1.

O uso da animação é hierárquico e diferenciado de acordo com a fase do processo de simulação, ou seja, os componentes e elementos gráficos vão sendo incrementados visualmente a partir de uma animação simples, na primeira fase, até uma representação completa do modelo na última fase. Os níveis propostos são os seguintes:

1) Nível I: uma animação 2D com representação gráfica simplificada e mudança de cores seria o suficiente para as fases de depuração e verificação do modelo. É uma animação bastante abstrata e com poucos detalhes;

2) Nível II: também caracterizado com um sistema 2D, mas incluiria alguns movimentos e alteração de cores. Seria uma animação menos abstrata, tornando possível a utilização deste nível nas fases de verificação, depuração e análise, podendo também ser empregado como uma ferramenta de comunicação geral;

3) Nível III: o detalhe em relação ao anterior é maior e também mais realístico (sem abstração). Caracterizado como uma animação 3D com movimentos

significativos e representação gráfica detalhada através de ícones e símbolos bastante representativos. É utilizado como uma ferramenta para o nível gerencial, de treinamento, educacional e vendas. Na análise de resultados, esta abordagem pode ser classificada dentro da categoria de interação de pós-processamento, uma vez que a animação é utilizada somente para a visualização dos resultados.

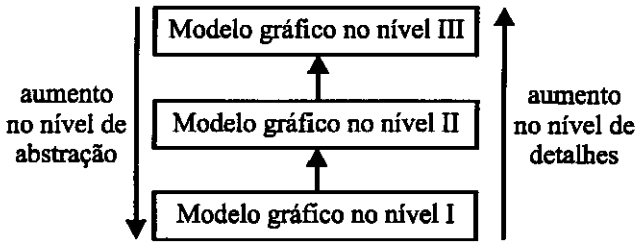


FIGURA 3.1 - Abordagem hierárquica para animação na simulação.

Um ponto que deve ser ressaltado é que simulação interativa visual é fundamentalmente diferente de animação, uma vez que a motivação de VIS é a integração com o processo de simulação, não simplesmente a representação gráfica dos resultados da simulação. VIS caracteriza-se pelo controle (steering) da simulação, ou seja, há a visualização de dados intermediários e capacidade de interação com o modelo durante o transcorrer da simulação. Podem ser modificados parâmetros e definições de variáveis, forma de apresentação dos resultados, etc fazendo com que as consequências no processo de simulação sejam imediatas. Deste modo, VIS engloba a realização interativa da etapa de experimentação, utilizando representações gráficas e, em alguns casos, animação [BIS90, STA90, VUJ90].

3.4 Sistemas e linguagens de simulação com recursos gráficos

Em geral, até poucos anos atrás, as linguagens de simulação tradicionais como GPSS, Simscript, SLAM, etc requeriam a especificação do modelo a ser simulado através de uma descrição puramente textual que era executada sem a intervenção do usuário. Recentemente com os avanços da área de sistemas gráficos, as linguagens de simulação começaram a incorporar recursos para a melhoria de sua interface com o usuário. Surgiram sistemas gráficos e interativos com técnicas simplificadas de animação de resultados, como SLAMSYSTEM [BAN94] para modelos escritos em SLAM. Já SIMGRAPHICS [CAC91], para modelos Simscript II.5, permite a construção de interfaces gráficas para o acompanhamento dos experimentos através de cenários icônicos e gráficos diversos. Como evolução de GPSS, surgiu GPSS/PC [COX87] que incorpora facilidades de interação gráfica e animação em tempo de execução.

SLAMSYSTEM [BAN94] é um sistema de simulação integrado construído sobre Microsoft Windows para uso em computadores pessoais. Todas as facilidades são acessadas em sua interface gráfica dirigida por "pull-down menus". O modelo pode ser construído através de um construtor gráfico para a estrutura de rede

de filas e um editor de formulários para a especificação dos parâmetros de cada um dos símbolos da rede. O sistema possui uma análise dos resultados através de gráficos estatísticos como histogramas, diagramas de torta e de barras, gráficos X-Y, etc. Várias janelas podem ser abertas simultaneamente para diferentes formas de análise dos resultados. Os resultados podem ser analisados de forma gráfica ou animada, em tempo de execução ou em pós-processamento. SLAMSYSTEM também permite a construção do cenário através de suas próprias facilidades gráficas ou através da importação de um cenário de outros pacotes de CAD. A animação é especificada através da associação de eventos na simulação com ações de animação. Quando acontece um evento, ocorre uma alteração na representação gráfica de uma entidade do modelo, causando uma ação de animação.

GPSS/PC [COX87] é uma das implementações mais difundidas de GPSS em computadores pessoais. A introdução de interação através de diferentes janelas e dispositivo de apontamento, permite que o modelo seja manipulado em tempo de execução. Entretanto, como a obtenção de imagens mais realistas implica em um maior tempo de execução, uma animação mais sofisticada em pós-processamento está disponível através de GPSS/PC Animator. O sistema não permite a geração de relatórios quando há interação, pois a alteração do modelo durante a execução distorce os resultados estatísticos.

Em GPSS/PC a visualização e manipulação da simulação é efetuada através de 6 janelas, cada uma com um conjunto de comandos, incluindo a possibilidade de interromper e continuar a simulação. Existem janelas para cada tipo de entidades principais em GPSS: "facilities", "storages", "blocks", "matrices", "tables" e "positions". Na janela "blocks", é apresentado o fluxograma do modelo, sendo que os blocos podem ser inseridos, removidos e manipulados. Na janela "positions" é exibida a animação da simulação onde é mostrada a evolução das transações durante a simulação. A animação é controlada através da modificação dos atributos gráficos das transações (forma, cor e posição). Existem dois modos distintos de controlar a animação: gerado automaticamente pelo sistema que realiza a prevenção contra colisão ("collision prevention mode") ou pode ser programado ("direct mode animation"), onde a tarefa de evitar colisões dos objetos (ícones associados às transações) fica a cargo do modelador. Assim como todas as demais janelas, os objetos podem ser manipulados através do dispositivo de apontamento, que pode ser usado para interromper, continuar e modificar posições das transações. Para obter simulações sem visualização/interação, GPSS/PC fornece uma janela estática, chamada "data window", onde o código fonte do modelo é exibido e não há visualização intermediária.

SIMGRAPHICS [CAC91] é uma biblioteca incorporada à linguagem de simulação Simscript II.5, que permite a construção de interfaces baseadas em cardápios e a obtenção de resultados da simulação sob as mais diversas formas gráficas (histogramas, gráficos, diagramas de barra, seqüências animadas, etc). A utilização dos recursos gráficos é realizada em duas etapas. Primeiramente, através de um editor gráfico guiado por cardápios, são especificados os ícones, a forma de entrada dos dados e os gráficos de saída. Este editor cria um arquivo que contém as descrições destas representações gráficas. A seguir, em tempo de execução, estes

arquivos são ligados ao programa escrito em Simscript II.5. Durante a execução do modelo, podem ser alterados os parâmetros do modelo, os ícones utilizados ou as saídas desejadas, mas estas modificações devem estar programadas no modelo descrito em Simscript. SIMGRAPHICS restringe os gráficos a um contexto bidimensional, o que pode ser uma séria limitação caso se queira empregar a simulação no terceiro nível da abordagem hierárquica para animação, como proposto por [JOH88]. MODSIM II [BEL90] é uma linguagem de simulação de propósitos gerais, orientada a objetos, que também utiliza os recursos oferecidos por SIMGRAPHICS. Este garante a exibição e interação através de ícones, gráficos e cardápios que podem ser associados a objetos e variáveis do programa.

Juntamente com este avanço gráfico nas linguagens de simulação, surgiram diversos sistemas para simulação. Basicamente estes sistemas possuem uma interface gráfica dirigida por cardápios que permite a construção, execução e análise de modelos de simulação. A grande maioria destes sistemas está voltada para a modelagem e simulação de sistemas de manufatura. Como exemplos, podem ser citados SIMFACTORY II.5, AutoMod, WITNESS, Taylor II e ProModel (todos apresentados resumidamente em [BAN94]). Todos tem em comum a construção do modelo através de uma ferramenta gráfica que posiciona a representação gráfica dos elementos de modelagem (ícones) em um cenário, criando desta forma uma representação gráfica para o modelo. Durante a execução do modelo é realizada a animação e diversos gráficos e dados estatísticos são oferecidos para a análise de resultados.

3.5 Análise dos ambientes de VIS

Esta seção tem por objetivo descrever resumidamente alguns ambientes de simulação que fornecem recursos visuais interativos para a análise dos resultados das simulações. Conforme as características e recursos apresentados por estes ambientes, pode-se enquadrá-los dentro da classificação proposta por Marshall [MAR90]. Os seguintes ambientes foram analisados durante o desenvolvimento deste trabalho: Proof [EAR90], CINEMA [POO90], Performance Analysis Workstation [MEL85], ambiente de simulação do sistema AMPLO [WAG92], SEE-WHY [FID81], ambiente de simulação "centrado no usuário" [ROO93] e VISE [LIN95].

3.5.1 Ambientes com apresentação dos dados em pós-processamento

Os ambientes de simulação que estão classificados dentro desta categoria, são aqueles que apresentam as tarefas do processo de simulação bem separadas no tempo (figura 3.2). Primeiramente, o usuário constrói o modelo de simulação, eventualmente através de uma ferramenta contendo recursos gráficos e interativos. A seguir, este modelo é submetido à simulação sobre a qual o usuário não tem controle. Os resultados gerados pela simulação são armazenados numa base de dados que pode ser acessada após a execução do modelo. A partir deste momento, o usuário tem à sua disposição uma série de recursos visuais e interativos que permitem exibir os dados armazenados no banco de dados criado durante a simulação. Nesta categoria de

visualização o usuário não possui recursos para interagir na fase de experimentação, somente na fase de análise de resultados.

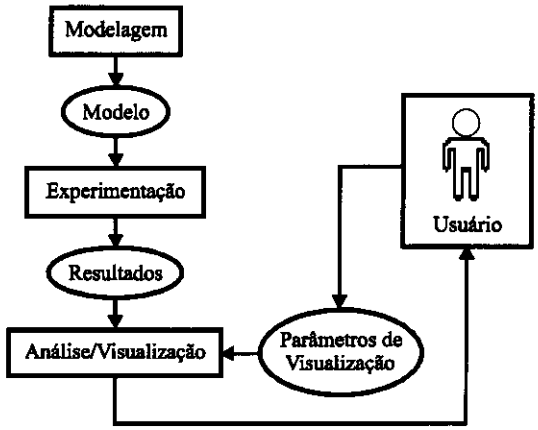


FIGURA 3.2 - Ambiente de simulação com a técnica de pós-processamento.

A maioria dos sistemas que realizam a apresentação dos resultados em pós-processamento é constituída pelos sistemas de animação desenvolvidos para a integração a modelos de simulação descritos em linguagens de simulação que não possuíam recursos gráficos para a análise de resultados. Exemplos de tais sistemas são Proof, para modelos em GPSS, e CINEMA, para modelos em SIMAN.

3.5.1.1 Proof

Proof [EAR90] é um animador em pós-processamento de resultados de simulação. Proof foi criado para ser utilizado com GPSS/H, mas por ter uma arquitetura aberta pode ser utilizado com qualquer outro software, desde que este seja capaz de produzir um arquivo ASCII. Este arquivo ASCII é o arquivo de "trace" que dirige a animação. Por ser utilizado em tempo de pós-processamento, Proof pode ser empregado como software de apresentação, com facilidades de produção de "slides". Para a execução de Proof são necessários dois arquivos: um arquivo de "layout" e um arquivo de "trace". O arquivo de "layout" é utilizado para armazenar as definições estáticas de geometria e comandos de inicialização. O arquivo de "trace" é usado para gravar, durante a simulação, as informações dependentes do tempo que controlam a animação.

A estrutura gráfica de Proof é composta por duas camadas: o cenário e os objetos. Através de uma série de menus o cenário e os objetos podem ser criados. Em relação aos objetos, dois conceitos importantes de Proof são "ObjectClass"e "Object". O primeiro é a descrição geométrica estática de algum tipo de objeto e corresponde ao conceito de tipo ou classe. O segundo são ocorrências ou instâncias dos tipos e variam conforme os valores de seus atributos. Toda a animação é realizada sobre estes objetos. As alterações no estado do modelo de simulação durante a simulação serão exibidas como movimentos, alteração de forma e cor, aparecimento e

desaparecimento de objetos. As primitivas de animação que podem ser usadas em um objeto são: "create", "destroy", "place", "set color", "set velocity" e "move". O movimento de um objeto pode estar associado a um "path" que é uma estrutura de dados composta por um número arbitrário de linhas ou arcos. Quando um objeto é colocado em um "path", ele o segue até o final. Os "paths" tem um papel muito importante em animações de sistemas de manufatura para transporte e manipulação de materiais. Baseado na sua estrutura de dados geométrica, Proof provê facilidades interativas de controle de visualização durante o processo de animação. Janelas sobre o cenário da animação permitem realizar funções como "panning", "zooming", rotacionar e mudar o ponto de vista.

3.5.1.2 CINEMA

CINEMA [POO90] é um sistema de animação também de propósitos gerais utilizado para animar modelos de simulação desenvolvidos na linguagem de simulação SIMAN. CINEMA consiste de dois módulos: um usado para construir a representação gráfica utilizada na animação e outro usado para executar o modelo de simulação SIMAN associado com a animação. Uma vez que CINEMA é um sistema de animação em tempo real, em oposição a sistemas de pós-processamento, o usuário pode atuar sobre o modelo SIMAN no decorrer da simulação através de um depurador interativo. Este depurador permite que se altere valores de variáveis, observe o status de uma variável que não participa da animação, monitore entidades e variáveis e, salve/recupere instantâneos da animação (imagens e valores). Através do uso de CINEMA, as contribuições da animação podem ser encontradas em todo o processo de simulação. Um exemplo interessante do uso de SIMAN/CINEMA foi apresentado por Johnson & Poorte [JOH88], que propuseram uma abordagem hierárquica da animação na simulação. Neste trabalho, a animação é utilizada em várias etapas de um estudo de simulação, alterando o nível de informação presente nas imagens a cada etapa.

CINEMA possui uma interface gráfica dirigida por "pop-up menus", permitindo que o usuário construa sua animação através da seleção das diversas opções existentes. O cenário de animação de CINEMA é constituído de objetos gráficos divididos em dois tipos de componentes: um componente estático, chamado "background", e um componente dinâmico. O componente estático do cenário não muda durante a execução da simulação e pode ser criado através das facilidades gráficas disponíveis no editor gráfico existente. Um ponto importante de CINEMA, é a possibilidade de importar um cenário de animação construído com o uso de ferramentas de CAD que possuem funções gráficas mais sofisticadas, incluindo imagens 3D. O componente dinâmico do cenário consiste de representações gráficas que mudam de forma, cor, tamanho ou posição conforme as mudanças no estado do modelo SIMAN. Um número grande de objetos dinâmicos estão disponíveis, representando entidades, recursos, filas, caminhos e variáveis. Estes objetos dinâmicos podem ser armazenados em bibliotecas e acoplados a diferentes cenários.

3.5.2 Ambientes que realizam a análise com monitoramento

Os ambientes de simulação que utilizam a técnica de monitoramento são aqueles que possuem uma série de recursos para auxiliar o usuário na fase de experimentação (figura 3.3). Estes recursos devem ser interativos para que o usuário conduza o processo de simulação. Os ambientes classificados nesta categoria, também devem possuir as características e facilidades existentes nos ambientes de pós-processamento para a análise de resultados. A grande diferença destes sistemas em relação aos que utilizam a técnica de pós-processamento é que durante a fase de experimentação o usuário pode acompanhar a simulação através de recursos de visualização e de condução da experimentação. Desta forma, o usuário pode analisar os resultados em tempo de execução e interromper a execução da simulação quando desejar.

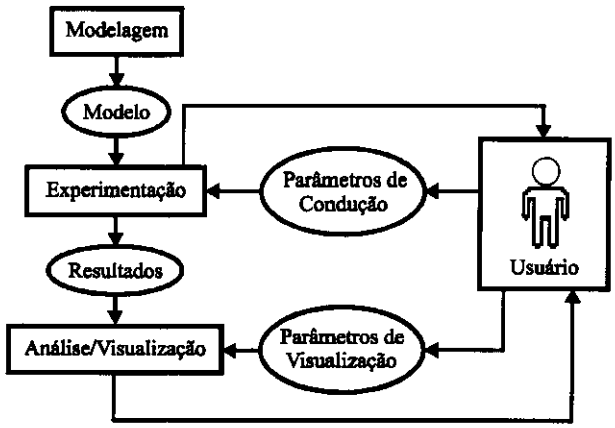


FIGURA 3.3 - Ambiente de simulação com a técnica de monitoramento.

Os ambientes de simulação que fazem uso desta técnica, geralmente oferecem representações gráficas para entidades do modelo (associação realizada na construção do modelo de simulação), facilidades de animação, múltiplas janelas exibindo diversas informações, gráficos e estatísticas. Além destes recursos visuais, o usuário tem a sua disposição uma série de recursos para permitir a condução da experimentação: determinação do tempo de execução da simulação, seja através de condições de parada ou indicação de um período de tempo; indicação e manipulação do conjunto de variáveis a serem monitoradas; operações para armazenar e recuperar estados da simulação. Um ponto que deve ser ressaltado é que estes ambientes não permitem a interação do usuário com o modelo, ou seja, o usuário não pode controlar parâmetros, variáveis ou atributos de entidades do modelo. Caso o usuário necessite realizar uma destas operações, ele deve aguardar pela interrupção da execução e depois alterar o modelo através da ferramenta que permite a construção do modelo de simulação. A partir deste momento, o usuário deve realizar novamente a execução da simulação, desde o início, uma vez que as características do modelo foram modificadas. Exemplos de tais sistemas são Performance Analysis Workstation e o ambiente de simulação do sistema AMPLO.

3.5.2.1 Performance Analysis Workstation

O sistema **Performance Analysis Workstation** [MEL85] foi um dos primeiros sistemas de simulação visual interativa surgidos. O sistema é destinado a modelagem e simulação de redes de filas e apresenta uma interface dirigida por cardápios. Este sistema é estruturado por três componentes principais: um editor gráfico, que possibilita a criação interativa de uma rede (primeiramente são criadas as filas e depois a topologia é indicada através do desenho das conexões); um editor textual, dirigido por preenchimento de formulários e utilizado para informar parâmetros das filas (disciplina da fila, capacidade, número de servidores, distribuições de probabilidade regendo chegadas, serviço e roteamento) e criar as suas populações iniciais; e um simulador, que é o responsável pela execução do modelo construído.

No simulador existem comandos selecionáveis e campos de informação sobre diferentes modos de execução e relógios de simulação. Existem dois modos de execução: passo a passo, onde o usuário indica o avanço da simulação através de uma tecla e, o modo contínuo, onde o simulador executa até ser interrompido. Em qualquer um destes modos, o usuário pode especificar um intervalo de tempo para poder observar o estado da simulação. A simulação pode ser interrompida a qualquer momento para a realização de alterações na descrição do modelo ou para a requisição de estatísticas. As estatísticas são exibidas em diversas janelas, sobrepostas à principal, conforme a solicitação do usuário. Também são oferecidas facilidades para salvar e recuperar estados intermediários da simulação, com o objetivo de retomar uma sessão após a sua interrupção.

3.5.2.2 Ambiente de simulação do sistema AMPLO

Outro exemplo de ambiente que utiliza a técnica de monitoramento é o ambiente de simulação do sistema AMPLO [WAG92] que permite a simulação de sistemas digitais modelados através de linguagens de descrição de hardware. Embora orientado para uma aplicação específica (projeto de sistemas digitais), este ambiente implementa muitos dos recursos sugeridos em VIS. Os sistemas digitais são modelados graficamente como redes de agências, blocos que se comunicam através de sinais. Esta rede de agências pode ser hierárquica, ou seja, as agências podem ser descritas em função de outras agências. Na construção do modelo de simulação, o usuário deve selecionar nesta hierárquia de agências, as agências que farão parte do modelo de simulação. Esta seleção pode ser realizada através de cardápios ou por intermédio de um "browser" interativo que percorre a estrutura hierárquica. O próprio construtor de modelos se encarrega de gerar uma descrição simulável do modelo, uma vez que o mesmo deve ser uma rede de agências primitivas, sem hierarquia.

Durante a fase de experimentação, o ambiente permite o monitoramento e a condução dos experimentos, com visualização gráfica do modelo de simulação e das variáveis monitoradas (formas de ondas dos sinais). Diversas facilidades estão disponíveis para o usuário durante a execução do modelo: ativação e desativação das variáveis monitoradas; indicação do modo de monitoramento das variáveis, em tempo

de execução ou para análise em pós-processamento; especificação de condições de parada e determinação do tempo de simulação; salvar e recuperar estados da simulação, permitindo o retorno a um tempo anterior e consultas a valores de variáveis específicas, mesmo que estas não estejam sendo monitoradas. A análise em pós-processamento é realizada através de um arquivo de "trace" gerado durante a experimentação, contendo os valores das variáveis monitoradas. Embora o ambiente de simulação do sistema AMPLO seja enquadrado como um ambiente que realiza a análise com monitoramento, ele possui duas facilidades que podem ser enquadradas como controle da simulação: vinculação de estímulos ao modelo (seqüência de eventos associados às variáveis de entrada e/ou às variáveis internas do modelo) e atribuição de valores à variáveis. Tais facilidades são classificadas como uma interação entre o usuário e o modelo.

3.5.3 Ambientes que utilizam a técnica de controle

Os ambientes de simulação que são classificados dentro desta categoria são aqueles que apresentam uma série de recursos para permitir que o usuário interaja com o modelo durante a simulação (figura 3.4). Estes recursos, geralmente estão disponíveis em três ferramentas distintas: uma que permite a visualização e interação com o modelo de simulação, normalmente através de recursos gráficos; um módulo que permite ao usuário controlar a simulação através dos recursos interativos disponíveis; um módulo que permite a análise dos resultados através de diversas facilidades de visualização. Através da integração destes três módulos, o usuário possui um ambiente de simulação geralmente chamado de ambiente de VIS.

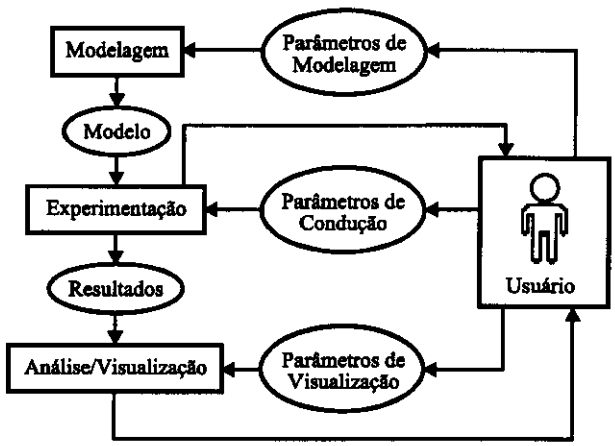


FIGURA 3.4 - Ambiente de simulação com a técnica de controle.

Após criar o modelo de simulação, o usuário o submete ao ambiente de simulação que fornece uma série de recursos interativos permitindo o controle sobre a execução. Além dos recursos oferecidos pelos ambientes que permitem monitoramento e pós-processamento, o usuário tem a sua disposição recursos que permitem interagir com o modelo de simulação, tais como: alteração de parâmetros

do modelo; alteração de valores de variáveis e atributos de entidades e redefinição de distribuições. Esta interação do usuário com o modelo é realizada durante a experimentação, causando automaticamente reflexos nos resultados. Desta forma, a principal diferença entre os ambientes que utilizam a técnica de monitoramento e os ambientes que utilizam a técnica de controle é que estes podem interagir com o modelo de simulação e controlar a experimentação, enquanto aqueles somente podem conduzir a experimentação. Como exemplos de tais sistemas podem ser citados SEE-WHY, o ambiente de simulação "centrado no usuário" proposto por [ROO93] e VISE.

3.5.3.1 SEE-WHY

O primeiro sistema que se utilizou de características visuais interativas foi ***SEE-WHY*** ([FID81] apud [FRE91]). SEE-WHY é um sistema de simulação discreta de propósitos gerais que oferece facilidades integradas para exibição e interação em tempo de execução. Embora sendo uma ferramenta antiga de simulação, a representação gráfica utilizada e a variedade de gráficos foram revolucionárias para a época. Os modelos são desenvolvidos como programas FORTRAN incorporando rotinas da biblioteca SEE-WHY que suportam todas as funções apropriadas para a simulação. Esta biblioteca também provê facilidades para controlar a simulação e a interação com o usuário. As rotinas para definição e manipulação dos elementos de modelagem ("entities", "sets", "vessels", "histograms" e "distributions") formam a maior parte da biblioteca. A característica principal destes elementos de modelagem em SEE-WHY são os atributos relacionados com a exibição (posição, cor, altura e largura dos elementos). A interação com o usuário em SEE-WHY foi um ponto importantíssimo para o crescimento de sistemas de VIS. Os seguintes recursos estão disponíveis: interromper a simulação e consultar ou alterar qualquer parâmetro dos elementos de modelagem e atributos de entidades; especificar novamente as distribuições; estabelecer o modo de execução da simulação (passo a passo ou contínuo); salvar e recuperar estados intermediários, podendo reinicializar o tempo de simulação.

3.5.3.2 Ambiente de simulação centrado no usuário

Rooks [ROO93] desenvolveu um ambiente de simulação interativa que apresenta o paradigma "centrado no usuário". Este paradigma permite que sistemas de simulação sejam desenvolvidos com interfaces consistentes entre usuário, modelo e sistema de simulação. Esta abordagem requer que o projetista da simulação defina os modos com os quais o usuário, o modelo e o sistema de simulação irão interagir. O paradigma "centrado no usuário" é orientado para os usuários do sistema de simulação. A idéia é retirar do modelo de simulação os aspectos relativos ao controle de experimentos. O projetista do modelo de simulação deve se preocupar somente com a lógica e estruturas de dados do modelo, e o analista da simulação deve interagir com o modelo e o sistema de simulação para realizar os experimentos adequados. Rooks propõe, desta forma, um ambiente de propósito geral para desenvolvimento de aplicações de simulação, possibilitando a configuração com diferentes linguagens. O

ambiente possui uma biblioteca de facilidades para o controle da execução do modelo e interação, que são independentes do modelo de simulação. Este ambiente é apresentado como um "framework"que contém procedimentos projetados para suportar todos os aspectos de modelagem e análise como: especificação e manipulação dos dados, entradas e saídas gráficas, intervenção e interação na execução do modelo, atividades de pré e pós-processamento e controle da experimentação.

O sistema proposto por Rooks é composto por quatro subsistemas: ambiente de pré-processamento, ambiente de desenvolvimento do modelo, ambiente de análise do modelo e sistema de cenário. O ambiente de pré-processamento permite que através de vários procedimentos os dados brutos sejam organizados, analisados e convertidos para o formato compatível com o do sistema. No ambiente de análise do modelo é realizada a interação entre usuário e modelo através dos recursos preparados no sistema de cenário. Duas ferramentas estão disponíveis neste ambiente: um processador de interações que oferece os recursos necessários para um ambiente de VIS [ROO91]; e um controlador de execuções que contém uma série de procedimentos para o controle da execução do modelo. Todas as intervenções realizadas pelo usuário são interpretadas; os resultados apropriados são gerados, com imediato reflexo na representação visual. O sistema de cenário permite a descrição dos experimentos e é formado por um conjunto de estruturas de dados e procedimentos que conectam o modelo e os dados da simulação em uma árvore hierárquica de cardápios e tabelas. Esta forma de organização sendo bem definida na especificação do cenário, permite uma comunicação eficiente dos dados, parâmetros e experimentos entre o modelo e o sistema de simulação. Todos os dados do ambiente são armazenados em tabelas de dados que são apresentadas ao usuário na forma de cardápios ou planilhas. Finalmente, o ambiente de desenvolvimento do modelo permite a construção do modelo de simulação através de um editor gráfico, para criar as representações gráficas, e um editor textual, para as especificações necessárias. Esta descrição do modelo de simulação é compilada e gera um modelo executável para ser utilizado pelo ambiente de análise do modelo. Durante a experimentação, o usuário não pode modificar o modelo de simulação, apenas interagir com os seus parâmetros e variáveis.

3.5.3.3 VISE

O ambiente ***VISE*** (Visual Interactive Simulation Environment) proposto por Lindstaedt [LIN95] é um ambiente de simulação visual interativa que utiliza a linguagem Simscript II.5 como ferramenta de modelagem e oferece uma série de recursos de interação, controle e visualização com a finalidade de facilitar a atividade de simulação. Este ambiente provê características de associação de representações gráficas a entidades do modelo, especificação da dinâmica destas representações e possibilidade de interação à nivel de experimento. Baseado na proposta de Johnson [JOH88] que descreve uma abordagem hierárquica para a animação na modelagem de simulação, este ambiente oferece recursos bastante úteis quando esta abordagem hierárquica é adotada. VISE possui três diferentes classes de recursos: recursos de interação, recursos de controle e recursos de visualização.

Através dos recursos de interação de VISE, o usuário pode interagir com as variáveis e atributos das entidades do modelo através das seguintes facilidades: visualização, atribuição, alteração e monitoração. O usuário também pode alterar os tipos e parâmetros das distribuições de probabilidades utilizadas no modelo. Os recursos de controle são diversos mecanismos oferecidos pelo ambiente que permitem um controle da execução da simulação. Entre eles destacam-se: escolha do modo de execução (passo a passo, por tempo específico, ocorrência de um evento ou até determinado tempo), controle de pontos de parada (recurso bastante útil na fase de depuração e podem ser de dois tipos: parada condicional ou por ocorrência de evento.), controle da lista de eventos, status da simulação (estatísticas) e operações de carga e salvamento de estados da simulação (bastante útil quando uma interação foi realizada no processo e quer se voltar a este momento). A classe de recursos de visualização permite que se especifique associações de variáveis e atributos de entidades a ícones com a finalidade de se visualizar graficamente a simulação. No decorrer da simulação é possível alterar os parâmetros destas associações. Dentro desta classe, também são oferecidos recursos para acompanhar a alteração de determinadas variáveis e atributos de entidades através de gráficos estatísticos como histogramas, diagramas de torta, gráficos X-Y e X-Y-Z, relógios, diais, etc.

3.6 Características e requerimentos para ambientes de VIS

VIS é, portanto, uma abordagem que requer a participação do usuário com o objetivo de conduzir os experimentos com sucesso. Um sistema completo de VIS deve dispor de facilidades que auxiliem ao usuário com menos conhecimento e/ou experiência em simulação a evitar muitas decisões errôneas que ele poderia cometer em várias das atividades da fase de experimentação, tais como: especificação dos dados de entrada da simulação, análise e interpretação dos resultados da simulação, validação do modelo de simulação, etc.

O ponto principal de VIS é prover ferramentas ao usuário para que ele possa melhorar o entendimento da dinâmica do sistema que está sendo modelado. Neste sentido, Rooks [ROO91] apresenta 4 requisitos que podem ser aplicados a modelos computacionais através do processo de simulação levando em consideração as formas de interação que podem ocorrer entre o usuário, o modelo e o ambiente de simulação: a ***intervenção*** visa prover o usuário de uma forma de interação com o modelo; a ***inspeção*** deve permitir ao usuário o acesso a todos os dados relevantes do modelo para realizar os experimentos; na ***especificação*** o usuário deve ser capaz de especificar os parâmetros do modelo, de acordo com os objetivos para a análise do modelo; a ***visualização*** deve fornecer ao usuário a capacidade de visualizar os dados do modelo de forma que ele possa compreender a sua dinâmica.

Cada um destes requisitos pode ser considerado como necessário para um ambiente de VIS e todos agrupados são suficientes para se obter um sistema completo de VIS. A intervenção é um requisito fundamental e sem ela um sistema não pode ser considerado interativo, uma vez que o usuário não pode interagir com os parâmetros do modelo e o ambiente durante o processo de simulação. Inspeção, especificação e visualização podem ser incluídos como modos de intervenção. A intervenção do

usuário pode ser realizada de diferentes maneiras. Uma maneira seria quando o usuário realiza um comando no ambiente e este produz uma mudança no estado do modelo ou do sistema ou este comando causa uma nova situação que espera por outra ação do usuário. Este é um tipo de intervenção que é iniciada pelo usuário. Outra forma de intervenção seria quando o modelo está num estado pré-determinado e o processo de simulação fica aguardando uma informação do usuário. Esta segunda forma de intervenção é iniciada pelo modelo. De forma semelhante a esta última, uma intervenção iniciada pelo sistema ocorreria quando o ambiente de VIS alcançasse uma condição tal que necessitasse de uma informação do usuário para prosseguir.

Uma intervenção bem estabelecida seria quando o usuário detecta o momento (tempo) correto de realizar a interação. As intervenções causadas pelo modelo e pelo sistema, determinam o tempo correto da interação. Já a intervenção iniciada pelo usuário é realizada de acordo com sua percepção e entendimento dos dados que estão sendo exibidos para ela. Algumas ferramentas devem ser fornecidas pelo sistema quando uma intervenção é realizada. Sempre que ocorre uma interação, alguma decisão foi tomada em algum ponto do processo de simulação, ocasionando uma determinada resposta do sistema. Deste modo é necessário que esteja disponível um mecanismo que permita salvar a descrição completa do estado do sistema. É muito importante que se tenha um histórico de todas as intervenções realizadas no modelo e os respectivos reflexos, de modo que o usuário possa recuperar o estado do sistema antes de uma determinada intervenção que ele julgou que produziu os resultados incorretos no momento atual da simulação. Através de uma análise no histórico das intervenções, o usuário pode encontrar qual a intervenção que ocasionou este problema e, recuperar o estado da simulação naquele ponto, sem a necessidade de recomeçar a simulação desde o início.

Baseando-se no estudo de diversos sistemas e especificações [VUJ90, FRE94a, FRE94b], chegou-se a algumas conclusões de quais seriam as características desejáveis a um sistema de VIS a saber:

a) visualização gráfica do modelo de simulação;

b) possibilidade de interação do usuário com o modelo, ou seja, com sua representação gráfica;

c) possibilidade de interação a nível de experimento;

d) possibilidade de armazenar informações em uma base de dados para análise em pós-processamento;

e) possibilidade de interação em pós-processamento.

Conforme estas características, a figura 3.5 apresenta a arquitetura de um ambiente de VIS.

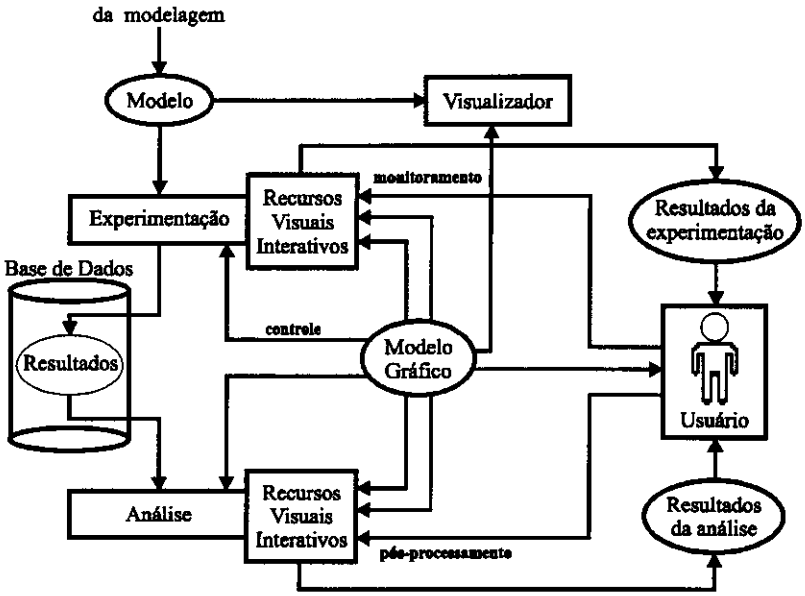


FIGURA 3.5 - Arquitetura de um ambiente de VIS.

Dentro dos aspectos de interação com o usuário, a interação a nível de experimento é o primeiro nível de interação desejado. Alguns dos recursos de interação que devem estar disponíveis ao usuário neste nível são: seleção do modo de execução da simulação (passo a passo, por quantidade de tempo, por tempo específico ou com interrupção até que se atinja algum ponto ou condição de parada), ativação e desativação do controle sobre o monitoramento de variáveis, alteração de parâmetros da simulação entre duas execuções, alteração das estatísticas exibidas e das representações gráficas utilizadas para as entidades do modelo permitindo uma melhor análise da execução, operações de salvamento e carga da simulação como um todo para o controle do usuário quando ele está realizando uma série de modificações durante a experimentação. Considerando estes recursos e as necessidades dos usuários no processo de análise da simulação, Freitas [FRE93] identifica as seguintes tarefas básicas: navegação pelo universo de objetos; seleção de um conjunto de objetos ou parte de um objeto; consulta a atributos de objetos; preparação de processamento (modificação de atributos, inicialização de parâmetros); processamento (simulação, cálculos, monitoramento, controle) e anotação de textos e armazenamento de dados.

Outro ponto muito importante durante a fase de experimentação são os recursos visuais disponíveis. Freitas [FRE93] apresenta uma classificação de ferramentas que são coleções de recursos visuais e interativos que suportam a realização das tarefas básicas identificadas na simulação. Estas ferramentas podem ser classificadas como de mapeamento, exploração, preparação e controle de processamento e registro. As ferramentas de mapeamento correspondem às facilidades oferecidas ao usuário para a associação de entidades e atributos a representações visuais. Diversas representações visuais podem ser utilizadas, tais como: ícones, diagramas, gráficos, tabelas, modelos geométricos, mapas e seqüências. Para que a análise dos resultados seja eficiente, é necessário que a escolha da representação visual para uma entidade ou atributo seja realizada de forma criteriosa.

Em relação a isto, Freitas [FRE93] apresenta uma metodologia para seleção de representações visuais. As ferramentas de exploração correspondem a facilidades de suporte a navegação, seleção e consulta dos objetos em estudo. A navegação permite observar um objeto das mais diversas posições (vistas) possíveis, possibilitando um maior entendimento do mesmo por parte do usuário. A seleção de um conjunto de entidades tem por objetivo diminuir o volume de dados a ser analisado. As consultas permitem que sejam observados os valores dos objetos indicados. As ferramentas de preparação e controle de processamento permitem uma análise das informações existentes no modelo, através do estudo das entidades do modelo quando da variação dos parâmetros do modelo. As ferramentas de mapeamento, navegação, seleção e consulta são recursos destinados ao monitoramento da simulação. Para controlar a simulação, são necessários recursos que permitam o controle da execução dos experimentos, como os implementados por Lindstaedt [LIN95]. Finalmente, as ferramentas da classe registro fornecem os recursos para a anotação de textos em representações visuais e armazenamento de imagens, para análise posterior ou para apresentação.

Os recursos que permitem o controle da execução da simulação são recursos que permitem uma interação entre o usuário e o modelo de simulação. Deste modo, independentemente da forma como foi construído o modelo de simulação (de forma textual ou através de recursos gráficos), é desejável que ele seja visualizado graficamente para uma melhor interação com o usuário. Através de recursos de navegação e de seleção sobre a representação gráfica do modelo, o usuário pode alterar os seus parâmetros, alterar os valores de variáveis e atributos de entidades, modificar uma determinada distribuição, etc. Esta interação do usuário com o modelo, entretanto, não permite que sejam realizadas uma alterações como, por exemplo, remoção de uma entidade ou alteração de uma conexão; somente é permitida a interação com parâmetros e variáveis. Uma forma alternativa para a visualização do modelo de simulação é a empregada em VISE [LIN95]. Todos os objetos (variáveis e entidades) passíveis de alteração no transcorrer da experimentação são apresentados em diversas listas. Desta forma, através da seleção de um objeto em uma das listas o usuário pode modificar seu valor e/ou atributo.

A análise dos resultados da simulação é uma área que está sendo bastante estudada. Com simulações de natureza estocástica, esta área é essencial para o entendimento da dinâmica do modelo de simulação e o relacionamento de seus componentes, através de uma correta análise estatística e interpretação dos resultados [LAW91]. De acordo com a caracterização dos dados pelo analista, diversas abordagens tem sido apresentadas para a análise dos resultados da simulação [SEI91], tendo em comum a utilização de um "batch" para a análise. Os dados são analisados em pós-processamento, ou seja, após o término da simulação, considerando que os parâmetros e as condições do modelo de simulação não mudam ao longo da simulação. Diversas ferramentas tem sido desenvolvidas para a análise dos resultados. Um exemplo é SIMSTAT [BAN94], que é uma ferramenta gráfica interativa para análise estatística nos dados de entrada e saída da simulação.

A interação a nível de pós-processamento permite que o usuário faça o controle da visualização dos resultados da simulação em regime de pós-

processamento, ou seja, somente após o término da simulação. Dentro deste nível de interação, o usuário pode selecionar variáveis a serem observadas e avançar e retroceder o tempo de simulação para observar quadro a quadro as informações apresentadas. Este avanço e retrocesso no tempo de simulação permite que o usuário analise diversas vezes os resultados até que ele chegue a um entendimento dos mesmos. Para que esta facilidade esteja disponível no ambiente é necessário que o mesmo permita que os dados sejam coletados para pós-processamento durante a execução da simulação. Esta coleta de resultados deve ser realizada em intervalos de tempo determinados.

Para facilitar a análise de fenômenos contínuos e dados complexos, Ellson & Cox [ELL88] preconizam a utilização de glifos (símbolos utilizados na comunicação visual) e técnicas de computação gráfica, como controle de cor e movimento, para que os resultados da simulação possam ser exibidos de forma mais eficiente. Alguns princípios são apresentados para a construção de visualizações científicas: **(a)** usar somente a informação importante: selecionar somente as variáveis e entidades fundamentais para o entendimento do problema e, exibir somente elas; **(b)** usar a forma do símbolo para representar grandezas vetoriais: para representar a informação direcional podem ser utilizadas setas ou formas tridimensionais; já a informação de magnitude pode ser representada através da cor ou do tamanho do símbolo; **(c)** usar a cor para representar grandezas escalares: a escolha do conjunto de cores é crucial, pois as alterações de cores na apresentação devem refletir com coerência às alterações de valores nas variáveis analisadas; **(d)** usar a animação para representar evolução no tempo: se não puder ser em tempo real, gravar as imagens para exibição posterior; **(e)** usar técnicas de realismo para representar a transição de um estado para outro. Claramente estas colocações apresentadas pelos autores enquadram estas características como uma técnica de visualização em pós-processamento, podendo ser enquadrada na melhor das hipóteses, em alguns casos, como monitoramento.

Já Post et al. [POS94] apresentam três diferentes técnicas aplicadas à visualização de campos vetoriais: visualização seletiva, visualização estatística e "turbulent particles". Estas técnicas estão relacionadas com a redução e simplificação dos dados devido a sua complexidade na simulação. As duas primeiras técnicas não são restritas a vizualizações de campos vetoriais e tem por finalidade a redução dos dados a serem visualizados, tornando a visualização mais clara e precisa. A primeira técnica habilita o usuário a selecionar partes de interesse dos dados a serem analisados. A segunda visualiza quantidades estatísticas somente sobre a região selecionada pelo usuário e não sobre o conjunto total dos dados. A terceira técnica é dedicada a uma classe particular de problema de dinâmica dos fluidos e utiliza partículas para exibir movimentos errôneos causados pela turbulência.

Generalizando as duas primeiras técnicas de visualização propostas, a **visualização seletiva** é uma abordagem que permite ao usuário a seleção de uma parte dos dados gerados, visando uma diminuição dos mesmos para uma análise direta. Isto significa que a massa de dados é submetida a um critério seletivo e a partir dele são extraídos apenas os dados que satisfazem as restrições impostas pelo usuário. Esta técnica deve estar disponível para o usuário através uma ferramenta à nível de

pós-processamento, de modo que ele possa selecionar alguns aspectos do modelo em que ele está interessado, tais como entidades e atributos. Desta forma, a análise é realizada somente sobre estes aspectos do modelo e não sobre todo o modelo, permitindo uma melhor visualização dos dados. Springmeyer [SPR92] chama de "data culling" esta tarefa de selecionar um conjunto de entidades ou parte da massa de dados. Após a tarefa de seleção, a técnica de **visualização estatística** utiliza estatísticas sobre os dados selecionados através da visualização de ícones. É permitido que o usuário defina o tipo de representação gráfica utilizada para cada estatística. Deste modo, com o auxílio destas duas técnicas o usuário pode, através de uma visualização mais pontual e mais simplificada sobre determinados aspectos, aumentar a sua compreensão do modelo.

Estes recursos de interação, controle e visualização suportam diversas fases do processo de simulação: verificação e validação do modelo, experimentação e análise de resultados. Os recursos oferecidos agilizam a especificação da visualização do comportamento do modelo facilitando a validação do mesmo. Também através destes recursos, a análise da sensibilidade do modelo em relação à alteração de certos parâmetros é aumentada. A fase de experimentação também é beneficiada pelo maior controle que se tem sobre a execução da simulação permitindo que o usuário, através da análise da execução, possa interferir em determinados aspectos do experimento, modificando desta forma o andamento da execução. Na fase de análise de resultados, as facilidades oferecidas possibilitam uma melhor interpretação dos resultados, pois tornam possível um acompanhamento mais efetivo de alterações nos estados de variáveis e atributos de entidades do modelo.

4 Modelagem interativa visual

4.1 Introdução

A análise de sistemas através de simulação está baseada na modelagem do sistema a ser simulado utilizando uma linguagem (de simulação ou de programação), e na submissão desse modelo a experimentos com diferentes parâmetros. A simulação interativa visual surgiu com o propósito de facilitar a condução dos experimentos de simulação pelo usuário, possibilitando a visualização de dados intermediários e controle completo do experimento e de parâmetros do modelo. Os conceitos de interatividade apresentados em VIS começaram a ser propagados com naturalidade para todas as fases do processo de simulação. Deste modo, o processo de modelagem do sistema começou a fazer uso destas características. Simultaneamente, a comunidade de Pesquisa Operacional, devido ao aumento no poder de processamento e de recursos gráficos dos computadores, começou uma nova tendência no desenvolvimento dos modelos de simulação. Ela consiste em primeiro desenvolver a visualização da representação gráfica do sistema em estudo. Esta abordagem ficou conhecida como **modelagem interativa visual** (VIM - Visual Interactive Modeling) [HUR86, ELD91]. Desta forma, VIM se refere à construção do modelo de simulação com o auxílio de um ambiente gráfico-interativo.

Segundo Elder [ELD91], VIS está para VIM assim como simulação está para modelagem. Enquanto que VIS se concentra nas necessidades do usuário do modelo, VIM concentra-se no construtor do modelo de simulação. O aspecto principal de VIM, o qual a diferencia da modelagem tradicional, é que ela incorpora recursos gráficos para modelar o sistema em estudo. O modelo gráfico permite que tanto o usuário quanto o modelador tenham uma maior flexibilidade no modelo e possam explorar de uma forma mais dinâmica o sistema, ganhando um melhor entendimento das características do mesmo. Com isto, os usuários começam a ter uma maior confiança no uso do modelo, pois são capazes de criticá-lo com mais segurança contribuindo desta forma para sua validação. A principal contribuição de VIM é a sua habilidade de reduzir as barreiras na comunicação entre o modelador e o usuário do modelo, uma vez que para um usuário leigo fica mais fácil o entendimento do sistema quando este está descrito de uma forma visual do que descrito através de uma linguagem de programação qualquer.

Através de VIM o usuário pode rapidamente criar diferentes cenários do modelo para o mesmo sistema, permitindo que ele julgue com mais confiança as características do modelo e do sistema. Este processo dá ao usuário um maior conhecimento do sistema em estudo. A idéia inicial da construção gráfica do modelo (cenário) nunca esteve disassociada de uma futura animação do mesmo. Hurrion [HUR86] afirma que VIM consiste nos seguintes passos: 1) desenvolvimento do modelo do sistema em estudo; 2) incorporação de um método de animação para o modelo; e, 3) interação com o modelo com o propósito de explorar estratégias

alternativas. A habilidade de ter a representação visual do modelo e a possibilidade de interagir com ela são os dois componentes principais da modelagem interativa visual.

A construção do modelo de simulação se tornou uma tarefa mais fácil desde a introdução de VIM. Entretanto a verificação, validação e experimentação sobre estes modelos requer um pouco mais de cuidados. Tobias [TOB91] apresenta alguns aspectos neste sentido, mostrando um estudo de caso de um serviço de reparos de carros.

Um modelo interativo visual combina gráficos significativos com fáceis interações para estimular a criatividade e compreensão [ELD91]. A promoção do processo de "gerar e testar" facilita de uma forma mais rápida o ciclo de aprendizagem. No desenvolvimento de modelos visuais interativos, alguns princípios já são aceitos. Este princípios são provenientes de evidências constatadas e não de experimentações rigorosas [BEL87]. Eles são:

(1) o usuário do modelo deve participar da elaboração das representações gráficas e da especificação da interface;

(2) a representação gráfica do modelo deve estar disponível mesmo antes da especificação do modelo matemático;

(3) tornar a interação a mais genérica possível, a fim de não limitar as questões que o usuário possa vir a fazer sobre o modelo;

(4) transferir o uso do modelo diretamente para o usuário final.

O processo de VIM pode ser dividido em duas partes: a construção do modelo de simulação e o projeto do cenário. Ambos podem ser armazenados em uma base de dados para o desenvolvimento de futuros modelos. Com o auxílio de uma ferramenta visual interativa, o usuário deve ser capaz de desenvolver o modelo de simulação através da descrição de sua estrutura e de seu comportamento. Na descrição da estrutura do modelo, geralmente através de um editor gráfico-interativo, são informados quais são os tipos de entidades do modelo, seus atributos e os relacionamentos entre eles. O usuário também deve ser provido de um editor gráfico e/ou uma linguagem gráfica que permita a descrição do comportamento das entidades presentes no modelo, ou seja, como elas se comportam ao longo do tempo. Independentemente de como o modelo de simulação é construído (forma textual ou através de recursos gráficos), o sistema deve prover uma ferramenta altamente amigável e composta de diversos recursos gráficos que permita ao usuário projetar o cenário que será utilizado para a apresentação de resultados. Através de operações de seleção, posicionamento e conexão da representação gráfica dos objetos (atributo especificado na descrição da estrutura do modelo), o usuário pode construir o cenário desejado.

4.2 Ambiente de modelagem

Quando Hurrion [HUR76] apud [BEL87] apresentou pela primeira vez o termo "visual interactive simulation", seu propósito era o de visualizar os resultados intermediários e possibilitar interação com o modelo para modificação de parâmetros

e variáveis. Até este momento, não era mencionada explicitamente a construção do modelo de simulação utilizando-se de recursos gráficos e interativos. Atualmente diversos ambientes de modelagem tem sido desenvolvidos, sendo constituídos por inúmeras ferramentas que auxiliam o desenvolvimento de modelos de simulação [BAL92, PAU94]. O objetivo principal destes ambientes é aumentar a produtividade do modelador, com ênfase na construção do modelo de simulação através de recursos gráficos. Uma proposta de arquitetura típica de um ambiente de modelagem pode ser vista na figura 4.1.

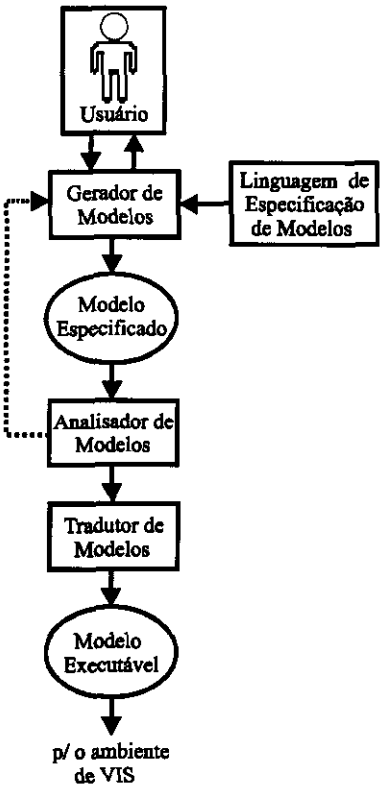


FIGURA 4.1 - Arquitetura de um ambiente de modelagem.

O módulo gerador de modelos permite que o modelador construa graficamente o modelo de simulação e a sua visualização para uma posterior análise por parte do usuário. Por intermédio de uma linguagem de especificação de modelos (LEM), o modelador especifica a lógica do modelo. O propósito principal da LEM é descrever o comportamento dos elementos do sistema modelado. Diversas técnicas diagramáticas, tais como diagramas ciclo-atividade e redes de Petri têm sido utilizadas para a especificação de modelos [SCH92, KIE94]. O módulo gerador de modelos juntamente com a linguagem de especificação de modelos constituem no processo de VIM. O módulo gerador de modelos é o ponto mais estudado hoje em dia em um ambiente de modelagem e será tratado com mais detalhes a seguir.

Após a construção do modelo de simulação, o módulo analisador de modelos verifica a correção da especificação do modelo gerada. Caso a especificação do modelo seja assegurada pelo analisador de modelos, ela é passada ao tradutor de modelos que a transforma em uma representação executável. Normalmente este modelo executável é representado através de uma linguagem de programação, como C ou Pascal. A partir deste momento, o ambiente de VIS pode ser ativado para realizar os experimentos sobre o modelo construído.

4.3 Linguagens visuais de programação

A maioria dos modelos de simulação são programas escritos em alguma linguagem, seja de simulação ou de propósitos gerais. Entretanto, com o avanço dos recursos gráficos, em muitos sistemas atuais o modelo de simulação é construído através de uma linguagem visual [CHA87, MYE90]. Tais linguagens são geralmente providas por editores gráficos diagramáticos. Deste modo, tanto para a visualização de resultados como para a construção do modelo, é possível analisar as linguagens e sistemas de simulação do ponto de vista das linguagens visuais.

A noção de linguagem visual de simulação leva ao exame das abordagens adotadas para linguagens visuais de programação (VPL - Visual Programming Languages). Por linguagens visuais de programação (ou linguagens gráficas de programação) entende-se a classe de linguagens que se vale de símbolos gráficos tanto para representar como para construir programas. Myers [MYE90] adota uma taxonomia que divide as linguagens visuais em **programação gráfica** e **visualização de programas**. A primeira se refere aos sistemas que utilizam técnicas de computação gráfica na construção de programas, enquanto que na segunda, estão os sistemas que utilizam técnicas de computação gráfica para visualizar programas estática ou dinamicamente, conforme a visualização seja realizada antes ou durante a execução do programa. Dentro de visualização de programas, Myers faz a separação entre visualização de código, visualização de estruturas de dados e visualização de algoritmos. Neste ponto, é importante salientar a distinção entre as duas últimas formas de visualização. Na visualização de dados a representação gráfica está associada aos dados, de forma que quando ocorrem alterações o sistema as exibe automaticamente. Já na visualização de algoritmos, o programador é responsável pela exibição das alterações produzidas nos dados através da escrita de código extra no programa.

Freitas [FRE92] propõe a seguinte classificação das linguagens visuais, considerando o modelo subjacente à linguagem visual (construção visual):

a) **icônicas** - quando o usuário utiliza ícones para visualizar e/ou manipular as construções;

b) **diagramáticas** - quando a linguagem utilizada é baseada em diagramas. É a mais utilizada para a construção de aplicativos que utilizam o paradigma "data-flow", onde podem ser incluídos os sistemas de visualização AVS [UPS89], ApE [DYE90] e Iris Explorer [SIL91];

c) **tabulares** - quando a linguagem faz uso do preenchimento de tabelas ou formulários para a construção de sentenças;

d) **demonstracionais** - quando o usuário exemplifica (demonstra) as construções para o sistema através de técnicas interativas.

Como salientado por Freitas [FRE94b], não se pode afirmar que uma abordagem reune mais condições que as outras de ser adotada mais amplamente. Para situações específicas é interessante utilizar uma das quatro abordagens, já para um ambiente de programação de propósitos gerais pode-se adotar o uso simultâneo de várias abordagens (ambientes "multiparadigma" [REI87, BOR90]). A abordagem icônica é a mais empregada para a visualização de estruturas de dados (inclusive com animação), enquanto que a diagramática para a visualização da estrutura de programas.

4.3.1 Programação Gráfica

Analisando os sistemas apresentados no capítulo anterior, pode-se notar que muitos apresentam tanto as facilidades de programação gráfica como de visualização de dados e algoritmos. Em VIS o objetivo principal é a observação do estado interno do modelo de simulação durante a realização dos experimentos e a visualização dos resultados de sua execução. Desta forma, VIS está relacionado com o uso das técnicas de visualização de dados e de algoritmos. Já em VIM, quando existe construção visual interativa do modelo de simulação, esta linguagem está enquadrada como programação gráfica. Neste caso, podem ser citados o sistema IMDE [OZD91], o ambiente ISI [NOL91], o ambiente GPVSS [BIS90] e algumas ferramentas de sistemas de simulação comerciais tais como o pacote Blocks para a construção de modelos SIMAN e os sistemas Slamsystem 2.0 e Tess para a construção de modelos em SLAM.

Segundo Ozden [OZD91], as abordagens de programação gráfica utilizadas pelas linguagens de simulação são:

a) redes e diagramas de blocos: são usados em linguagens de simulação convencionais como GPSS, SLAM e SIMAN. Aqui as atividades das entidades são descritas por uma seqüência de blocos ou redes de nodos;

b) ícones, cardápios, formulários (tabelas) e janelas: são incorporados às interfaces de linguagens de programação com a finalidade de termos o ambiente de programação mais amigável. Algumas linguagens empregam esta forma de interface onde através de uma navegação interativa o código pode ser modificado. Na simulação, este estilo de programação tem sido aplicado em áreas específicas como redes de filas para representar, por exemplo, sistemas de manufatura.

c) diálogos e menus estruturados em árvore: a programação baseada em diálogos é originada da idéia de que todo o programa de simulação tem uma especificação estruturada do modelo independente da área de aplicação. Por esta razão, um diálogo estruturado genérico com o usuário pode ser preparado antecipadamente para obter as informações necessárias para qualquer modelo de simulação. Uma forma diferente de diálogos é baseada em um conjunto de menus

estruturados na forma de uma árvore. O usuário escolhe um caminho para a especificação do modelo, iniciando a partir da raiz da árvore em direção aos níveis mais baixos (folhas) através de suas escolhas a partir dos menus.

A metodologia de programação gráfica em um ambiente de simulação deve trazer as seguintes vantagens: facilitar o uso do ambiente de simulação; aumentar a produtividade do modelador; minimizar os erros de programação; aumentar a visualização do problema; servir como uma boa comunicação para as pessoas. Desta forma, os seguintes recursos (facilidades) devem estar disponíveis:

a) um editor gráfico: utilizado para a criação de novas classes de objetos com sua representação gráfica. Estas podem ser armazenadas em uma biblioteca de simulação, possibilitando a alteração da representação gráfica do objeto e até mesmo a criação de um novo objeto a partir de um já existente. Para uma maior facilidade de acesso aos objetos e sua representação gráfica, as consultas a esta biblioteca poderiam ser realizadas através de um navegador gráfico e hierárquico.

b) um construtor e interpretador gráfico de modelos: utilizado para construir graficamente os modelos de simulação, através do relacionamento dos objetos criados no editor gráfico. Esta construção basicamente seria o posicionamento da representação gráfica dos objetos em um cenário e a conexão entre os mesmos. A seguir, esta representação gráfica do modelo seria traduzida para uma linguagem reconhecida pelo simulador.

c) um construtor de visualização: o modelo criado em b) pode ser visualizado graficamente durante a execução da simulação. Seria interessante que pudessem ser criadas diferentes janelas de visualização sobre diferentes aspectos do modelo para facilitar a verificação do mesmo.

4.4 Modelagem gráfica hierárquica

Freqüentemente se um modelo é complexo, ele pode ser particionado em uma rede de submodelos interconectados [GOR90]. Por sua vez, cada submodelo pode ser representado por uma nova rede de submodelos ou pode conter elementos básicos do modelo de simulação. Esta modelagem introduz hierarquia na descrição dos sistemas. A estratégia de modelagem hierárquica permite a construção do modelo de simulação através de duas técnicas básicas [CER94]: abstração (agregação) e refinamento. A abstração é utilizada para a agregação de um número de elementos do modelo de níveis mais baixos em um único elemento de nível mais alto. Isto permite uma simplificação na complexidade da estrutura do modelo e é a base da estratégia "bottom-up" (primeiro é criada uma biblioteca de submodelos e depois eles são unidos para formar o modelo completo). Refinamento é a técnica de desenvolver uma descrição detalhada de um simples bloco de modelagem num nível hierárquico mais alto. Isto permite desenvolver um modelo começando com uma descrição rudimentar do sistema, sendo a base para a estratégia "top-down" (primeiro é criado o modelo de mais alto nível e progressivamente vai substituindo os blocos por submodelos

detalhados). Embora a estratégia "top-down" aparente ser a mais poderosa e usada estratégia de construção de modelos, ambas "bottom-up" e a combinação das duas estratégias também são bastante utilizadas.

Dentre as diversas vantagens da abordagem de modelagem hierárquica, podem ser citadas:

a) tornar mais clara a estrutura do modelo (quando um modelo representa um sistema muito complexo, o mesmo pode ser particionado em diversos submodelos para facilitar seu entendimento e depois através de uma metodologia "top-down" o modelador pode refinar cada um dos submodelos e adicionar mais conexões entre os mesmos e mesmo definir novos submodelos dentro dele);

b) evitar as repetições (um dado submodelo pode ocorrer diversas vezes no mesmo modelo);

c) modularidade (cada submodelo pode ser testado independentemente e também podem ser definidos com parâmetros, os quais podem receber valores diferentes a cada chamada. Através desta variação nos parâmetros, pode-se alterar o número de elementos no modelo sem a necessidade de recompilar o modelo);

d) habilita o ocultamento de informações, permitindo que informações locais aos módulos não sejam identificadas pelos níveis superiores. Desta forma, se houverem alterações na descrição dos módulos nos níveis mais baixos, os módulos nos níveis superiores continuam intactos.

e) facilita as mudanças no modelo e a depuração/validação (o processo de modularidade permite que cada módulo seja depurado e validado em separado);

f) facilita o processo de apresentação de resultados e operação do modelo (através da abordagem hierárquica, os resultados e a operação podem ser apresentados em diferentes níveis de detalhes, dependendo do público alvo).

A modelagem hierárquica se torna mais poderosa quando da utilização de recursos gráficos e interativos. Um ambiente gráfico que suporte uma modelagem hierárquica deve possuir recursos capazes de: criar, visualizar, editar e armazenar cada submodelo; ligar graficamente submodelos com outros elementos de modelagem, inclusive outras instâncias de submodelos; substituir graficamente submodelos por outros elementos de modelagem; visualizar os resultados, incluindo animação, para todo o modelo e para cada instância de um submodelo; e associar ícones aos submodelos criados (um editor de ícones também deveria estar disponível). Através do posicionamento deste ícone na representação gráfica do modelo, é realizada uma chamada ao submodelo correspondente (é criada uma instância do mesmo), que está armazenado em uma base de dados. Este ambiente gráfico deve ser composto por um "browser" para permitir que o modelador "navegue" sobre a hierarquia do modelo, podendo visualizar e alterar detalhes nos níveis hierárquicos. Um ambiente com múltiplas janelas permitiria que o modelador visualizasse diferentes submodelos em diferentes janelas. Também deve estar à disposição do modelador, quando da criação de uma instância de um submodelo, a especificação dos atributos textuais tais como nome do submodelo, nome dos parâmetros e seus valores, etc. Como exemplo de ambientes gráficos que suportam a construção do

modelo de simulação de uma forma hierárquica, podem ser citados: ISI [NOL91], RESQME[GOR90] e REDES[WAG88].

4.5 Detecção de erros na modelagem gráfica

Atualmente muitos ambientes de simulação que suportam recursos de modelagem são gráficos e interativos. Desta forma através da manipulação de diagramas gráficos o usuário pode construir o modelo de simulação. Normalmente estes ambientes de modelagem fornecem ao usuário uma interface dirigida por cardápios e uma "palette" de ícones, onde o usuário constrói o modelo através da seleção, posicionamento e conexão destes ícones. Adicionalmente, algumas informações textuais são necessárias para a especificação dos atributos do modelo e submodelos, caso seja um modelo hierárquico. A combinação de formas gráficas e textuais fazem com que certas regras sejam cumpridas para que o modelo esteja corretamente especificado (as informações textuais devem ser verificadas sintatica e semanticamente). Alguns tipos de erros, durante a construção do modelo, podem ser completamente eliminados através deste tipo de interface gráfica, enquanto que outros precisam ser detectados pelo modelador durante a construção do mesmo [GOR91]. Claro que, mesmo criando o modelo graficamente, muitas informações são fornecidas textualmente, tais como: atribuir um nome a um objeto, declarar as variáveis necessárias, especificar alguns parâmetros para as distribuições, atribuir valores a variáveis e parâmetros, etc. Desta forma, sempre que algum tipo de informação textual é fornecida, deve-se verificar se ela está correta.

As interfaces gráficas permitem que, através da manipulação de ícones em um diagrama e seus correspondentes atributos, o modelador possa criar um modelo de simulação. Uma alternativa seria a especificação total do modelo textualmente em uma linguagem de simulação, onde para cada sentença se tem uma sintaxe adequada e, portanto, as chances de ocorrerem erros são bem maiores. Através da interface gráfica, muitas destas declarações podem ser substituídas por manipulações no diagrama, reduzindo desta forma os requerimentos de conhecimento de sintaxe pelo usuário. Com operações como seleção e apontamento, as interfaces gráficas podem substituir diversas entradas textuais. Por exemplo, erros de sintaxe podem ser eliminados quando se realizam conexões entre elementos, bastando para isto somente o apontamento dos nodos de origem e destino no diagrama. Além disto, o ambiente pode verificar se esta conexão está semanticamente correta baseando-se nos tipos de nodos envolvidos.

Outra forma onde a interface pode auxiliar o usuário é quando da necessidade de fornecer entradas específicas para um determinado objeto. Através de um apontamento no objeto desejado, o usuário teria a sua disposição uma janela com todos os atributos deste objeto para que um valor fosse atribuído a cada um dos seus atributos. Este tipo de recurso evita que o usuário especifique um atributo erradamente e permite que ele tenha uma visão global do estado de cada um dos atributos. Outro exemplo, ainda dentro desta idéia de exibir as informações de um objeto selecionado em uma janela, seria a alteração de um determinado parâmetro de

um submodelo. Para isto, bastaria ao usuário selecionar o ícone correspondente ao submodelo e uma janela com todos seus parâmetros e valores correspondentes seria exibida para uma possível alteração.

Um outro recurso interessante que deve ser previsto em ambientes que permitem a construção do modelo graficamente é a exibição e localização do erro para o modelador. Em descrições puramente textuais não haveria problema nenhum, pois a especificação do modelo é submetida a um compilador que a verifica sintatica e semanticamente. Qualquer erro encontrado é exibido em uma lista seqüencial que indica o tipo de erro e qual a sua localização na especificação, de forma que através de um editor de texto o modelador pode efetuar as devidas correções. Em ambientes gráficos este processo é um pouco mais complicado, por duas razões. Em primeiro lugar, a representação gráfica do modelo pode não ser totalmente visível (a ferramenta deve possuir recursos de "scroll") o que dificulta a localização do erro. Isto fica ainda mais complicado se o modelo for hierárquico (a ferramenta deve possuir o recurso de navegação). Em segundo lugar, por ser um ambiente gráfico todos os atributos textuais ficam invisíveis. Para consultar um determinado atributo de uma entidade do modelo, deve-se selecionar a representação gráfica (ícone) desta entidade. Esta forma, de localização do erro fica muito dispendiosa para o modelador.

Para contornar este problema de localização de erros e explorando a natureza gráfica do modelo, poderiam estar disponíveis recursos visuais que permitissem a rápida localização do erro. Por exemplo, se um determinado atributo textual de uma entidade está incorreto, o ícone associado a esta entidade pode ser exibido com uma cor diferenciada que indique erro (por exemplo vermelho), o mesmo vale para um erro na definição de um caminho (conexão). Caso este erro seja cometido em um nível hierárquico qualquer, todo o caminho até a localização do erro deve ser exibido com esta cor de indicação de erro.

4.6 Ambientes gráficos de construção de modelos

O objetivo desta seção é descrever de forma resumida as características de alguns ambientes de simulação que permitem a construção gráfica do modelo. Os seguintes ambientes foram analisados durante o desenvolvimento deste trabalho: IMDE [OZD91] um ambiente orientado a objetos baseado em Smaltalk-80; GPVSS [BIS90] um ambiente que permite a construção e execução de modelos visuais; ISI [NOL91] um ambiente integrado com uma interface interativa visual para modelos em SIMAN; SIMFLEX [VUJ90] um ambiente de simulação orientado a objetos para sistemas de manufatura; e, SimView [DIJ96] um ambiente de modelagem interativa visual.

4.6.1 IMDE

Os princípios e conceitos de programação gráfica foram utilizados por Ozden [OZD91] no estudo e desenvolvimento do ambiente **IMDE** (Integrated Model

Development Environment). Este ambiente consiste de um conjunto de ferramentas que permitem a especificação, desenvolvimento e verificação do modelo em um ambiente orientado a objetos baseado na linguagem Smaltalk-80. O modelador utiliza-se de um formalismo gráfico baseado em diagramas de ciclo-atividade (ACD - Activity-Cycle Diagrams) para a definição do modelo de simulação de forma gráfica e interativa. Esta descrição pode ser interpretada pelo sistema e executada automaticamente, fazendo com que o modelador não necessite ter conhecimento sobre a linguagem Smaltalk-80. Os ACDs são redes de atividades que possuem características especiais para representar conceitos de simulação, tais como sincronização de atividades, produção e utilização de recursos. Estes diagramas possuem um alto nível gráfico de abstração e são independentes da linguagem utilizada, sendo desta forma uma poderosa representação gráfica para modelos de simulação.

Dentro do ambiente existe a ferramenta GraphSIM (Graphical Simulation Modeling) que permite construir interativamente um diagrama ciclo-atividade. Como o ambiente foi construído utilizando-se da linguagem Smaltalk e esta utiliza-se dos botões do mouse para a programação, as ferramentas do ambiente se valem dos botões do mouse para a sua comunicação. GraphSIM é composto por três áreas (janelas) principais: a área superior contém os comandos para controle da execução de um modelo de simulação; a área intermediária é utilizada para a construção interativa de ACDs e é composta por quatro janelas contendo os objetos de simulação, os recursos, as atividades e os elementos das atividades; a área inferior, a maior das três áreas, é utilizada para visualizar a representação gráfica do ACD.

A construção gráfica de um ACD em GraphSIM inicia pela definição dos objetos de simulação na respectiva janela. Para cada objeto é informado o seu nome e a definição se ele é permanente ou temporário, deste modo é apresentada uma lista na janela de objetos com o nome de todos os objetos de simulação criados. O passo seguinte é a definição dos recursos, onde para cada um o usuário deve informar o nome e a quantidade. A seguir, deve ser realizada a definição dos processos dos objetos de simulação. Isto é realizado através do apontamento de um objeto na janela de objetos, onde o usuário informa a descrição das atividades para este objeto selecionado. Após a definição dos atributos de todas as atividades de um objeto, devem ser identificadas as conexões entre estas atividades. Atualmente nesta versão de GraphSIM, somente podem ser realizadas conexões seqüenciais.

Para a construção interativa do ACD, o usuário deve realizar um apontamento com o mouse em uma das quatro janelas. Para cada janela é exibido um cardápio específico contendo várias opções para a construção dos elementos do ACD. Após todas estas definições, sempre que um objeto de simulação é selecionado na janela de objetos, o seu ACD é apresentado na janela de visualização gráfica. A partir deste momento, o usuário pode executar o modelo de simulação criado diretamente da interface de GraphSIM. Na área superior da interface o usuário encontra os comandos para iniciar e parar a simulação. Os resultados da simulação são exibidos em uma outra janela do sistema para a análise da simulação.

4.6.2 GPVSS

O ambiente de simulação **GPVSS** (General Purpose Visual Simulation System) [BIS90] é um ambiente que auxilia ao usuário na construção e execução de modelos visuais para simulação. Este ambiente possui os seguintes recursos: criar e visualizar graficamente o modelo, especificar interativamente a lógica do modelo e gerar uma versão executável do modelo na linguagem de programação C. O ambiente possui uma interface com múltiplas janelas dirigida por "pop-up menus", ícones e botões sensíveis ao teclado e ao mouse. Os dados e os componentes do modelo são armazenados em uma base de dados relacional INGRES. GPVSS contém 3 módulos principais: gerador de modelos, translação do modelo e simulação visual. Através do módulo gerador de modelos, pode-se criar graficamente o modelo e a sua forma de visualização, definir submodelos e sua lógica e especificar objetos e atributos. O módulo de translação do modelo permite que a especificação visual do modelo, criada pelo gerador de modelos, seja transformada em simulação executável na linguagem de programação C. O módulo da simulação visual é usado para executar a simulação utilizando ou não representação gráfica para a apresentação dos resultados. Cada um destes módulos é ativado através da seleção de um ícone específico, localizado na interface principal do ambiente.

Dentro do ambiente GPVSS, o módulo gerador de modelos é o responsável pela criação do modelo de simulação. Quando este gerador de modelos é ativado na interface icônica do ambiente, um editor gráfico contendo vários recursos torna-se disponível ao usuário para que ele projete a representação gráfica que será utilizada na apresentação de resultados. Dois tipos de representações gráficas devem ser criadas: as representações de "background" (estáticas) e as representações dos objetos (dinâmicas). Após a criação destas representações, elas devem ser associadas com o modelo. Esta operação é realizada através da ativação de um botão na interface que armazena estas imagens em arquivos específicos em um banco de dados relacional INGRES, onde ficam relacionadas estas representações com o modelo. Após este processo, o usuário tem a sua disposição o editor de modelos que é responsável pela definição dos submodelos e caminhos. A definição de um submodelo é realizada pela seleção das representações gráficas apresentadas na janela de visualização do modelo, agrupando-as através de uma envoltória. A partir deste momento, os caminhos que conectam os submodelos devem ser definidos pela indicação de quais submodelos que fazem parte deste caminho. Os caminhos em GPVSS são uni-direcionais, de tal forma que se um objeto dinâmico se move nos dois sentidos entre dois submodelos, o usuário deve especificar dois caminhos. O próximo passo é a especificação dos submodelos e objetos dinâmicos e seus atributos. Para a especificação de um submodelo são apresentadas várias subjanelas, cada uma com uma característica específica para a descrição da lógica do modelo, onde o usuário faz a especificação através de macros da linguagem de programação C. Para a especificação dos objetos dinâmicos, uma janela é apresentada contendo várias características que devem ser informadas. Finalmente, quando todos os submodelos e objetos dinâmicos forem especificados, estes são igualmente armazenados na base de dados.

4.6.3 ISI

Nolan [NOL91] utilizou SIMAN como base de um ambiente integrado e inteligente de simulação. Os autores reportam **ISI** (Inteligent Simulation Interface), uma interface interativa visual para SIMAN. ISI prevê a construção do modelo utilizando uma representação gráfica e, em separado, a especificação do experimento. A representação gráfica utilizada é uma espécie de fluxograma, onde blocos representam processos. A modelagem nesse sistema é de certa forma hierárquica: o usuário pode utilizar porções de modelo já desenvolvidas e armazenadas como macros. Macros podem ser agrupadas em bibliotecas diferentes, dependendo da aplicação, transformando ISI em interfaces de propósitos específicos.

Na ferramenta de especificação gráfica de ISI estão presentes todas as facilidades interativas convencionais como apontamento, seleção de componentes a partir de cardápios, operações com janelas, etc. Os parâmetros dos blocos tem seus valores atuais informados interativamente através de um cardápio "pop-up", onde pode ser seletivamente alterados. Quando a modelagem está completa, a descrição gráfica do sistema é utilizada em todas as etapas subsequentes. Esta representação e todas as outras informações produzidas ao longo da simulação são armazenadas numa base de conhecimentos.

A definição do experimento vem a seguir, através da especificação de parâmetros com disponibilidade de recursos, condições iniciais, tempo de execução, etc. Depois desta fase, código SIMAN é automaticamente gerado.

Duas formas de tratamento dos resultados da simulação estão disponíveis: animação em tempo real e pós-processamento. A animação em tempo real usa a mesma representação gráfica desenvolvida em tempo de modelagem; ícones representando entidades são movimentados entre blocos de fluxograma. O estado de cada bloco é atualizado dinamicamente. Gráficos de barras mostrando a disponibilidade de recursos em "buffers" são exibidos; variáveis podem ser observadas e ter seus valores alterados durante o andamento da simulação. Outras estatísticas podem ser coletadas e visualizadas dinamicamente. ISI fornece também a possibilidade de pós-processamento interativo, dirigido por cardápios, permitindo que os resultados da simulação sejam plotados numa variedade de formatos ou, ainda, exportados, para outros pacotes como Lotus 1-2-3. Formatos incluem gráficos de linha ou barras e histogramas que podem ser superpostos ou plotados separadamente.

4.6.4 SIMFLEX

SIMFLEX [VUJ90] é um ambiente de simulação orientado a objetos para sistemas flexíveis de manufatura (FMS - Flexible Manufacturing Systems). Os sistemas de manufatura são considerados como aplicações naturais considerando as características do paradigma de orientação a objetos. SIMFLEX foi desenvolvido utilizando-se das capacidades gráficas oferecidas em Smalltalk-80. Thomasma [THO90] apresenta uma visão geral de aplicações Smalltalk em simulação de

manufatura, onde alguns destes sistemas possuem características de VIS. Atualmente, um grande número de sistemas de manufatura tem sido baseados em aplicações Smalltalk. SIMFLEX utiliza técnicas de inteligência artificial para a modelagem da representação do conhecimento, onde o domínio da modelagem do conhecimento é codificado em uma rede semântica através do uso de NRL (Network Representation Language), uma aplicação desenvolvida em Smalltalk-80. Elzas [ELZ86] apresenta uma visão da aplicabilidade de técnicas de IA para representação do conhecimento em modelagem e simulação.

O primeiro passo em SIMFLEX é a definição de uma hierarquia de classes, a qual suporta implementação de capacidades de VIM e VIS. A aplicação MVC (Model-View-Controller) desenvolvida em Smalltalk-80 e contida em SIMFLEX, permite a construção de aplicações interativas através da definição do modelo, da representação gráfica e da interação do usuário. As informações sobre o modelo de simulação e as atividades realizadas durante a modelagem e o processo de simulação são armazenadas na classe *Model*. A representação da descrição gráfica do modelo de simulação e da animação utilizada durante a experimentação estão contidas na classe *FMSView*, que é uma subclasse da classe *Canvas*. Já a classe *FMSController* controla a interação do usuário.

A interface de SIMFLEX consiste de um conjunto de botões para a ativação das atividades de modelagem e simulação, uma janela para a exibição da representação do cenário, uma janela para a exibição da representação do modelo de simulação e, uma janela para a exibição dos resultados da simulação e mensagens ao usuário. Para o desenvolvimento do modelo de simulação o usuário é provido de um ambiente gráfico baseado em ícones, que permite a construção de suas representações do sistema: a) representação externa (visual), contendo um cenário baseado em ícones utilizado para a apresentação de resultados; b) representação interna (lógica), contendo um conjunto de objetos representando os componentes do sistema modelado, a qual constrói o modelo de simulação. SIMFLEX possui uma base de dados de simulação, desenvolvida em Smalltalk-80, que permite que o usuário armazene o modelo de simulação criado. Desta forma, o modelo pode ser utilizado para um outro experimento ou para a construção de outro modelo.

O processo de modelagem em SIMFLEX inicia com a definição de características gerais do sistema. Um determinado componente é modelado interativamente pela ativação de botões, onde o cenário define suas características e associa um ícone a ele. Este ícone deve ser posicionado no local correto dentro da janela de representação gráfica do sistema, de forma que esta representação possa ser utilizada mais tarde para animação. O modelo de simulação, propriamente dito, é gerado em paralelo. O usuário é inquirido somente para posicionar os nodos na representação visual da rede semântica.

4.6.5 SimView

O ambiente de modelagem interativa visual **SimView** [DIJ96] permite a especificação gráfica do modelo, animação e decomposição hierárquica para dar suporte ao processo de modelagem organizacional. A ênfase de SimView é no projeto da interface, como os modelos são especificados e visualizados, como eles são vistos durante a experimentação e o modo como são apresentados seus resultados. O propósito de SimView é auxiliar os modeladores a obter uma melhor compreensão do comportamento dinâmico de uma organização como um todo, incluindo interações com o ambiente, para poder tomar decisões observando mudanças organizacionais. Para ilustrar uma aplicação de SimView é apresentado um estudo sobre uma companhia ferroviária que transporta açúcar em um estado da Austrália.

Um modelo de simulação em SimView consiste de duas entidades elementares: os objetos são as entidades que permanecem em uma posição estática e não mudam ao longo do tempo; os ítens são as entidades que se movem através do sistema e podem mudar durante a simulação. Ítens são agrupados em classes, cada uma tendo sua própria representação gráfica (ícone), seqüência de interações (fluxos) e um conjunto de atributos. A modelagem do comportamento do sistema é alcançada através da construção visual dos fluxos dos ítens. Um caminho define uma seqüência simples de interações entre um item de uma determinada classe e um ou mais objetos no sistema. O caminho consiste de vários passos que são movimentos de um objeto para outro. Um fluxo é definido como a coleção de todos os caminhos para uma classe.

O processo de construção do modelo em SimView consiste de três fases: edição dos objetos, edição do fluxo e edição da interação. Normalmente estas fases são realizadas de uma forma seqüencial. Na edição dos objetos, os objetos que representam os componentes do sistema são escolhidos a partir de uma "palete" de objetos (ícones do tipo "bitmap") e são posicionados na tela. Uma grande quantidade de objetos são oferecidos e podem ser agrupados em três categorias: objetos de serviços, objetos de controle e objetos de modelagem. Para diferentes estados de um objeto (por exemplo, ocupado, indisponível ou livre) diferentes ícones podem ser associados. O uso de diferentes ícones para cada estado reflete as mudanças nas condições operacionais de um objeto e permite um rápido entendimento do que está ocorrendo no sistema. Na edição do fluxo, os caminhos entre os objetos de simulação são criados. Antes de adicionar os caminhos no modelo, as classes devem ser criadas. Cada classe tem seu ícone associado e os caminhos são criados para cada classe (diferentes classes não podem compartilhar o mesmo caminho). Na edição da interação, as interações para cada classe com os objetos que estão incluídos no seu fluxo são especificadas. Estas interações ocorrem quando os ítens se movem de um objeto para outro através do caminho.

Antes do modelo ser construído em SimView, ítens e objetos devem ser identificados no sistema real. Normalmente as entidades dentro do sistema que estão em uma posição estática são modeladas como objetos, enquanto que entradas e saídas do sistema são modeladas com ítens. Existe em SimView um tipo especial de objeto,

chamado "portal". Estes objetos são usados para modelar vários níveis de detalhes e representam um componente do sistema que é modelado em mais detalhes em um submodelo. A criação de submodelos forma um modelo hierárquico e permite que os modelos sejam compostos de várias janelas, cada uma contendo um submodelo.

4.7 Características para um ambiente de VIM

Considerando os diversos sistemas de simulação estudados que permitem a construção do modelo de simulação, pode-se chegar a algumas características desejáveis para um ambiente de VIM, a saber:

a) construção gráfica da estrutura do modelo;
b) construção gráfica/visual do cenário;
c) especificação do comportamento das entidades;
d) recursos de interação e visualização;
e) possibilidade de associar ícones às entidades do modelo;
f) possibilidade de armazenar o modelo em uma base de dados.

Com base nestas características, a figura 4.2 apresenta a arquitetura de um ambiente de VIM. A característica principal de um ambiente de VIM é que ele seja provido de um conjunto de ferramentas e recursos gráficos que permitam a construção do modelo de simulação. Dentro deste aspecto, a estrutura do modelo de simulação deve ser construída através de um editor gráfico. Tal editor permitiria descrever as entidades presentes no modelo e os relacionamentos existentes entre elas. Seria interessante que este editor gráfico utilizasse a abordagem da modelagem gráfica hierárquica, oferecendo ao modelador os recursos e vantagens mencionados na seção 4.4. Também através de um editor gráfico, o modelador constrói a representação visual (cenário) do modelo. No cenário são colocadas as representações estáticas (imagens de fundo) e as representações gráficas (dinâmicas) das entidades do modelo. Através de operações de seleção, posicionamento e conexão das entidades o modelador é capaz de representar visualmente o modelo. Por intermédio destes dois editores gráficos, o usuário somente constrói a representação gráfica do modelo (sua estrutura e apresentação). Também é necessário que se especifique a lógica do modelo, ou seja, o comportamento de cada uma das entidades descritas no modelo. Desta forma, deve-se ter um outro editor/linguagem gráfica para este propósito.

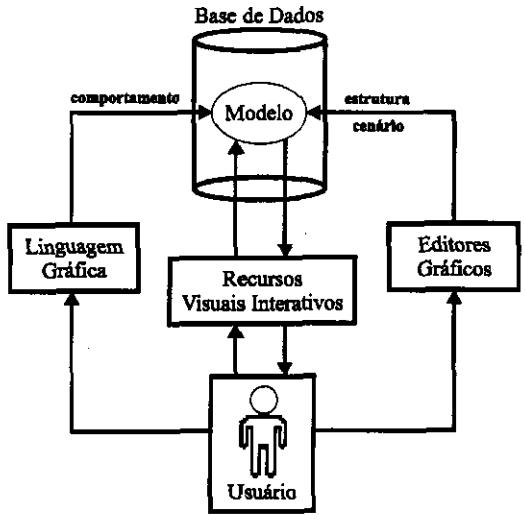


FIGURA 4.2 - Arquitetura de um ambiente de VIM.

A característica principal de um ambiente de VIM é que ele seja provido de um conjunto de ferramentas e recursos gráficos que permitam a construção do modelo de simulação. Dentro deste aspecto, a estrutura do modelo de simulação deve ser construída através de um editor gráfico. Tal editor permitiria descrever as entidades presentes no modelo e os relacionamentos existentes entre elas. Seria interessante que este editor gráfico utilizasse a abordagem da modelagem gráfica hierárquica, oferecendo ao modelador os recursos e vantagens mencionados na seção 4.4. Também através de um editor gráfico, o modelador constrói a representação visual (cenário) do modelo. No cenário são colocadas as representações estáticas (imagens de fundo) e as representações gráficas (dinâmicas) das entidades do modelo. Através de operações de seleção, posicionamento e conexão das entidades o modelador é capaz de representar visualmente o modelo. Por intermédio destes dois editores gráficos, o usuário somente constrói a representação gráfica do modelo (sua estrutura e apresentação). Também é necessário que se especifique a lógica do modelo, ou seja, o comportamento de cada uma das entidades descritas no modelo. Desta forma, deve-se ter um outro editor/linguagem gráfica para este propósito.

Outro aspecto muito importante em um ambiente de VIM são os recursos de interação e visualização disponibilizados para o usuário. Recursos de navegação sobre a representação gráfica do modelo (principalmente se for um modelo hierárquico) e de seleção de entidades para especificação de atributos são vitais para um ambiente de VIM. Também deve ser permitido que se especifique a forma de apresentação dos dados provenientes da simulação, tais como entidades, atributos, variáveis e estatísticas. Isto é muito importante durante a fase de experimentação para uma melhor compreensão do modelo que está sendo simulado. Deve-se garantir que a representação gráfica de uma determinada informação seja significativa, e deve-se permitir que o usuário faça inferências sobre ela. Diversas formas de representação gráfica devem estar disponíveis para o usuário, tais como: gráficos, histogramas, diais, relógios, etc.

Dentro dos recursos oferecidos ao usuário, deve haver a possibilidade de criar e associar ícones às entidades do modelo. A representação das entidades por ícones que lembrem seu significado no sistema real é de fundamental importância para a fase de apresentação de resultados. Para dar uma maior flexibilidade ao usuário, ele poderia criar seus próprios ícones através de um editor gráfico específico. Para cada entidade do modelo, o usuário pode associar um ícone criado neste editor gráfico ou associar ícones na forma de "bitmaps". Também seria interessante o uso abordagem hierárquica na modelagem proposta por Johnson [JOH88] onde, recursos de visualização distintos para uma mesma informação, com complexidades crescentes, são associados a diferentes fases do processo de simulação. Esta abordagem seria muito vantajosa para a construção do cenário, uma vez que um cenário com um realismo muito grande, comparando-se com o sistema real, somente deve ser construído quando da apresentação final dos resultados.

Finalmente, um ambiente de VIM deve ter a capacidade de armazenar o modelo, assim como todos os demais dados relacionados com o mesmo, em uma base de dados. Este recurso é muito importante, pois permite que o usuário realize várias experimentações sobre o modelo onde, a cada nova experimentação o usuário pode fazer diferentes alterações no modelo. Desta forma, o usuário pode através destas inúmeras experimentações compreender melhor o funcionamento do sistema que está sendo modelado. Também na construção de um novo modelo de simulação, pode-se fazer uso de diversos componentes de outros modelos previamente armazenados na base de dados. Esta característica seria melhor explorada caso o modelo fosse construído segundo a abordagem da modelagem gráfica hierárquica.

5 Uma abordagem integradora para VIS e VIM

5.1 Introdução

O termo VIM ainda encontra-se confuso na literatura de simulação e é muitas vezes absorvido por VIS, que tem sido utilizado genericamente para caracterizar os ambientes de simulação com recursos gráficos e interativos. Muitos autores classificam seus sistemas como ambientes de VIS, mas na realidade são ambientes que somente permitem a construção gráfica do modelo de simulação através de diversos recursos e depois simulam um executável deste modelo. Na verdade tais ambientes foram projetados para oferecer facilidades para a construção do modelo de simulação e não facilidades para o controle da experimentação e análise de resultados. Considerando tanto os aspectos de modelagem quanto de experimentação, os ambientes de simulação podem ser classificados dentro de três categorias: VIS, VIM e VIS+VIM.

Ambientes de VIS possuem interação com o usuário na fase de experimentação, possuindo uma série de recursos interativos e visuais. O modelo de simulação é construído separadamente (pode até ser através de recursos gráficos) e o ambiente de simulação obtém a descrição deste modelo para realizar a experimentação. A interação possível entre o usuário e o modelo durante a simulação é a de controle, o que quer dizer que o usuário não pode modificar o modelo de simulação, apenas interagir com seus parâmetros e variáveis.

Ambientes de VIM permitem a construção do modelo de simulação através de recursos visuais e interativos. Tais ambientes estão interessados em oferecer recursos para o modelador, não possuindo recursos para a fase de experimentação do modelo. Posteriormente, um ambiente de simulação utiliza a descrição do modelo gerada pelo ambiente de VIM para realizar a experimentação.

Ambientes de VIS+VIM possuem tanto os recursos de VIM como de VIS. Existe a interação do usuário com o modelo e a fase de experimentação, mas em momentos distintos. A experimentação é realizada sobre um modelo estático, ou seja, o usuário não pode modificá-lo, apenas alterar o conteúdo de variáveis, parâmetros e atributos de entidades. Caso haja a necessidade de alterar o modelo, deve-se interromper a experimentação, realizar as alterações no modelo e depois reiniciar a experimentação. Tais ambientes possuem, na realidade, dois "sub-ambientes" internos, um de VIS e outro de VIM, que não trabalham em paralelo, disponibilizando em sua interface todas as ferramentas necessárias para ambientes de modelagem e simulação.

5.2 A abordagem VISM

A idéia de uma nova abordagem, chamada de VISM (***Visual Interactive Simulation and Modeling***), vem da constatação da não existência de um ambiente integrado que possua recursos que permitam, em tempo de

simulação/experimentação, a interação do usuário tanto com o modelo como com o experimento. Um ambiente de VISM (figura 5.1) é um ambiente completo de modelagem e simulação, onde o usuário, através de múltiplas janelas, tem à sua disposição ferramentas que oferecem tanto recursos para a experimentação quanto para a construção e modificação do modelo de simulação, mesmo durante a experimentação.

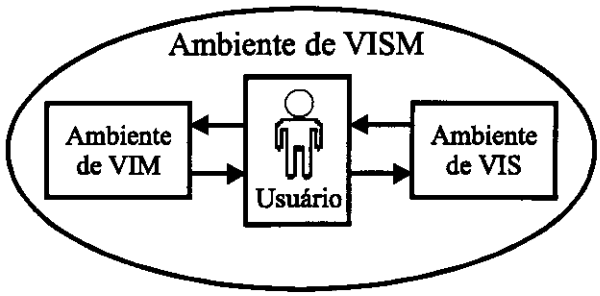


FIGURA 5.1 - Ambiente de VISM

Imagine-se que o usuário tenha à sua disposição uma janela que corresponda a um ambiente de VIS e outra correspondendo a um ambiente de VIM. O modelo que está sendo visualizado na janela de modelagem é aquele sobre o qual está sendo realizada a experimentação na outra janela. Deste modo, durante a experimentação o usuário pode modificar o modelo, utilizando a mesma ferramenta gráfica-interativa que permite a sua construção, e continuar com os experimentos, sem a necessidade de interromper a simulação e começar novamente. Qualquer alteração no modelo causará automaticamente efeitos na experimentação e análise de resultados. A simulação de um sistema de tráfego seria um exemplo de aplicação deste novo paradigma. Durante a simulação, o usuário está visualizando a dinâmica do sistema na janela de experimentação. Para contornar os gargalos existentes no sistema, tais como o engarrafamento em um determinado cruzamento, o usuário poderia introduzir ou remover semáforos em diferentes pontos sem que ele tenha a necessidade de recompilar o modelo e reinicializar a simulação. Alguns problemas surgirão com a introdução destas modificações dinâmicas no modelo, pois observa-se que os dados estatísticos coletados antes e depois destas modificações estarão distorcidos [STA90], causando desta forma um regime transitório no qual o usuário deve tomar muito cuidado na interpretação dos resultados.

Outras janelas poderiam ser utilizadas para visualizar os dados de saída em tempo real. Desta forma, o usuário poderia selecionar a melhor forma de visualização (representação gráfica) para cada aspecto do modelo (entidades, atributos, filas, etc). Esta associação entre uma representação gráfica e um determinado aspecto do modelo poderia ser modificada de forma dinâmica, conforme o nível de entendimento do sistema. Como VISM deve permitir o uso da modelagem hierárquica, ou seja, o modelo é constituído de vários submodelos, outro recurso muito interessante é a associação de diferentes janelas para cada submodelo, tanto para propósitos de visualização como modificação.

Ambientes de simulação baseados na abordagem VISM devem conter pelo menos as seguintes ferramentas:

a) um editor gráfico para a especificação da estrutura do modelo. Este editor deve permitir a associação de representações gráficas às entidades do modelo. Para um melhor entendimento do sistema, o usuário pode modificar estas representações gráficas nas diferentes fases do processo de simulação (verificação, validação e análise);

b) uma linguagem gráfica para a especificação do comportamento do modelo, de acordo com a abordagem de modelagem desejada (por exemplo orientação a eventos ou processos);

c) um editor gráfico para a definição do cenário, utilizado para a apresentação visual do modelo. O cenário deve ser provido de recursos de animação e possuir interatividade durante a experimentação;

d) um tradutor, que transforma a especificação do modelo em uma representação interna (para uma simulação interpretada) ou em um código executável (para uma simulação compilada). Enquanto que a simulação interpretada é melhor para as fases de verificação e validação, a simulação compilada pode ser mais eficiente para a análise de resultados, caso não sejam mais necessárias modificações no modelo;

e) um módulo de experimentação que suporte todos os recursos visuais e interativos esperados em um ambiente de VIS. Os recursos disponíveis neste módulo devem estar integrados com os recursos oferecidos pelas ferramentas gráficas;

A principal característica e vantagem desta abordagem é permitir a modificação do modelo de simulação durante a experimentação. Como conseqüência disto, os ambientes de modelagem e experimentação devem estar totalmente integrados. Isto permite que o usuário faça alterações no modelo sem a necessidade de encerrar a experimentação, realizar suas alterações e iniciar novamente a experimentação.

5.3 Modelagem diagramática

Diversos métodos têm sido propostos para a especificação de modelos de simulação, não existindo um método fixo para um problema específico [PAU94]. Existem dois grandes conjuntos de métodos: de um lado estão os métodos diagramáticos, os quais dão uma representação bastante simples do modelo; do outro lado estão os métodos formais que se apóiam fortemente na matemática os quais são, geralmente, de difícil compreensão. Como apresentado na seção anterior, ambientes de simulação baseados na abordagem VISM necessitam de ferramentas gráficas para a construção do modelo de simulação. Desta forma, a modelagem diagramática será o método utilizado na abordagem VISM.

A **modelagem diagramática** permite que os modelos de simulação sejam descritos como redes de ícones conectados. Este tipo de modelagem usa tipicamente um número limitado de símbolos, facilitando desta forma a tarefa de modelagem, tornando a compreensão do modelo mais fácil uma vez que sua sintaxe e semântica são geralmente mais simples do que uma seqüência de ações nas representações procedurais (linguagens de programação) e, melhorando a comunicação entre o modelador e o usuário. A característica mais poderosa da modelagem diagramática é a capacidade de habilitar a modelagem hierárquica. Ceric [CER94] apresenta uma visão geral de diversos métodos de modelagem diagramática usados em simulação discreta orientada a eventos. Alguns dos métodos mais representativos são redes de Petri, diagramas de atividades, gráficos de eventos, diagramas ciclo-atividades e diagramas de blocos GPSS.

É importante observar que desenvolver um modelo de um sistema não é simplesmente desenhar diagramas, mas utilizá-los adequadamente como forma de auxílio durante este processo. Os diversos aspectos de um sistema devem ser representados por notações diagramáticas adequadas. A tentativa de representar todos os aspectos num único diagrama pode causar confusão a quem estiver desenvolvendo o sistema ou tentando entendê-lo. Desta forma, é importante que a especificação de um modelo do sistema não seja realizada sobre uma única notação diagramática, mas sim sobre a agregação de notações diagramáticas, cada qual representando um aspecto distinto do sistema.

O paradigma de orientação a objetos permite que se faça uma analogia entre a construção de um modelo de simulação e o desenvolvimento de um sistema orientado a objetos. Segundo Rumbaugh [RUM97], quando do desenvolvimento de um sistema orientado a objetos, este pode ser modelado segundo três modelos básicos relacionados e cada um abrangendo aspectos importantes do sistema: o modelo estático, o modelo dinâmico e o modelo funcional. A metodologia que combina estes três modelos é denominada de OMT ("Object Modeling Technique").

No **modelo estático** é definida a estrutura das classes de objetos no sistema em termos de identidade, estrutura interna (atributos e métodos) e seus relacionamentos. O modelo estático representa os aspectos estáticos e estruturais do sistema. A partir do modelo estático, os modelos dinâmico e funcional podem ser definidos. O **modelo dinâmico** serve para identificar os aspectos do sistema relacionados ao tempo e à seqüência de operações. O modelo dinâmico representa a estrutura de "controle" do sistema. Esta descreve as seqüências de operações que ocorrem em resposta a incentivos externos, sem no entanto descrever o que elas fazem ou como foram implementadas. No **modelo funcional** identificam-se os aspectos relacionados à transformação de valores em termos de funções, restrições e dependências funcionais. Cada um destes modelos descreve um aspecto do sistema, mas contém referências aos outros modelos. As interconexões entre estes três modelos devem ser claras, limitadas e explícitas. A idéia é que o acoplamento entre estes modelos seja o mínimo possível de modo que eles possam ser separados com maior facilidade.

Com base na metodologia OMT e com um projeto bem feito, isolando os diferentes aspectos do sistema, cada um dos três modelos que compõem o modelo do sistema pode ser descrito de uma forma diferente. Por exemplo, o modelo dinâmico utilizado para descrever o comportamento das entidades do modelo, pode ser escrito através de um diagrama de estados, redes de Petri ou até diretamente em código C++. Todas estas formas de descrição estariam à disposição do usuário, que utiliza uma ou outra conforme seu conhecimento.

6 VISME - Um ambiente de simulação segundo a abordagem VISM

6.1 Introdução

Este capítulo apresenta a estrutura e os recursos planejados para o ambiente *VISME* (***Visual Interactive Simulation and Modeling Environment***). *VISME* é um ambiente de modelagem e simulação baseado na abordagem VISM, apresentando uma interface única baseada em múltiplas janelas, que fornece todos os recursos necessários para construir e experimentar modelos de simulação. A arquitetura deste ambiente pode ser vista na figura 6.1.

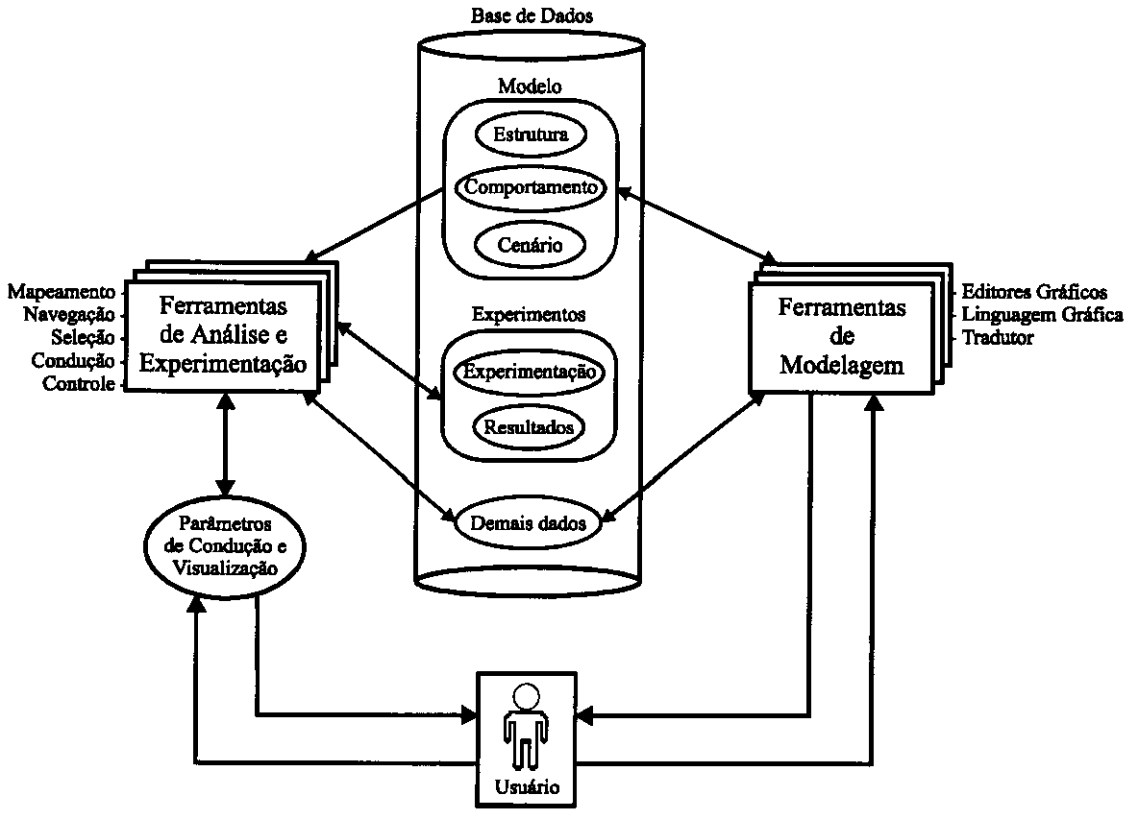


FIGURA 6.1 - Arquitetura do ambiente VISME.

O ambiente *VISME* contém dois módulos principais, cada um contendo um conjunto bem distinto de ferramentas: um para a modelagem e outro para a experimentação e análise do modelo. Através destas ferramentas, que devem possuir recursos visuais e interativos, o usuário interage com o modelo de simulação (sua estrutura e comportamento). As ferramentas de modelagem são as responsáveis pela construção do modelo de simulação de forma gráfica e interativa, enquanto que as

ferramentas de experimentação e análise são responsáveis pela execução da simulação e análise dos resultados.

O ambiente *VISME* possui uma interface única que fornece todos os recursos necessários aos módulos de modelagem e experimentação. Esta interface é rica em elementos gráficos e poderosa em recursos de interação com o usuário. Através dela o usuário pode experimentar sobre o modelo sendo simulado (monitorar variáveis/atributos, alterar distribuições de probabilidade, consultar atributos dos objetos), modificar o modelo (criar/remover objetos/instâncias) sem a necessidade de interromper a experimentação, armazenar/buscar informações na base de dados, associar variáveis/atributos ou parâmetros com janelas de visualização com capacidade de exibição gráfica. A idéia é que o usuário possa criar/remover estas janelas de visualização durante a experimentação, com a finalidade de observar os aspectos do modelo que ele julgue relevantes para a sua compreensão. Além destas janelas de visualização (janelas temporárias) existem outras 3 janelas fixas no ambiente *VISME* que podem ser habilitadas/desabilitadas a qualquer momento, cada uma relacionada com uma ferramenta de modelagem.

Um ponto importante do ambiente *VISME* é a sua capacidade de armazenar seus dados em uma base de dados. Os dois módulos principais (suas ferramentas e recursos) interagem com a base de dados que contém todas as informações necessárias para a modelagem e experimentação de um modelo. A base de dados em *VISME* pode ser dividida em três partes: dados de experimentação, dados de modelagem e dados gerais. Nos dados de experimentação podem estar armazenadas todas as informações que o usuário julgar importantes durante a experimentação, tais como: histórico de seus experimentos, estados intermediários e dados a serem analisados em pós-processamento. Nos dados de modelagem estão armazenadas as informações relativas ao modelo de simulação, tais como: sua estrutura (diagrama de classes), seu comportamento (diagrama de estados) e o seu cenário. Tais informações estarão armazenadas de tal forma que futuros modelos possam fazer uso delas. Nos dados gerais estão armazenadas as demais informações pertinentes a todo o processo como por exemplo a biblioteca de representações gráficas (ícones).

Os próximos ítens descrevem a funcionalidade de cada uma das ferramentas e recursos presentes no ambiente *VISME*.

6.2 Ferramentas de experimentação e análise

As ferramentas de experimentação e análise são as responsáveis pela experimentação do modelo. Estas ferramentas suportam todos os recursos visuais interativos esperados de um ambiente de VIS. O usuário interage com estas ferramentas através dos parâmetros de condução e visualização. Quando o usuário realiza uma operação (ativação de uma ferramenta) de experimentação no modelo, o ambiente normalmente solicita ao usuário que ele forneça uma determinada informação (parâmetro) para que esta operação seja executada. Estas ferramentas são

divididas em três categorias: recursos de monitoramento, recursos de controle e recursos de análise.

6.2.1 Recursos de monitoramento

Nesta categoria de recursos estão aqueles que permitem acompanhar a experimentação através de recursos de visualização e de condução. Dentre os principais destacam-se:

a) mapeamento: permite associar entidades e atributos do modelo a representações gráficas (ícones, gráficos, tabelas, etc). Estas associações podem ser modificadas durante a experimentação.

b) navegação: permite "navegar" sobre a representação gráfica do modelo (estrutura), principalmente se ele for hierárquico.

c) seleção: permite selecionar objetos do modelo e conjunto de objetos tanto para a execução de uma atividade sobre este(s) objeto(s) como para a diminuição do volume de dados a ser analisado.

d) condução: permite conduzir o experimento através da seleção do modo de execução da simulação (passo a passo, por quantidade de tempo, por tempo específico ou até que se atinja uma condição), ativação/desativação do monitoramento de variáveis, salvamento e recuperação de estados intermediários, etc.

6.2.2 Recursos de controle

São aqueles que permitem a interação com o modelo durante a experimentação através da modificação de parâmetros, valores de variáveis e atributos. A modificação da estrutura do modelo e do comportamento de seus objetos é realizada diretamente nas ferramentas de modelagem. A seção 6.6 descreve de uma forma mais detalhada os recursos de controle existentes no ambiente *VISME*.

6.2.3 Recursos de análise

São aqueles que permitem a análise dos resultados durante e após o término da experimentação. Para que seja realizada uma análise sobre uma determinada informação, é necessário que durante a experimentação do modelo sejam informadas quais as suas características que serão analisadas. Na análise em pós-processamento, estas informações serão armazenadas na base de dados onde após o término da experimentação o usuário, através dos recursos disponíveis, pode analisá-las mais detalhadamente.

Os recursos de análise oferecidos podem ser obtidos de duas formas: visualização estatística e animação. A visualização estatística permite acompanhar através de gráficos (tais como histogramas, diagramas de barras, relógios e gráficos

X-Y e X-Y-Z) e estatísticas (tais como média, variância, desvio padrão, mínimos e máximos) a alteração do conteúdo de determinadas variáveis e atributos de entidades ao longo da simulação. A animação em *VISME* é usada para a apresentação visual do fluxo de atividades no modelo. Existem vários pontos na simulação na qual a animação dos dados é geralmente coletada e apresentada, por exemplo: chegada e saída de entidades temporárias, aquisição e liberação de recursos e, início e final das atividades. Tais eventos podem ser representados de diversas formas: mudando a cor ou a forma da representação gráfica (ícone) da entidade pode indicar um estado de livre ou ocupado e a movimentação de ícones (com ou sem mudança na aparência) pode refletir os movimentos das entidades no tempo e espaço. A animação é mais fácil de acompanhar e analisar do que informações textuais (não significa que elas não devam existir, muito pelo contrário, às vezes uma informação textual é mais clara do que a animação) e pode ajudar os usuários a aumentar sua intuição sobre a dinâmica do processo e rapidamente observar uma anomalia ou um gargalo no sistema.

6.3 Ferramentas de modelagem

As ferramentas de modelagem são as responsáveis pela construção visual interativa do modelo de simulação. O modelo de simulação é composto por 3 partes (figura 6.2): sua estrutura, seu comportamento e o cenário. Cada uma destas partes contém aspectos relevantes e distintos do modelo, mas todos necessários para uma descrição completa do mesmo. Para a modelagem da estrutura do modelo (parte estática) é utilizado um diagrama de classes. A modelagem do comportamento (parte dinâmica) é obtida através de diagramas de estados e o cenário (parte funcional) é construído por intermédio de um editor gráfico específico.

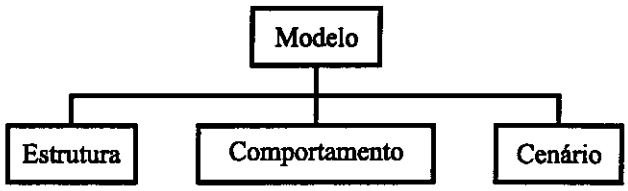


FIGURA 6.2 - Composição do modelo em VISME.

A seguir são apresentadas as diversas ferramentas sugeridas para apoio à construção e manipulação do modelo de simulação.

6.3.1 Diagrama de classes

Esta ferramenta é um editor gráfico onde é definida a *estrutura estática* do modelo de simulação. Este diagrama é uma rede de classes e relacionamentos e está baseado no diagrama de informações proposto por Copstein [COP97]. Nele são descritos quais são os tipos de classes de entidades presentes no modelo, quais são os

atributos de cada classe e quais são os relacionamentos existentes entre estas classes. O diagrama de classes é um gráfico cujos nodos são as classes de objetos e cujos arcos são os relacionamentos entre as classes. Através deste diagrama os modelos são construídos de forma hierárquica utilizando uma abordagem top-down, permitindo o uso dos mecanismos de hierarquia de herança e de agregação.

Uma classe é representada graficamente por um retângulo, sendo que este apresenta diversas informações sobre esta classe descrita (figura 6.3). No lado direito do retângulo existem três ícones que indicam as abordagens que compõem o paradigma de simulação utilizado para descrever o comportamento da classe [COP96]. O ícone superior especifica a forma de recepção das mensagens (ativo ou passivo); o ícone intermediário indica a abordagem de descrição do comportamento (orientado a eventos, processos ou atividades); e, o ícone inferior apresenta a abordagem utilizada para a troca de mensagens (mensagens ou portas). No centro do retângulo especifica-se o nome da classe descrita e no canto superior esquerdo a sua cardinalidade, que é utilizada para limitar o número máximo de instâncias desta classe no modelo. Também deve ser informado se esta classe é permanente ou temporária, ou seja, se suas instâncias estarão sempre presentes na experimentação (permanentes) ou se irão aparecer e desaparecer durante a mesma (temporárias).

FIGURA 6.3 - Representação gráfica de uma classe.

Uma vez que o modelo estático é descrito hierarquicamente, deve-se definir primeiramente a classe de nível mais alto na hierarquia representando todo o modelo. Esta classe no primeiro nível inclui todas as demais classes que são definidas em um sub-diagrama. Como o domínio de aplicação do ambiente *VISME* está concentrado em modelos discretos onde existe formação de filas, no momento do detalhamento desta classe de mais alto nível o diagrama padrão da figura 6.4 é apresentado, ou seja, todos os modelos em *VISME* são construídos a partir das classes padrões, "storage" e "facility". Estas duas classes foram baseadas nos conceitos da linguagem GPSS [COX87] que as utiliza para representar as entidades do sistema. Desta forma, o modelador descreve o sistema informando quais os recursos que este irá oferecer, sendo estes pertencentes a uma das seguintes classes:

a) **Facility**: são recursos que possuem um único servidor, ou seja, somente podem ser utilizados por uma única entidade em um dado instante de tempo. Caso uma entidade tente utilizar uma "facility" ocupada, então ela deve aguardar na fila específica desta "facility" até que ela fique desocupada. As "facilities" tem como característica a formação de filas próprias. No exemplo da agência bancária, a caixa específica para pessoas idosas pode ser modelada como uma "facility".

b) **Storage**: são recursos que possuem um ou vários servidores trabalhando em paralelo, ou seja, podem atender mais de uma entidade em um dado instante de tempo. A capacidade (número de servidores) de um "storage" deve ser definida quando de seu instanciamento no cenário, correspondendo ao número máximo de atendimentos em paralelo que este recurso admite. As "storages" tem como característica o compartilhamento de uma fila única por todos os seus servidores. Desta forma, caso uma entidade tente utilizar uma "storage" onde todos os seus servidores estão ocupados, então ela deve aguardar na fila única deste "storage" até que um dos servidores fique desocupado, independente de qual seja. No exemplo da agência bancária, as caixas de atendimento de fila única podem ser modeladas como uma "storage" com capacidade igual ao número de caixas existentes nesta agência.

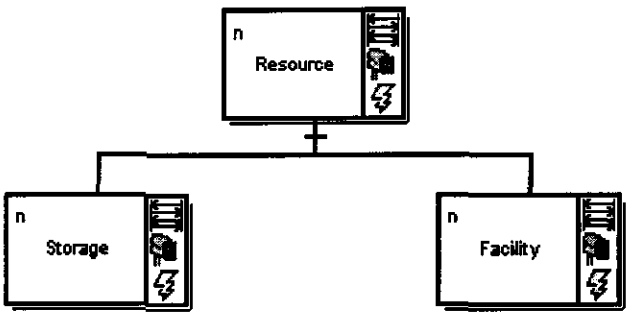


FIGURA 6.4 - Classes padrões dos modelos em VISME.

As classes descritas no diagrama de classes podem se relacionar com as demais classes através de três relacionamentos: herança, agregação e conhecimento. O **relacionamento de herança** permite que a classe compartilhe parte de sua descrição com outras classes. As classes Gerentes, Terminais e Caixas da figura 6.5 compõem uma hierarquia de herança onde a superclasse é a classe "Storage". Conforme observado na figura 6.4, os recursos definidos no diagrama de classes sempre serão classes derivadas das classes "Storage" e "Facility". Uma vez que o comportamento das filas são definidos nas classes "Storage" e "Facility", as subclasses herdam seus métodos e atributos. Adicialmente, estas subclasses podem incrementar ou redefinir o comportamento de sua superclasse.

O **relacionamento de agregação** indica que uma classe é composta por outras classes que são detalhadas em um subdiagrama. Este relacionamento é representado por um retângulo com bordas mais espessas. A figura 6.3 indica que a classe BANCO é uma classe composta cujo seu subdiagrama é apresentado na figura 6.5. Os **relacionamentos de conhecimento** tem a finalidade de informar que uma determinada classe conhece outra classe, permitindo deste modo que suas instâncias possam se comunicar através do sistema de mensagens definido [COP96]. Estes relacionamentos são representados por arcos que possuem sentido (unidirecionais ou bidirecionais) e cardinalidade indicada nas suas extremidades ("um para um", "um para muitos" ou "muitos para muitos"). Caso a cardinalidade seja maior do que 1, esta

deve ser especificada. Na figura 6.5 pode-se observar um relacionamento de conhecimento entre a classe Cliente e a classe Recurso. Neste caso, qualquer instância da classe Cliente conhece todas as instâncias existentes da classe Recurso, inclusive as classes que a compõem.

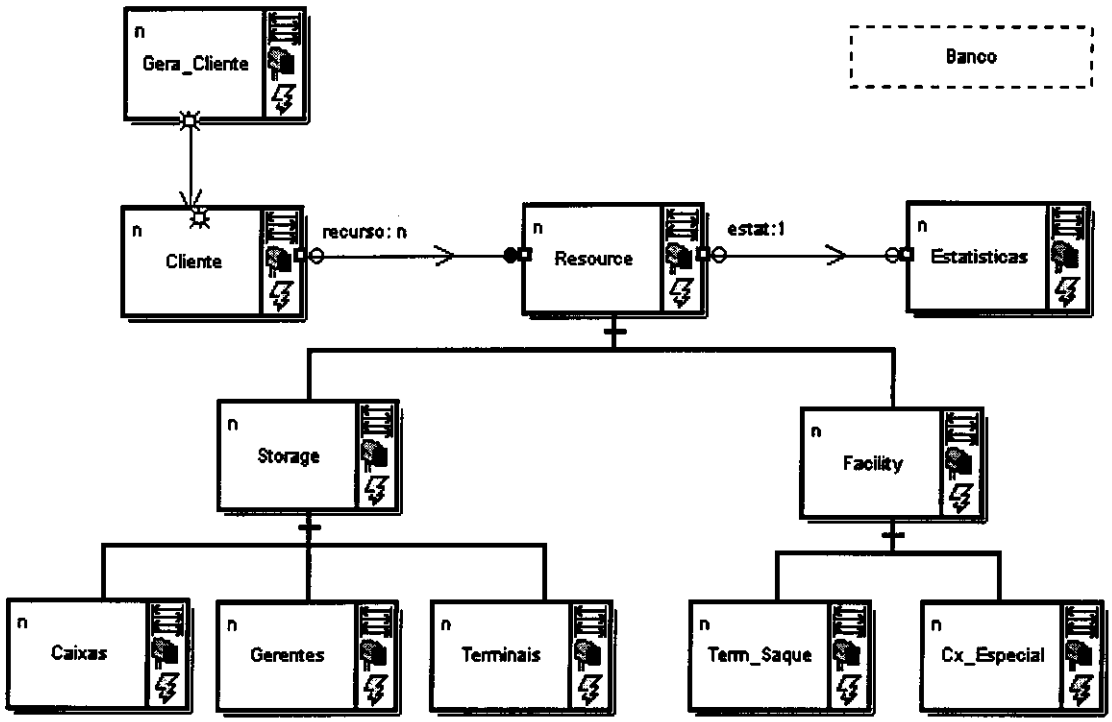


FIGURA 6.5 - Diagrama de classes da agência bancária.

Além destes três tipos de relacionamentos descritos acima, existe ainda o **relacionamento de criação** que representa a relação entre geradores de elementos de uma classe com a própria classe. Na figura 6.5 este relacionamento é observado entre as classes Gera_Cliente e Cliente, indicando que uma instância da classe Gera_Cliente gera instâncias da classe Cliente. Esta relação é comum em modelos de simulação, mostrando a existência de uma entidade permanente responsável pela criação de instâncias de uma entidade temporária segundo uma determinada distribuição de probabilidade.

Além dos atributos especificados pelo usuário para a descrição da classe, existem outros atributos importantes para a etapa de experimentação. Através da ferramenta de mapeamento, o diagrama de classes permite que se associe uma representação visual a uma classe (a seu atributo correspondente). Para auxiliar o usuário na sua tarefa de modelagem e experimentação, um atributo muito útil a uma classe é a visibilidade de sua representação visual durante a simulação. Isto se deve ao fato de que algumas classes modelam entidades do sistema a ser simulado e outras não. Tais classes estariam visíveis durante a construção do modelo e cenário, mas não durante a fase de experimentação. Como exemplos podem ser citadas as classes de geração de entidades temporárias, as classes consumidoras responsáveis pela coleta

de estatísticas (classes Gera_Cliente e Estatisticas da figura 6.5). Um tratamento especial deve ser dado em relação as classes "Storage" e "Facility". Estas duas classes possuem na realidade duas representações visuais: uma para a classe e outra para a fila descrita nela. As subclasses destas classes podem ter suas próprias representações visuais (redefinição deste atributo da superclasse) estando sempre visíveis durante a experimentação, o que não ocorre com a representação visual da fila.

6.3.2 Diagrama de estados

No ambiente *VISME*, a especificação do comportamento dos objetos presentes no modelo será realizada através de um **diagrama de estados** como proposto em [RUM97]. O diagrama de estados é um gráfico cujos nodos são estados e cujos arcos são transições entre estados causadas por eventos. Esta ferramenta é um editor gráfico onde é definida a **estrutura dinâmica** do modelo. Através deste diagrama serão especificados os comportamentos de cada uma das classes descritas no diagrama de classes. A dinâmica do modelo consiste em múltiplos diagramas de estados, um para cada classe descrita no diagrama de classes. Estes diagramas funcionam de forma concorrente e podem mudar de estado de forma independente. Os diversos diagramas de estado para as diversas classes combinam-se em um único modelo dinâmico por intermédio de eventos compartilhados.

O diagrama de estados especifica a seqüência de estados causados por uma seqüência de eventos. As transições indicam os possíveis caminhos de mudança de um estado para outro. Para que um estado passe para outro estado é necessário que exista uma transição unindo estes dois estados e que o **evento** (estímulo externo) que rotula esta transição seja recebido pelo objeto. Se existir mais de uma transição partindo de um estado, o primeiro evento que ocorrer faz com que a transição correspondente dispare. A figura 6.6 mostra a notação para diagrama de estados que será utilizada.

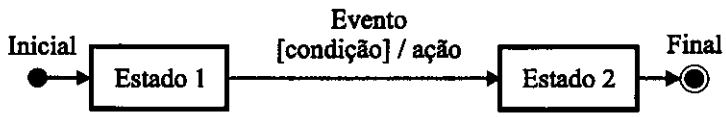


FIGURA 6.6 - Notação para diagrama de estados.

Um **estado** é representado através de um retângulo contendo um nome que é utilizado para informar o estado em que se encontra o objeto. O estado é uma abstração de valores de atributos e relacionamentos de um objeto. O estado especifica a reação do objeto aos eventos de entrada. A maioria dos sistemas modelados através de diagramas de estados possuem um estado inicial e outro final. O estado inicial ocorre quando da criação de um objeto (primeiro estado a ser realizado quando do início do ciclo de vida do objeto), enquanto que o estado final significa a destruição do objeto (último estado a ser realizado no ciclo de vida do objeto). Caso um sistema descrito tenha mais de um estado final, estes devem ser mutuamente exclusivos, o que

quer dizer que apenas um deles pode ocorrer durante a execução do sistema. O estado inicial é representado por um círculo preenchido e o estado final por um círculo preenchido dentro de um círculo simples.

Uma **transição** é representada graficamente por uma seta que liga dois estados distintos ou um estado a ele mesmo e é rotulada com o nome do evento que a causou. Todas as transições que partem de um estado devem corresponder a diferentes eventos, isto é, em um determinado instante de tempo uma e somente uma transição que parte de um estado pode disparar. As transições são constituídas por um evento, uma condição e uma ação e podem ser especificadas através de qualquer uma das maneiras conforme mostrado na figura 6.7. Observa-se que as transições não necessariamente devem ter a notação geral contendo um evento, uma condição e uma ação, mas sempre deve existir um evento rotulando-as. Por exemplo, a última transição apresentada na figura 6.7, faz com que haja uma mudança de estado quando ocorre o evento especificado.

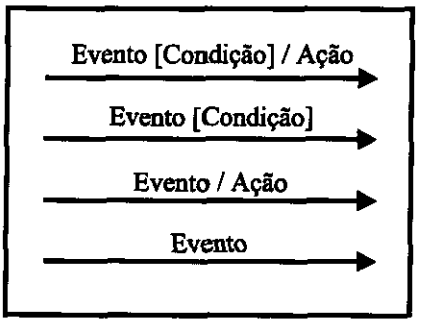


FIGURA 6.7 - Notações possíveis para transições.

Caso haja a definição de uma **condição** (função lógica) associada a uma transição, a mudança de estado através desta transição somente ocorre caso a condição seja satisfeita. As condições são utilizadas como "guardas" nas transições e são representadas como uma expressão lógica limitadas por colchetes. Também pode-se especificar opcionalmente uma ação asssociada a transição, representada após uma barra ("/"). Uma **ação** é uma operação imediata cuja duração é insignificante em relação à resolução do diagrama de estados. A execução de uma ação está sempre condicionada ao disparo da transição, ou seja, sempre que ocorre um evento e a condição, caso exista, seja satisfeita. As ações podem ser ativações de métodos do próprio objeto ou uma seqüência de declarações na linguagem de programação C++.

Um ponto deve ser ressaltado em relação ao estado final do diagrama de estados utilizado no ambiente *VISME*. Como é permitida a interação do usuário durante a experimentação, é possível que um objeto seja removido devido a uma intervenção do usuário. Como todos os objetos possuem um comportamento especificado através de um diagrama de estados, eles devem necessariamente se encontrar em um estado deste diagrama em qualquer instante de tempo durante a experimentação. Deste modo, fica implícito que o estado final pode ser atingido de qualquer estado definido no diagrama (não precisa ser representado no diagrama de

estados). Internamente o objeto removido da experimentação recebe uma mensagem de "end", executa o seu método "destrói" e atinge o estado final. É evidente que além deste estado final "automático", o usuário pode descrever no diagrama um estado final normal.

A figura 6.8 apresenta o comportamento de um gerente de um banco descrito através de um diagrama de estados. Este gerente é uma classe do modelo que foi definida no diagrama de classes mostrado na figura 6.5.

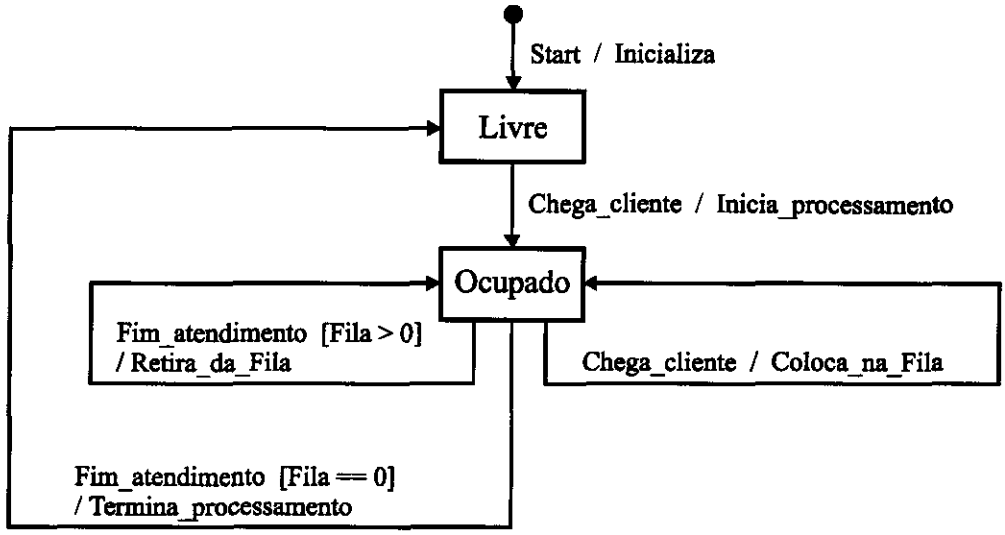


FIGURA 6.8 - Diagrama de estados da classe gerente.

Todas as instâncias de uma classe (objetos) são descritas pelo mesmo diagrama de estados (tem o mesmo comportamento), mas cada uma tem seu próprio diagrama. Isto significa que cada objeto tem seu próprio estado, podendo estar em um determinado estado do diagrama. A descrição comportamental de um objeto deve especificar o que o objeto realiza em resposta aos eventos que recebe. Os métodos das classes descritos no diagrama de estados são associados às transições e são executados em resposta aos correspondentes eventos. Caso seja necessário diferenciar o comportamento entre dois objetos (instâncias da mesma classe), então deve-se descrever duas classes distintas no diagrama de classes. Estas duas classes provavelmente serão subclasses da classe original, de modo que apenas algumas características do comportamento sejam diferenciadas e as demais herdadas.

6.3.3 Editor de cenário

Esta ferramenta é um editor gráfico específico utilizado para o projeto do cenário que é a representação visual do modelo. É neste cenário onde será realizada a interação com o usuário durante a fase de experimentação. Na realidade, o cenário do ambiente *VISME* é um **diagrama de instâncias**. É no cenário onde serão criadas as instâncias do modelo de simulação, bem como o relacionamento entre elas. Sem o

diagrama de instâncias (isto é, somente com o diagrama de classes e diagrama de estados), o modelo do sistema descrito seria válido para qualquer sistema semelhante que contivesse os mesmos objetos descritos. Para que um modelo seja simulado é necessário que ele não seja genérico, ou seja, é necessário definir um modelo em particular. Isto é feito através do diagrama de instâncias. Desta forma, o usuário pode construir diferentes cenários que poderão representar diferentes modelos de um determinado sistema. No exemplo da agência bancária, para o mesmo diagrama de classes e diagrama de estados podemos ter dois cenários distintos: um de um banco onde todos os clientes que entram tem acesso a todas as caixas disponíveis e outro onde existem caixas diferenciadas para clientes específicos. A figura 6.9 apresenta o cenário para a agência bancária descrita na seção 2.3.

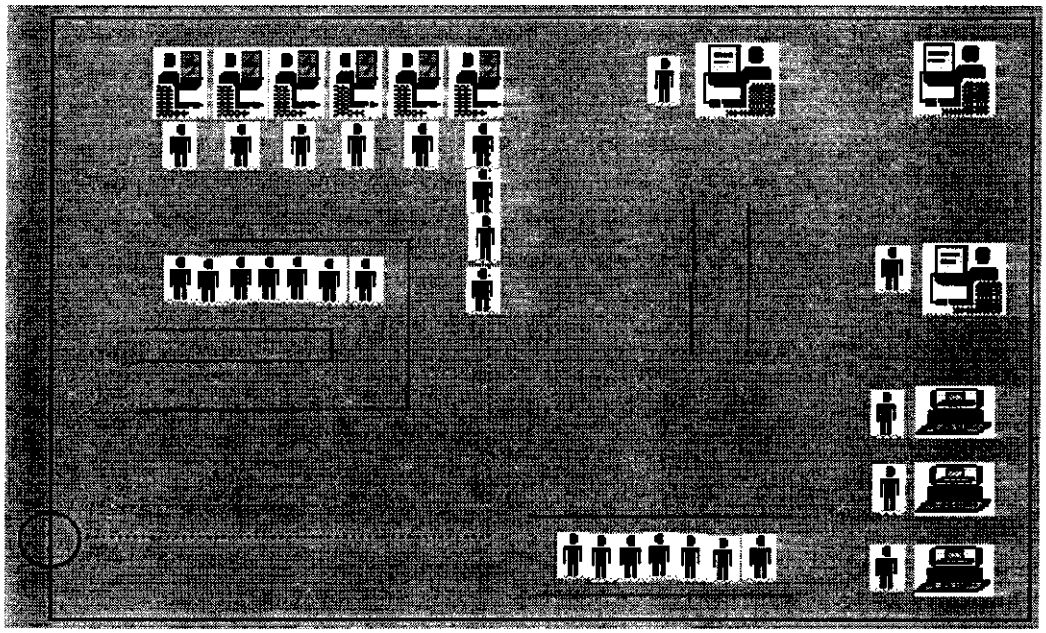


FIGURA 6.9 - Cenário de uma agência bancária.

Para a construção do cenário, dois tipos de representações devem ser manipuladas: o fundo que representa o "layout" estático do cenário e as representações visuais (ícones) dos objetos (recursos) definidos no diagrama de classes. Dois conjuntos de ferramentas estão disponíveis para a definição do fundo estático: a) uma paleta de ícones que deve ser carregada de uma biblioteca de ícones. Normalmente escolhe-se uma biblioteca com ícones representativos com o tipo de sistema que está se modelando. b) uma paleta com uma série de recursos de edição que permitem realizar desenhos tais como linhas, retângulos, círculos, textos, etc. Através das operações de seleção e posicionamento, o usuário pode definir o fundo estático. Deve-se salientar que este fundo não tem nenhuma influência na experimentação, seu único objetivo é tornar o cenário o mais realístico possível.

Uma vez que o cenário é um diagrama de instâncias, deve-se especificar quais serão as instâncias das classes descritas no diagrama de classes que definirão o modelo específico que está se construindo. A definição de uma nova instância é realizada através da seleção da representação visual da classe da qual se quer criar esta instância em uma paleta contendo as representações visuais das classes descritas no diagrama de classes. Assim como o fundo estático, deve-se realizar o posicionamento no cenário da representação visual da instância selecionada. No caso das entidades temporárias, é necessária a definição da instância da classe que gera estas entidades (no exemplo da figura 6.5 seria a classe Gera_Cliente). Isto significa que na construção do cenário e durante a experimentação, as instâncias das entidades temporárias não podem ser criadas interativamente pelo usuário como os recursos (entidades permanentes).

Como conseqüência das possíveis intervenções do usuário durante a experimentação, a paleta contendo as representações visuais das classes se auto-configura sempre que alguma mudança no diagrama de classes ocasione uma alteração de uma das representações visuais disponibilizadas para o cenário. O cenário também é sensível a este tipo de modificação, ou seja, quando a representação visual de uma classe for modificada, todas as suas instâncias no cenário também serão modificadas automaticamente.

Após a definição do fundo e o posicionamento de cada instância (sua representação visual) dos objetos, deve-se realizar o relacionamento entre as instâncias definidas no cenário. Isto é necessário pois quando é especificado um relacionamento de conhecimento entre duas classes no diagrama de classes, não significa que todas as instâncias destas duas classes estarão relacionadas. Quando duas instâncias são relacionadas é realizada uma consistência para verificar se este relacionamento é compatível com os definidos entre as classes no diagrama de classes. Também neste momento deve-se especificar qual a trajetória (caminho) no cenário que deve ser seguida pela representação gráfica da instância visando uma animação. A animação será tratada com mais detalhes na seção 6.4. No exemplo da agência bancária existe o relacionamento de conhecimento entre as classes Cliente e Resource (figura 6.5). Conforme a modelagem realizada, o cliente pode ser de vários tipos: idoso, VIP, office-boy, etc. Caso não seja especificado o relacionamento entre as instâncias no cenário, estaria sendo assumido que qualquer cliente pode utilizar qualquer recurso da agência. Caso seja necessário definir uma agência em particular, onde determinados tipos de clientes somente podem utilizar alguns recursos específicos, será necessário estabelecer quais os recursos que podem ser utilizados para os clientes VIP, os recursos para os office-boys, e assim por diante. Esta particularidade é realizada através do relacionamento entre as instâncias de cliente com as instâncias dos recursos disponíveis no cenário da agência.

No cenário apresentado na figura 6.9 foi realizado o relacionamento entre os clientes e os recursos disponíveis na agência. Para cada um destes relacionamentos também foi definida a trajetória que o cliente deve seguir quando for utilizar um dos recursos disponíveis a ele. No caso da figura 6.9 somente foi apresentada a trajetória a ser seguida por um cliente quando ele entra na agência e for utilizar a caixa especial. Quando uma instância da classe Cliente é criada (um cliente chega na agência) ela é

enviada para utilizar um determinado recurso, direcionamento este regido por uma distribuição de probabilidade que define qual a tarefa que ela deve executar. Caso este recurso já esteja ocupado, esta instância fica aguardando na fila específica do recurso. Após a utilização do recurso esta instância cliente passa a executar uma nova tarefa, da lista de tarefas que foi atribuída a ela quando de sua criação.

Neste editor de cenários é que se tem a integração entre a modelagem e experimentação, ou seja, o controle completo do processo de simulação. Neste cenário pode-se remover/incluir entidades, alterar seu comportamento, modificar atributos e continuar a experimentação, sem ter a necessidade de interrompê-la e recomeçar novamente. No exemplo da agência bancária, o usuário poderia incluir um novo caixa, remover um terminal eletrônico, alterar o comportamento de uma determinado gerente, etc. Os recursos disponíveis para o controle da experimentação oferecidos para o usuário serão tratados com mais detalhes na seção 6.4.

6.3.4 Tradutor

O tradutor é uma ferramenta que permite a geração automática do código de simulação na linguagem C++ a partir da especificação gráfica do modelo. O tradutor é ativado sempre que um processo de experimentação é disparado, ou seja, sempre que o usuário interrompe o experimento e realiza uma modificação no modelo, seja estrutural, comportamental ou até na apresentação visual, um novo código é gerado automaticamente. Esta operação é totalmente transparente para o usuário. O código gerado é montado a partir das seguintes ferramentas, onde cada uma é responsável por uma parte do código: a) diagrama de classes: são criadas as definições das classes e seus atributos; b) diagrama de estados: a partir deste editor gráfico são definidos os métodos das classes que estão vinculados aos eventos associados às transições. Estes métodos são descritos diretamente na linguagem do código gerado e também fazendo uso de uma biblioteca de classes para a simulação [COP96]. c) editor de cenários: são inicializados os atributos que representam os relacionamentos estáticos (qual instância de uma classe que está relacionada com outra instância de outra classe). O código gerado pelo ambiente *VISME* será tratado com maiores detalhes no capítulo 7.

6.3.5 Editor de ícones

É um editor gráfico que permite criar as representações visuais (ícones) que serão associadas às classes (ao seu atributo correspondente) e aos estados dos diagramas de estados. Neste editor é possível agrupar um conjunto de ícones em uma biblioteca de ícones. Esta biblioteca de ícones tem a finalidade de reunir ícones comuns a um determinado assunto. Desta forma, quando da associação de ícones às entidades do modelo, pode-se carregar uma biblioteca específica que contenha os ícones mais representativos para o sistema que se está modelando.

6.4 Animação

Uma característica importante de *VISME* é a sua capacidade de animação, ou seja, a apresentação visual do fluxo de atividades no modelo. Antes, é necessário dizer que além da classificação tradicional das entidades em permanentes ou temporárias, as entidades em *VISME* também são classificadas de estáticas ou dinâmicas, conforme a sua forma de animação. As entidades estáticas são aquelas onde a posição de sua representação visual no cenário é fixa enquanto que as entidades dinâmicas podem se movimentar ao longo do cenário através de uma trajetória estabelecida. Com base nesta classificação, para o exemplo da agência bancária os clientes seriam definidos como entidades temporárias dinâmicas e os gerentes/caixas/terminais como entidades permanentes estáticas. Considerando uma fábrica montadora de automóveis, onde as peças são transportadas de uma seção para outra através de AGV's, poderíamos definir que as peças transportadas seriam entidades temporárias estáticas e os AGV's seriam entidades permanentes dinâmicas.

A animação em *VISME* pode ser vista de duas formas:

a) animação dinâmica: referente à movimentação das representações visuais das entidades dinâmicas ao longo de trajetórias definidas no cenário. Quando se realiza um relacionamento entre duas instâncias no cenário, deve-se especificar qual a trajetória (caminho) que deve ser seguida pela instância (entidade temporária dinâmica). No momento em que uma instância de uma entidade temporária dinâmica é criada, ela recebe um conjunto de tarefas (comportamento) que deverá executar enquanto estiver participando da experimentação (até ser "consumida"). Quando uma destas tarefas é disparada para ser executada, uma mensagem é enviada para o recurso que será ocupado informando a chegada desta instância em um determinado instante de tempo. Neste momento, é disparada uma mensagem para o sistema de animação indicando esta mesma operação. Com isto, é iniciada a movimentação da representação visual desta instância ao longo da trajetória estabelecida. Também dentro desta animação é possível definir uma seqüência de representações visuais, onde cada representação visual desta seqüência é exibida alternadamente durante a movimentação da instância, aparentando uma animação mais realística.

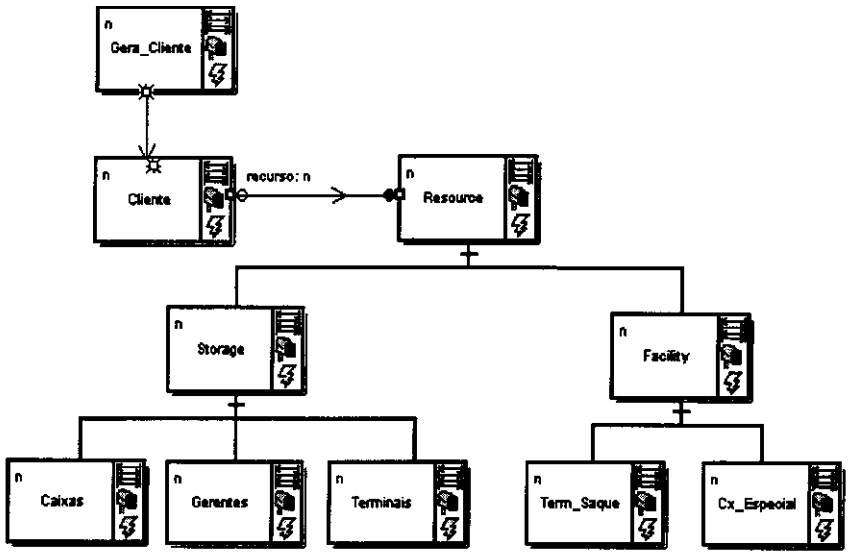
b) animação icônica: é realizada através da modificação de um atributo gráfico da representação visual da entidade. Esta modificação é realizada na definição do comportamento da entidade no diagrama de estados, onde para cada estado do diagrama podemos definir um atributo gráfico diferente. Os possíveis atributos gráficos são: um outro ícone, uma mudança de cor, uma alteração no tamanho ou até mesmo uma rotação do mesmo. Desta forma, quando uma entidade muda de estado e existe uma diferença entre os atributos gráficos desta entidade nestes estados envolvidos, existe uma mudança visual desta entidade no cenário. Existe internamente um "monitor" que fica permanentemente monitorando todas as instâncias do cenário e, sempre que ocorrer alguma modificação de estado em uma determinada instância, este "monitor" envia uma mensagem adequada ao sistema de animação.

A animação de recursos do tipo "storage" (entidades permanentes estáticas) é realizada de forma independente para cada um dos servidores disponíveis neste recurso. A "storage" como um todo possui seu comportamento descrito por um único diagrama de estados, ou seja, todos os servidores possuem o mesmo comportamento. A "storage" é composta por um conjunto de servidores onde cada um possui um ícone e, como cada servidor pode estar em um determinado estado do diagrama de estados, então é possível realizar uma animação icônica nesta "storage".

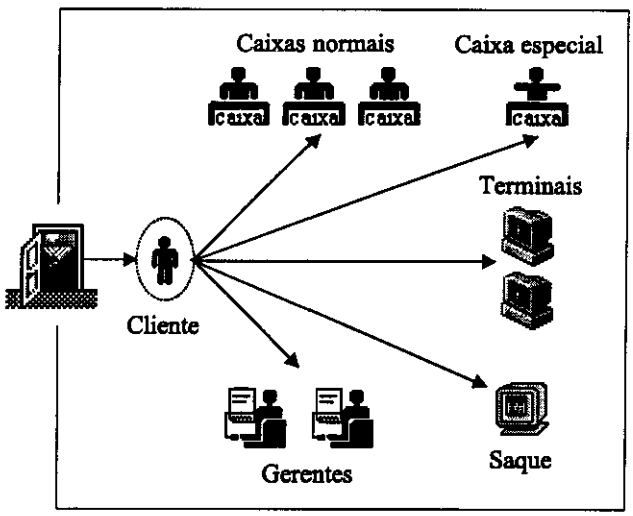
6.5 Alternativas de modelagem

O ambiente *VISME* permite que o usuário construa diferentes modelos para um mesmo sistema. Estes modelos podem ser armazenados numa base de dados. A primeira atividade a ser realizada na construção de um modelo é a definição de sua estrutura estática (diagrama de classes). Considerando o exemplo da agência bancária, pode-se fazer um estudo mais detalhado de como modelar a entidade temporária cliente. Supondo que para um sistema em particular possam existir os seguintes tipos de cliente: normal, vip e office-boy. Com base nisto, será feita uma análise de 3 casos com diferentes alternativas de modelagem que podem ser utilizadas pelo modelador com respeito às entidades temporárias. Em todos os casos, a classe Gera_Cliente é responsável pela criação das instâncias da entidade temporária cliente segundo uma determinada distribuição de probabilidade. A classe Cliente também contém uma lista de tarefas (comportamento descrito através do diagrama de estados) que deve ser executada pelas instâncias criadas.

A forma mais simples e natural de representar a estrutura estática desta agência é apresentada na figura 6.10a, onde todos os tipos de clientes conhecem todos os recursos disponíveis na agência. Para este diagrama de classes as seguintes conclusões podem ser extraídas: a) a classe Gera_Cliente cria instâncias de qualquer tipo de cliente, sendo o tipo de cliente um atributo da classe Cliente gerado segundo uma distribuição de probabilidade qualquer; b) todas as instâncias de cliente possuem o mesmo comportamento; c) a representação visual do cliente é a mesma para todos os tipos de cliente; d) a lista de tarefas definida para cada instância de cliente é independente do tipo de cliente gerado; e) todo o cliente que entra na agência conhece todos os recursos oferecidos por ela, ou seja, qualquer instância de cliente gerada pode se relacionar com qualquer recurso da agência. Isto significa que na construção do cenário o cliente deve ser relacionado com todos os recursos que ele pode utilizar, conforme a figura 6.10b. Neste tipo de modelagem não é possível disponibilizar um recurso específico para um determinado tipo de cliente.



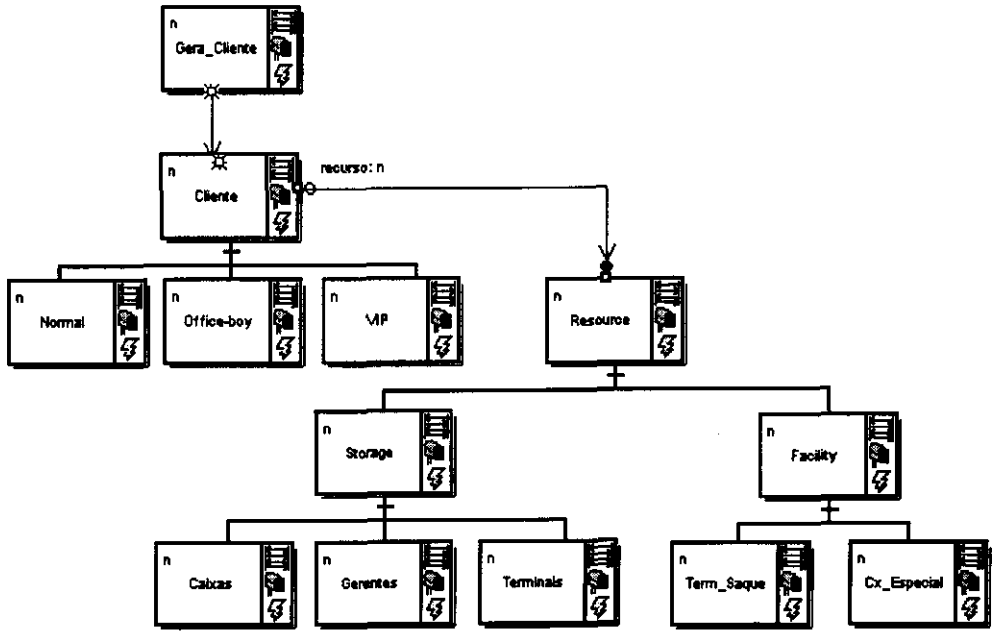
a) diagrama de classes



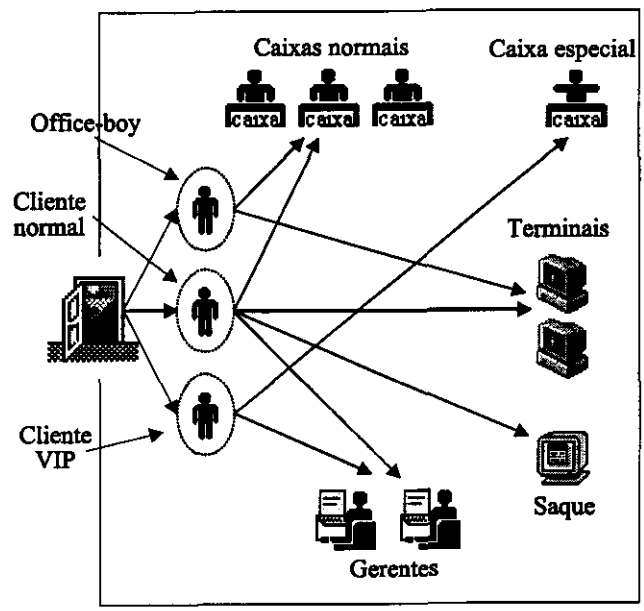
b) cenário

FIGURA 6.10 - Alternativa 1: todos clientes conhecem todos os recursos.

Supondo que seja necessária a diferenciação quanto aos recursos utilizados por determinados tipos de clientes, poderiam ser criadas subclasses para a classe Cliente, uma para cada tipo de cliente possível (figura 6.11a). A classe Cliente continuaria possuindo uma lista de tarefas, um comportamento próprio e sua própria representação gráfica. A diferença é que com o mecanismo de herança cada subclasse (cada tipo de cliente) pode redefinir as características da classe Cliente. Com esta nova modelagem, cada tipo de cliente pode ter sua própria lista de tarefas, seu comportamento próprio e sua própria representação gráfica desde que façam sobrecarga em relação a classe Cliente. Assim como no modelo anterior, todos os tipos de clientes podem utilizar qualquer recurso da agência.



a) diagrama de classes



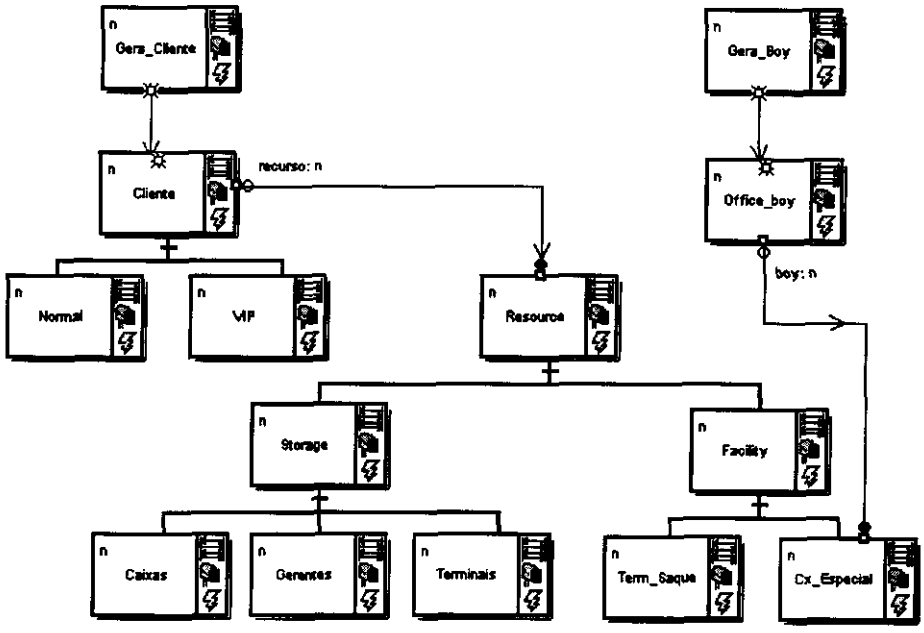
b) cenário

FIGURA 6.11 - Alternativa 2: clientes distintos podem conhecer recursos distintos.

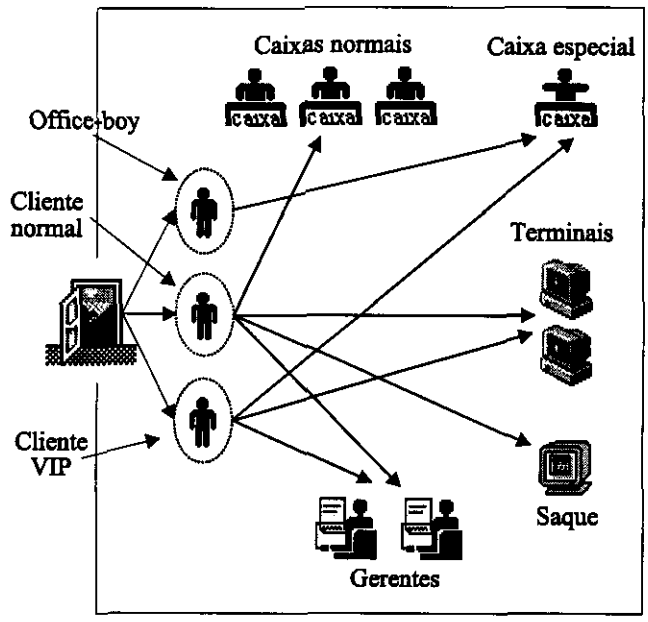
Nesta segunda alternativa de modelagem, na construção do cenário (figura 6.11b), deve-se realizar o relacionamento de cada tipo de cliente com os recursos que ele realmente irá utilizar. Com isto pode-se destinar um determinado recurso para um tipo específico de cliente. Em termos visuais e de entendimento do sistema, a primeira alternativa de modelagem não é tão clara quanto esta segunda, uma vez que não é possível distingüir que tipo de recurso pode ser utilizado por um determinado tipo de cliente. Na figura 6.11b pode-se verificar que as instâncias

geradas do cliente VIP somente irão se relacionar com os recursos Caixa especial ("facility") e Gerentes ("storage").

Uma outra forma de disponibilizar um recurso específico para um tipo de cliente em particular pode ser visto na figura 6.12a. Neste caso a classe Office_boy, que é uma entidade temporária assim como as demais subclasses da classe Cliente, somente pode se relacionar com o recurso da classe Cx_especial, não conhecendo os demais recursos disponíveis na agência bancária. No caso anterior também se poderia realizar somente este relacionamento para as instâncias da classe Office-boy. A diferença está no fato de que esta forma de modelar é mais restritiva do que a anterior, não permitindo que instâncias desta classe se relacionem com outros recursos, tornando o modelo mais seguro e confiável. Como observado na figura 6.12b, os demais tipos de cliente podem se relacionar com qualquer recurso, inclusive com a classe Cx_especial.



a) diagrama de classes



b) cenário

FIGURA 6.12 - Alternativa 3: cliente específico conhece recurso específico.

Outro ponto que pode ser analisado na modelagem é quanto a forma de modelar o tempo a ser consumido por uma entidade temporária na utilização de um recurso. Pode-se optar por uma maior complexidade no comportamento da entidade temporária ou dos recursos, ou seja, o tempo a ser consumido é especificado no comportamento de uma ou de outra. Para o caso do cliente, quando uma instância é criada ela já tem definido o tempo que irá ocupar os recursos a serem utilizados por ela. Dependendo da forma de modelar, pode-se especificar um tempo distinto para cada tipo de cliente. Já para o caso dos recursos, o cliente fica utilizando o recurso segundo um tempo especificado no comportamento do próprio recurso. Também pode-se especificar um comportamento mais complexo para o recurso que determine um tempo de atendimento diferenciado para o tipo de cliente que irá utilizá-lo.

6.6 Recursos de controle disponíveis

VISME, diferentemente dos outros ambientes descritos na literatura e disponíveis comercialmente, disponibiliza facilidades de modelagem que podem ser executadas a partir da interação com o cenário projetado durante a experimentação. Deste modo, caso o usuário deseje, ele pode continuar normalmente seu experimento do ponto onde ele foi interrompido, sem a necessidade de se iniciar um novo experimento. A seguir serão comentadas as principais facilidades e recursos oferecidos pelo ambiente *VISME* durante a experimentação.

A **modificação do comportamento das entidades** é um recurso que permite uma grande flexibilidade na correção do modelo por ocasião das fases de validação e análise do modelo de simulação. Este recurso é realizado através da

ferramenta do diagrama de estados (a mesma utilizada na modelagem) e pode ser executado de duas formas: a) a partir da seleção de uma instância no cenário. Através desta seleção automaticamente já se sabe qual a classe que esta instância pertence; b) selecionando uma classe no diagrama de classes. Para ambos os casos, o diagrama de estados da classe é ativado com o comportamento atual da classe. Caso seja realizada alguma alteração no comportamento desta classe, este passa a valer para todas as instâncias desta classe. Desta forma, não é permitido modificar o comportamento de uma única instância uma vez que todas as instâncias devem possuir o mesmo comportamento definido na classe. Internamente um novo código será gerado uma vez que o modelo está sendo modificado durante a experimentação. No exemplo da agência bancária, podemos citar a modificação do comportamento dos caixas que a partir de um determinado momento não aceitam mais determinados pagamentos e envia os clientes para um gerente.

Dentro deste contexto de alteração do comportamento das instâncias, ou seja, alteração do diagrama de estados, vários procedimentos podem ser realizados pelo usuário sendo que para cada um, uma determinada reação será efetivada no ambiente. O usuário pode: a) modificar o código de uma ação de uma determinada transição, incluir um novo estado, alterar o evento que dispara uma transição, alterar uma determinada condição. Tais operações não causam reflexos imediatos no modelo, somente ocorre uma nova geração de código. b) remover um determinado estado. Esta operação trará conseqüências quando houver uma instância neste estado removido. Neste caso, os seguintes procedimentos são tomados: a instância é reinicializada (executa-se o seu método start) e, se existir uma entidade temporária que esteja relacionada com esta instância, então esta entidade temporária é eliminada da simulação. Os relacionamentos não são modificados, indicando que esta instância continua normalmente sua operação após o reinício da experimentação. Após o recomeço da experimentação, deve-se tomar cuidado pois tem-se um novo período transitório da simulação até que a mesma se estabilize, ou seja, entre no regime permanente.

Também é possível que se troque o comportamento de uma classe por outro totalmente diferente, ou seja, ocorre a mudança de um diagrama de estados por outro. Neste caso, deve-se fazer uma comparação entre os dois comportamentos para verificar quais estados que permanecem e quais estados que foram retirados, para que sejam tomadas as medidas necessárias para reiniciar o experimento.

A **modificação dos atributos** de uma determinada instância, como tempos e distribuições de probabilidade, também pode ser executado durante a experimentação. Tais atributos foram definidos pelo modelador na classe a qual a instância pertence. Também podem ser modificados os atributos internos da simulação, tais como o estado em que se encontra cada uma das instâncias existentes durante a experimentação. Todas as instâncias de uma classe compartilham o mesmo diagrama de estados, mas cada uma encontra-se em um determinado estado deste diagrama. Com isto, pode-se modificar o estado em que uma determinada instância se encontra durante a experimentação. Neste caso, deve-se enviar uma mensagem de mudança de estado para as instâncias que estão relacionadas com esta instância que teve seu estado modificado. Estas alterações significam mudança de comportamento

apenas da instância selecionada. No exemplo da agência bancária, podemos modificar o tempo de atendimento de um determinado gerente/caixa, a geração de clientes passar de uma distribuição normal para exponencial, alterar o estado de um gerente de trabalho normal para operação tartaruga, etc.

A **inclusão de entidades** é uma facilidade que permite que, através dos mesmos recursos de construção do cenário na modelagem, sejam incluídas novas instâncias de entidades permanentes (recursos) durante a experimentação. A inclusão de novas entidades temporárias por intermédio do usuário não é possível, uma vez que este tipo de entidade é criada automaticamente por intermédio de um gerador. A inclusão de novas instâncias de recursos do tipo "storage" ou "facility" segue os mesmos procedimentos realizados quando da construção do cenário: posicionamento da representação visual da instância no cenário e especificação dos relacionamentos com as demais instâncias. Para o caso específico de um recurso do tipo "storage", além de incluir uma nova instância pode-se incluir mais um servidor em uma instância já existente. Neste caso, basta selecionar a instância desejada no cenário e informar quantos servidores a mais serão incluídos. A partir deste momento a "storage" é redimensionada e todas as instâncias de entidades temporárias que se encontram na fila única desta "storage" passam a reconhecer e utilizar este(s) novo(s) servidor(es). No exemplo da agência bancária, o usuário observa que a fila dos caixas está muito grande (horário do almoço) e resolve incluir mais um caixa. A partir deste momento, todos os clientes que se encontram na fila dos caixas passam a reconhecer este novo caixa.

Assim como a inclusão de entidades, existe a facilidade que permite a **remoção de entidades** do modelo durante a experimentação. No caso da remoção, tem-se que realizar tratamentos diferenciados quando da remoção de uma entidade temporária e de uma entidade permanente (recurso). Quando da remoção de uma entidade temporária, deve-se verificar seu estado atual na experimentação: se está ocupando um recurso ou se está em uma fila de atendimento. Caso ela esteja ocupando um recurso, deve ser enviada uma mensagem de final de atendimento para este recurso, fazendo com que o mesmo mude de estado conforme seu comportamento. Se a entidade temporária a ser removida estiver em uma determinada fila de atendimento, simplesmente deve-se atualizar o atributo de número de elementos nesta fila. Para a remoção de instâncias de entidades permanentes, tem-se que fazer uma diferenciação para recursos do tipo "storage" e "facility". No caso de "storages", pode-se remover a "storage" (todos os servidores) ou apenas um único servidor dela. Quando retira-se apenas um único servidor, a "storage" é redimensionada e, caso alguma entidade temporária esteja utilizando este servidor no momento, esta entidade retorna para o início da fila de atendimento. Quando remove-se a instância de uma "storage" ou "facility", o tratamento é o mesmo: todas as entidades temporárias que estão utilizando este recurso e as que estão na fila de atendimento são removidas da experimentação. Também todos os relacionamentos existentes com outras entidades são retirados do modelo. No exemplo da agência bancária, pode ser removida uma caixa (um servidor de uma "storage") pois observa-se que existem caixas ociosos, não havendo formação de fila.

Também para uma maior flexibilidade no modelo, o usuário pode **modificar os relacionamentos** existentes entre as instâncias no cenário. Esta operação permite que através da análise do modelo sendo simulado, o usuário possa alterar a dinâmica de funcionamento do modelo sem modificar sua estrutura estática e dinâmica. No exemplo da agência bancária, o usuário verifica que a fila única dos caixas está muito grande, ou seja, o tempo de espera dos clientes na fila está elevado. Para contornar este problema, o usuário pode aumentar o número de caixas disponíveis (inclusão de entidades) ou pode diminuir o fluxo de clientes nestas caixas através da modificação dos relacionamentos. Neste caso, pode-se modificar o relacionamento de algum tipo de cliente fazendo com que ele passe a se relacionar somente com outra instância de caixa (incluir um novo relacionamento) e não mais com as caixas anteriores onde havia o problema (remover um relacionamento). É evidente que esta modificação nos relacionamentos das instâncias deve ser consistente com os relacionamentos existentes entre as classes descritas no diagrama de classes.

A **modificação da estrutura estática** do modelo, ou seja, modificar o diagrama de classes, durante a experimentação é um recurso muito poderoso do ambiente *VISME* que o diferencia dos demais ambientes de simulação. Dentro deste contexto, podem ser realizadas as seguintes operações: remoção/alteração/inclusão de uma classe e/ou de um relacionamento. Quando a estrutura estática do modelo é modificada durante a experimentação, um novo código para o modelo é gerado automaticamente. Após o recomeço da experimentação, tem-se um novo período transitório, o que certamente o usuário deverá tomar cuidado na sua análise durante este período. Muitas destas modificações ocasionam reflexos imediatos no cenário.

A inclusão de uma nova classe no diagrama de classes durante a experimentação causa automaticamente uma nova geração de código, não ocasionando nenhum reflexo no cenário e na experimentação. Mas, a partir deste momento, o cenário passa a reconhecer a existência desta nova classe, ou seja, quando o usuário interagir com o cenário ele pode incluir instâncias desta nova classe. A remoção de uma classe no diagrama de classes faz com que todas as instâncias desta classe no cenário sejam removidas assim como todos os relacionamentos destas instâncias. Neste caso não existe a necessidade de gerar um novo código, apenas realizar uma seqüência de operações de remoção de instâncias e relacionamentos. Também podem ser realizadas alterações nos atributos das classes, o que certamente pode ocasionar problemas no comportamento, uma vez que os atibutos são utilizados na sua especificação. Neste contexto, pode-se alterar o atributo gráfico da representação visual da classe, refletindo imediatamente em todas as instâncias desta classe no cenário. Quanto aos relacionamentos existentes entre as classes no diagrama de classes, o usuário pode: incluir novos relacionamentos (ocasiona uma nova geração de código e, a partir deste momento, as instâncias destas classes podem se comunicar no cenário) e remover relacionamentos existentes.

Um recurso semelhante ao de remoção de entidades é o que permite **desabilitar um recurso** do modelo. A diferença entre ambos está no fato de que quando um recurso é desabilitado, as entidades temporárias que estão utilizando este recurso continuam sua operação normal. No caso da remoção de um recurso as

entidades temporárias que o estão utilizando também são removidas. Um recurso que foi desabilitado será removido somente depois de não haverem mais entidades temporárias utilizando-o (sua fila deve estar vazia). Durante este período, para as demais entidades presentes na experimentação, é como este recurso não existisse mais, ou seja, mais nenhuma entidade temporária consegue utilizar este recurso.

O **controle total do experimento** são todos aqueles recursos necessários ao acompanhamento da experimentação por parte do usuário com o objetivo de melhorar seu entendimento do sistema modelado. Estes recursos apresentam novas janelas de visualização para o usuário. Alguns exemplos: exibir os valores de um atributo de uma instância através de um histograma (recurso de associação), apresentar uma janela contendo os eventos futuros (estes podem ser modificados), alterar valores de atributos/variáveis e, controlar o modo de execução da simulação (passo a passo, até o próximo evento, até uma determinada condição, etc).

Para uma melhor apresentação visual do modelo, é possível realizar **modificações no cenário**. Este recurso permite que, durante a experimentação, o usuário altere o posicionamento das entidades, altere os caminhos especificados para as entidades temporárias e modifique a representação gráfica das entidades no cenário. Estas alterações não afetam o código gerado para a simulação.

7 Implementação

7.1 Introdução

Este capítulo tem por objetivo descrever o protótipo do ambiente *VISME* que foi desenvolvido com a finalidade de dar apoio à validação das idéias propostas neste trabalho. O protótipo é constituído por três ferramentas principais: a ferramenta de edição da estrutura do modelo (diagrama de classes), a ferramenta de descrição do comportamento (diagrama de estados) e a ferramenta de edição do cenário. Além destas três ferramentas também foi desenvolvida uma ferramenta para a análise dos resultados em pós-processamento.

Ao ser ativado, o protótipo do ambiente *VISME* apresenta a interface mostrada na figura 7.1. Esta interface realiza o gerenciamento de informações entre as ferramentas do ambiente. A interface principal do ambiente *VISME* contém um cardápio com diversas opções e uma barra de ícones com as operações mais utilizadas durante um processo de simulação no ambiente *VISME.*

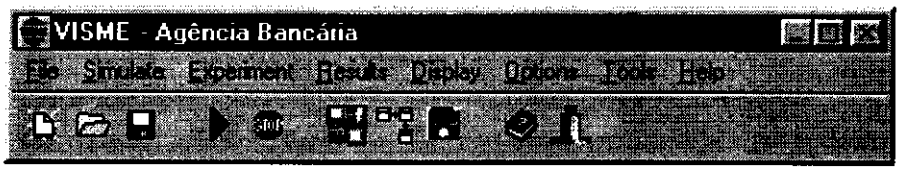


FIGURA 7.1 - Interface principal do ambiente VISME.

As ferramentas do ambiente *VISME* foram desenvolvidas mantendo-se uma padronização quanto as suas interfaces (figura 7.2). Na parte superior da interface é apresentado o nome da ferramenta e o nome do objeto que está sendo manipulado no momento. Abaixo aparece a área de cardápio e a barra de ícones específicos para cada ferramenta. A seguir vem a área de edição onde é construído o objeto manipulado pela ferramenta. A parte inferior é destinada às mensagens enviadas pela ferramenta para auxiliar sua operação.

O restante deste capítulo está organizado da seguinte forma: as características das ferramentas de edição da estrutura do modelo e de descrição do comportamento são descritas primeiramente. Esta descrição está limitada aos principais recursos disponíveis para o usuário, uma vez que detalhes de implementação destas ferramentas não são relevantes no contexto deste texto. A seguir é apresentada a biblioteca de classes para a simulação utilizada na descrição do comportamento dos objetos. Após, são apresentadas as características das ferramentas de construção do cenário e de análise dos resultados em pós-processamento. Finalmente, é apresentada com maiores detalhes de implementação a geração do código para a execução de um modelo de simulação.

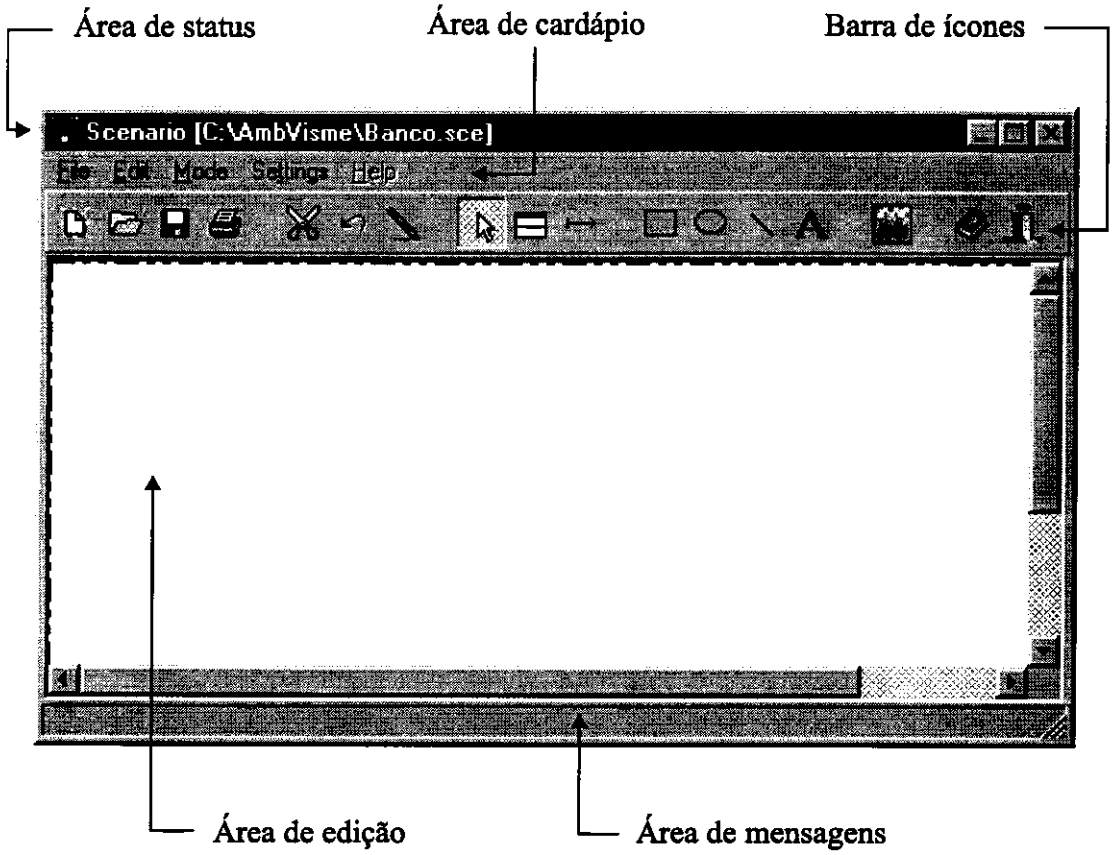


FIGURA 7.2 - Padrão de interface das ferramentas do ambiente VISME.

Com o propósito de facilitar a apresentação das principais características de cada uma das ferramentas, estas serão explicadas através do exemplo da agência bancária apresentado no capítulo 2.

7.2 Ferramenta de edição da estrutura do modelo

A ferramenta de edição da estrutura estática do modelo está baseada no diagrama de classes descrito no capítulo 6. Como comentado anteriormente, o diagrama de classes descreve as diferentes classes que irão constituir o modelo e os possíveis relacionamentos entre elas. A figura 7.3 apresenta a interface do editor que é ativado através da interface principal do ambiente *VISME* (figura 7.1).

Normalmente a construção de um modelo no ambiente *VISME* começa pela construção do diagrama de classes correspondente. Uma vez que o modelo é hierárquico, primeiramente deve-se definir a classe de mais alto nível conforme a figura 7.3. A classe Banco representa todo o modelo, incluindo todas as demais classes. Para a criação de uma classe deve-se selecionar o ícone correspondente na barra de ícones (figura 7.4) e indicar a posição desejada na área de edição. Pode-se observar na figura 7.4 os ícones de navegação nos subdiagramas que permitem

navegar pela hierarquia de agregações do modelo. Caso seja solicitado um componente que não tenha subdiagrama, então um novo subdiagrama vazio é criado.

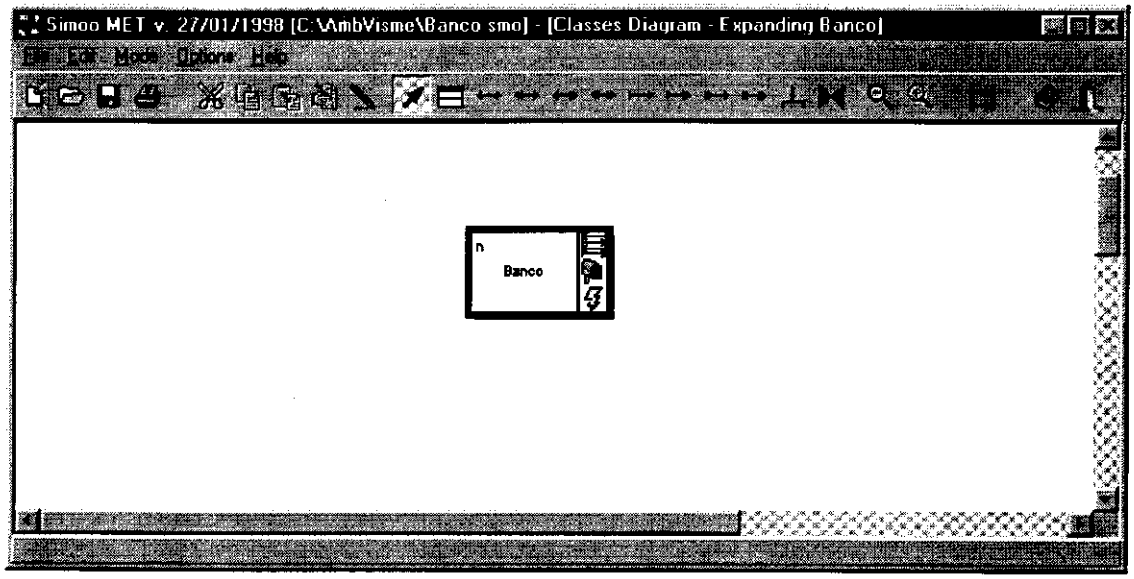


FIGURA 7.3 - Ferramenta para o diagrama de classes.

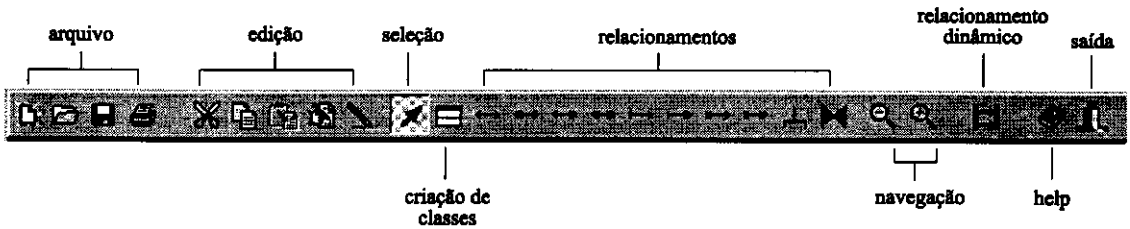


FIGURA 7.4 - Barra de ícones da ferramenta do diagrama de classes.

Para cada classe criada é apresentada a janela da figura 7.5 onde devem ser informadas as características da classe quando de sua criação. Nesta janela é necessário informar o nome da classe, as abordagens que compõem o paradigma (recepção das mensagens, forma de comunicação e descrição do comportamento) e seu tipo (permanente ou temporária). Após fornecer estas informações, a representação gráfica correspondente à classe recém criada é inserida no diagrama. Em qualquer momento do processo de simulação, é possível alterar a posição da classe no diagrama e suas características.

Após a definição das classes que comporão o modelo, devem ser realizados os relacionamentos entre as classes. Os relacionamentos são feitos utilizando-se os ícones existentes na figura 7.4 e selecionando-se as duas classes envolvidas. Com isto é desenhada uma seta ligando as duas classes. Caso seja necessária uma mudança no traçado do relacionamento, devem ser definidos diversos

pontos intermediários, formando segmentos de reta, antes de definir a classe destino. Durante a construção do diagrama o traçado do relacionamento pode ser modificado através da seleção e mudança de posição dos pontos que formam os segmentos de reta do relacionamento. A mudança de posição de uma classe envolvida no relacionamento também modifica o traçado do mesmo, sendo que, obviamente, o traçado do relacionamento somente tem influência numa melhor apresentação do diagrama. Para o modelo o que interessa são as duas classes envolvidas (origem e destino) e o tipo de relacionamento realizado conforme o ícone selecionado.

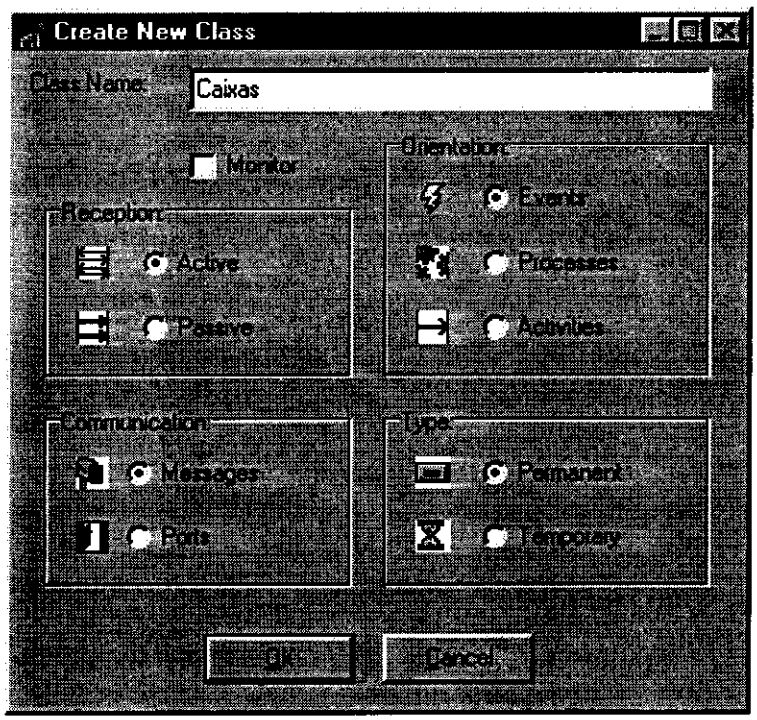


FIGURA 7.5 - Janela de criação de classes.

Através da seleção de uma determinada classe, podem ser realizadas diversas tarefas através da ativação de um cardápio específico (figura 7.6): vincular/visualizar a representação visual da classe (utilizada no cenário), vincular/modificar o comportamento da classe, expandir a classe para visualizar seu subdiagrama, etc.

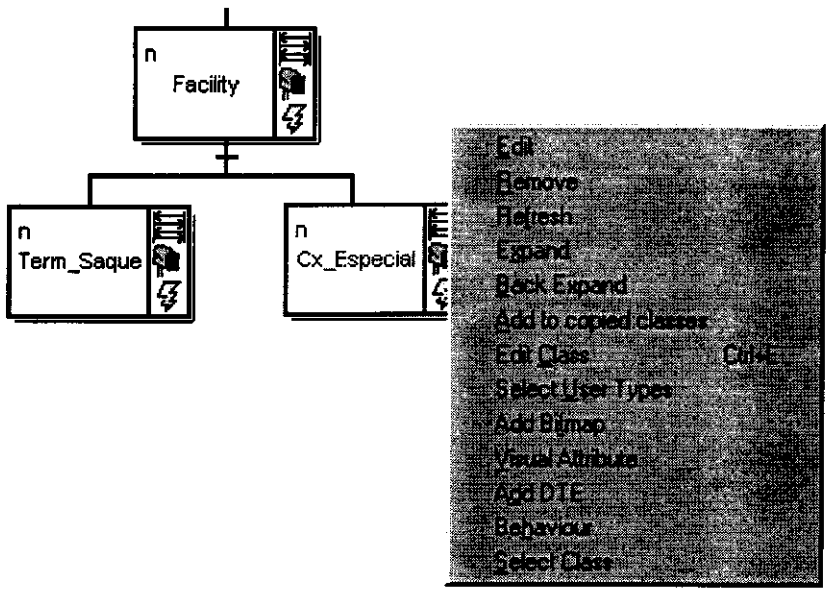


FIGURA 7.6 - Cárdápio de opções para uma classe selecionada.

7.3 Ferramenta de descrição do comportamento

A ferramenta de descrição do comportamento está baseada no diagrama de estados descrito no capítulo 6. Através desta ferramenta pode-se descrever o comportamento (estrutura dinâmica) de cada uma das classes descritas no diagrama de classes. A figura 7.7 apresenta a interface do diagrama de estados. O diagrama de estados pode ser ativado de duas formas: através da interface principal do ambiente *VISME* (figura 7.1) ou através da seleção de uma classe no diagrama de classes e selecionando a opção correspondente do cardápio ativado (figura 7.6).

A descrição do comportamento no ambiente *VISME* é realizada através de duas operações: especificação dos estados e definição das transições e suas propriedades. Para a criação de um estado qualquer seleciona-se o ícone correspondente ao estado desejado na barra de ícones (figura 7.8) e posiciona-se o estado na área de edição. A seguir, deve-se informar o nome do estado e qual a sua representação visual, se desejado, utilizada no cenário com o propósito de animação icônica. Quando se cria um estado inicial ou final, também deve-se especificar a transição associada a este estado, ou seja, deve-se definir o que acontece com o objeto no momento de sua criação e destruição.

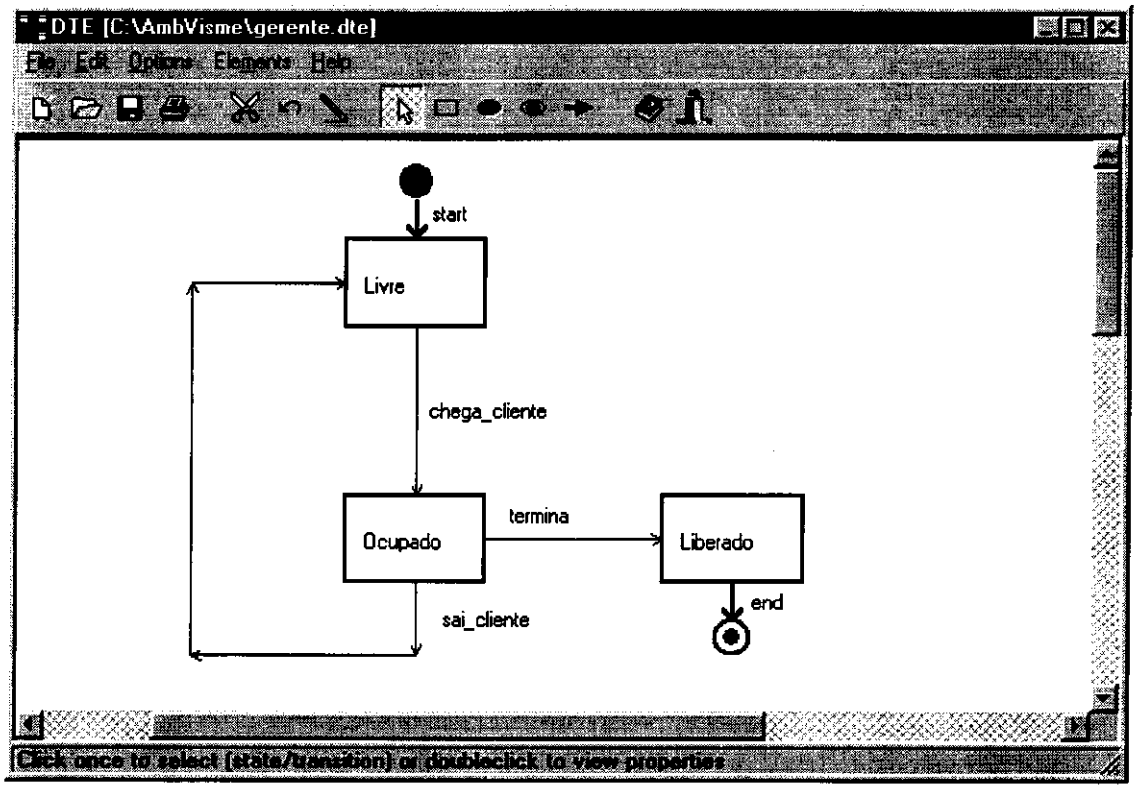


FIGURA 7.7 - Ferramenta para o diagrama de estados.

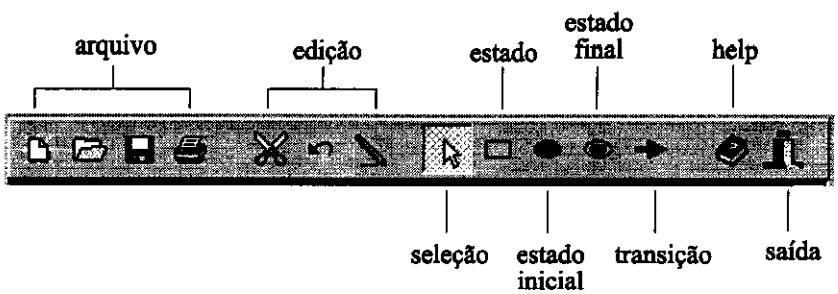


FIGURA 7.8 - Barra de ícones para a ferramenta de diagrama de estados.

Após a definição dos estados, definem-se as transições existentes entre os estados. Para definir uma transição, seleciona-se o ícone correspondente na barra de ícones e selecionam-se os dois estados envolvidos. Com isto é desenhada uma seta que liga o estado origem ao estado destino. Caso seja necessária uma mudança no traçado da transição, utiliza-se a mesma filosofia empregada para os relacionamentos no diagrama de classes. Para cada transição criada é apresentada a janela de propriedades da transição conforme a figura 7.9.

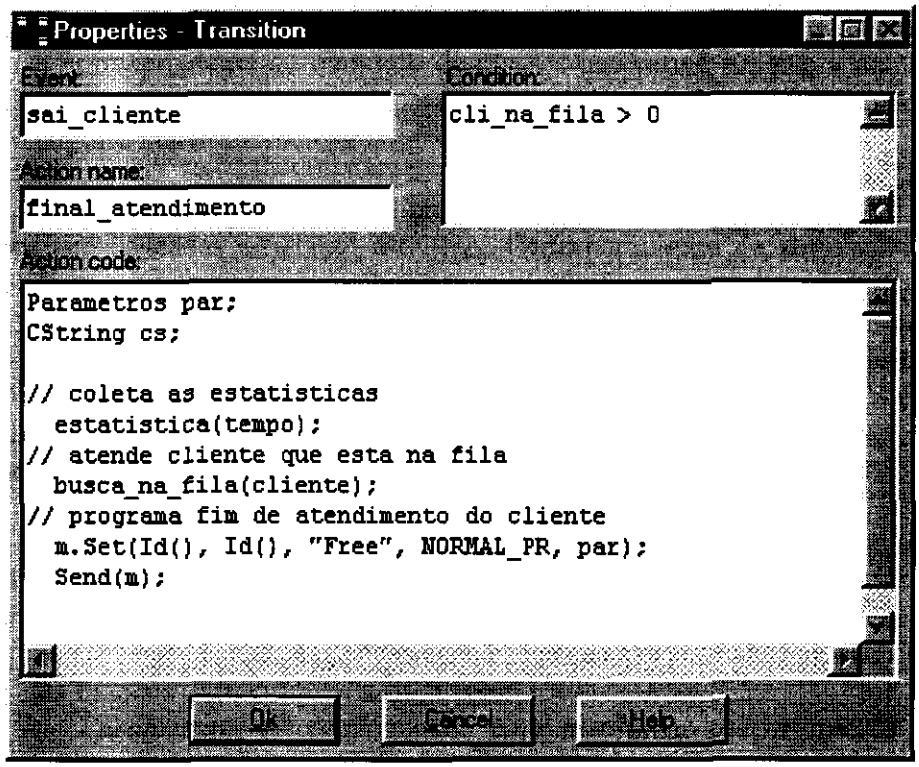


FIGURA 7.9 - Janela de propriedades de uma transição.

Na janela de propriedades da transição são informados o nome do evento que dispara a transição, a condição de guarda utilizando a sintaxe de C++, o nome da ação (método) que será executada quando a transição disparar, e o código da ação. Este código é escrito em C++ juntamente com as classes disponíveis na biblioteca de classes para a simulação, que será descrita na próxima seção. Neste código é descrita a interface da classe, ou seja, o conjunto de mensagens que ela é capaz de responder e o código que deve ser acionado para o tratamento destas mensagens.

7.4 Biblioteca de classes para simulação

A biblioteca de classes para simulação [COP97] contém as classes que implementam os diferentes paradigmas disponíveis no "framework" SIMOO utilizados para descrever o comportamento da classe. Esta biblioteca de classes utiliza o conceito de elemento autônomo que é um objeto ativo com "thread" própria de execução e uma fila de mensagens. Isto significa que, por as entidades serem autônomas, o usuário pode interagir com elas durante a simulação, quer para interferir em seu comportamento ou simplesmente para obter alguma informação delas.

Esta biblioteca de classes possui um conjunto de classes e funções que são necessárias para a construção do modelo. Desta forma, o usuário não se preocupa com a implementação de diversos elementos básicos da simulação, tais como: suporte para troca de mensagens, programação de eventos, tempo simulado, geração de números aleatórios, distribuições de probabilidade, coleta de estatísticas, visualização de

resultados, etc. Estas classes podem ser agrupadas em dois grandes conjuntos: a) mensagens: correspondente ao conjunto de classes que provê suporte a manipulação de strings e troca de mensagens, e; b) paradigmas: corresponde ao conjunto de classes que provê suporte aos diferentes paradigmas de simulação previstos.

As classes básicas da biblioteca que dão suporte ao sistema de troca de mensagens são baseadas na classe MSGEA. Cada paradigma de simulação oferece métodos distintos para troca de mensagens conforme suas características. O argumento destes métodos em qualquer um dos paradigmas é sempre um objeto da classe MSGEA. Para que dois objetos possam trocar mensagens é necessário que: a) exista um relacionamento definido entre suas instâncias no cenário; e b) o elemento autônomo que deseja enviar a mensagem conheça o identificador do elemento autônomo destino.

A seguir será apresentada uma breve descrição de cada uma das classes básicas da biblioteca. Diversos métodos são oferecidos para cada uma destas classes, podendo ser analisados mais detalhadamente em [COP97].

- *Cstring*: oferece um conjunto de métodos para a manipulação de strings baseados nas funções oferecidas pela linguagem C++.

- *Parametros*: oferece recursos para manipular os parâmetros de uma mensagem, uma vez que o número de argumentos pode ser variado conforme o tipo de mensagem trocada entre as entidades. Esta classe utiliza uma instância de Cstring para armazenar os argumentos sob a forma de strings. Cada um dos argumentos é referenciado através de um índice.

- *MSGEA*: representa uma mensagem de elemento autônomo. As mensagens trocadas entre as entidades do modelo são sempre instâncias da classe MSGEA. Os seguintes elementos compõem uma mensagem: identificador da origem, identificador do destino, identificador da mensagem, prioridade da mensagem e lista de parâmetros (lista de argumentos representada por uma instância da classe Parametros).

- *REFLIST*: trata com a manipulação de armazenamento dos identificadores das classes, contendo uma lista de Cstrings.

- *RN*: responsáveis pela geração de números pseudo-aleatórios. Possui métodos para geração de sementes e geração de seqüência de valores no intervalo [0,1].

- *DP*: responsável pela geração de números aleatórios segundo distribuições de probabilidade. As seguintes distribuições de probabilidade são oferecidas: Uniforme, Exponencial, Erlang, Gama, Weibull, Normal, Logonormal, Beta, Pearson tipo V, Pearson tipo VI, Triangular, Bernoulli, Binomial, Geométrica, Binomial negativa e Poisson. Todas elas foram implementadas segundo [LAW91].

O comportamento de um elemento autônomo é descrito utilizando-se orientação a eventos e comunicação por mensagens com receptor semi-ativo [COP97]. A classe que implementa o elemento autônomo (classe EA) constitui o topo da hierarquia de classes que implementa os paradigmas de simulação. Esta classe implementa o paradigma básico a partir do qual os demais paradigmas são derivados. Algumas das classes utilizadas para descrever as entidades de um modelo são (todas utilizam receptor semi-ativo): classe EVMSG (orientação a eventos e comunicação

por mensagens); classe PROCMSG (orientação a processos e comunicação por mensagens); classe EVPORTA (orientação a eventos e comunicação por portas); e, classe PROCPORTA (orientação a processos e comunicação por portas).

Uma vez que o ambiente *VISME* irá tratar de problemas discretos onde haja formação de filas, todas as classes que comporão um modelo em *VISME* irão utilizar o paradigma descrito pela classe EVMSG. Desta forma, a troca de informações entre as entidades no modelo somente será possível através de mensagens. Estas mensagens podem ser síncronas, quando a entidade emissora aguarda o tratamento da mensagem por parte da entidade receptora, ou podem ser assíncronas, quando a entidade emissora não necessita aguardar que a entidade receptora trate a mensagem. As entidades serão consideradas de receptor semi-ativo por possuirem uma fila de mensagens. Isto representa que a entidade emissora não determina quando a entidade receptora irá atender sua mensagem, pois esta mensagem será colocada numa fila de atendimento.

Cada uma das classes possui um conjunto de rotinas que podem ser utilizadas na descrição do comportamento das entidades. A classe EA também possui um conjunto de rotinas, e por ser ela a classe da qual todas as demais são derivadas, suas rotinas são visíveis por toda a hierarquia de classes. Suas rotinas são:

- CString Id(void): retorna o identificador do elemento autônomo.
- CString IdPai(void): retorna o identificador do elemento autônomo agregador.
- CString MetaClass(void): retorna o identificador da meta-classe.
- TIME Clock(void): retorna o tempo de simulação corrente.
- void Quit(void): solicita a destruição do elemento autônomo.
- int CancelEvent(TIME t, MSGEA m): permite o cancelamento de um evento futuro programado pelo próprio elemento autônomo.

A classe EVMSG que é derivada diretamente da classe EA, possui rotinas que permitem a programação de eventos futuros e consultas síncronas instantâneas. Suas rotinas são:

- int Send(TIMEt, MSGEA m): permite programar um evento m para um tempo futuro t.
- int Send(MSGEA m): permite programar um evento m para o tempo corrente.
- int SendReceiveNow(MSGEA *m): permite efetuar uma consulta síncrona instantânea a outro elemento autônomo.

7.5. Ferramenta de edição do cenário

A ferramenta de edição do cenário é utilizada para a construção da representação visual do modelo. Este editor gráfico é um diagrama de instâncias que define um sistema em particular através da criação de instâncias das classes definidas no diagrama de classes e os relacionamentos entre estas instâncias. A figura 7.10 apresenta a interface da ferramenta que é ativada através da interface principal do ambiente *VISME* (figura 7.1).

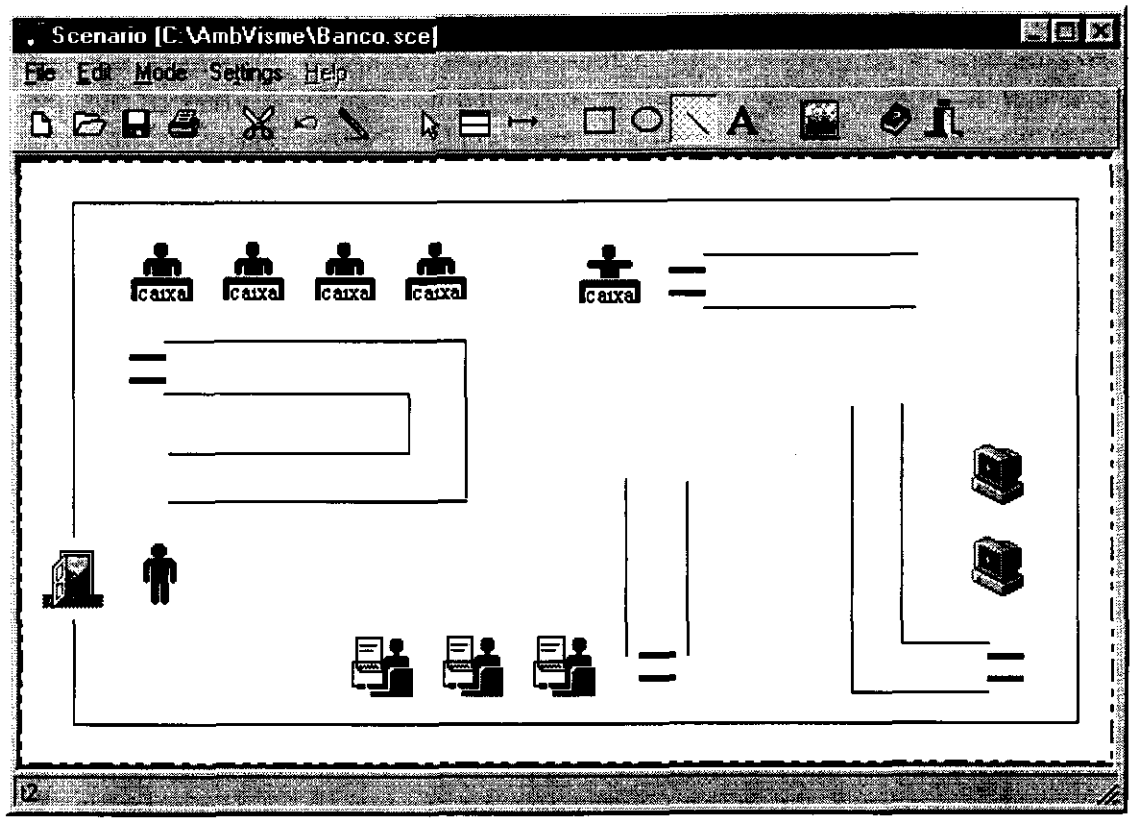


FIGURA 7.10 - Ferramenta de edição do cenário.

Na construção do cenário devem ser definidos o fundo estático, visando obter um maior realismo do modelo e a dinâmica do modelo através das representações visuais e relacionamentos dos objetos definidos no diagrama de classes. Utilizando-se dos recursos oferecidos na barra de ícones (figura 7.11), e através das operações de seleção e posicionamento define-se o fundo estático do modelo. Dentre os recursos disponíveis para a criação da parte estática do modelo destacam-se a seleção de "bitmaps" a partir de bibliotecas de ícones e recursos gráficos para criação de linhas, formas, textos, etc.

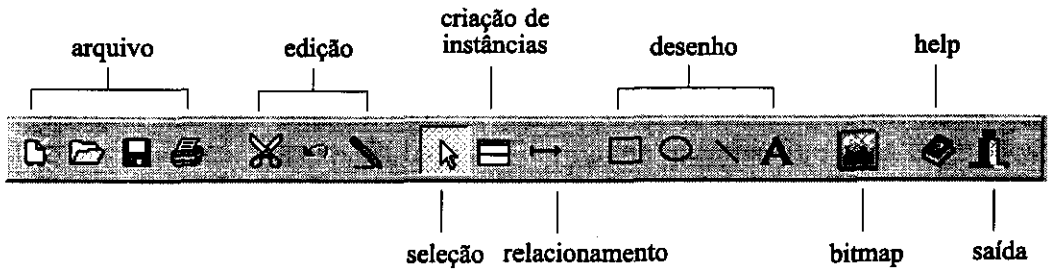


FIGURA 7.11 - Barra de ícones da ferramenta de edição do cenário.

O próximo passo na construção do cenário é a definição de sua parte dinâmica, ou seja, quais as instâncias que participarão do modelo, o posicionamento

de suas representações visuais no cenário e o seus relacionamentos com as demais instâncias do modelo. A criação de uma instância é feita a partir da seleção do ícone adequado na barra de ícones e a definição de sua posição no cenário. A janela de criação de instâncias (figura 7.12) é apresentada para cada instância criada devendo-se selecionar a classe da qual se quer criar uma nova instância e o nome da instância. A lista de classes apresentada nesta janela, com a respectiva representação visual de cada uma quando selecionada, refere-se às classes definidas no diagrama de classes. Esta janela é sensível às mudanças realizadas no diagrama de classes, ou seja, ela terá suas informações atualizadas sempre que houver uma modificação no diagrama de classes.

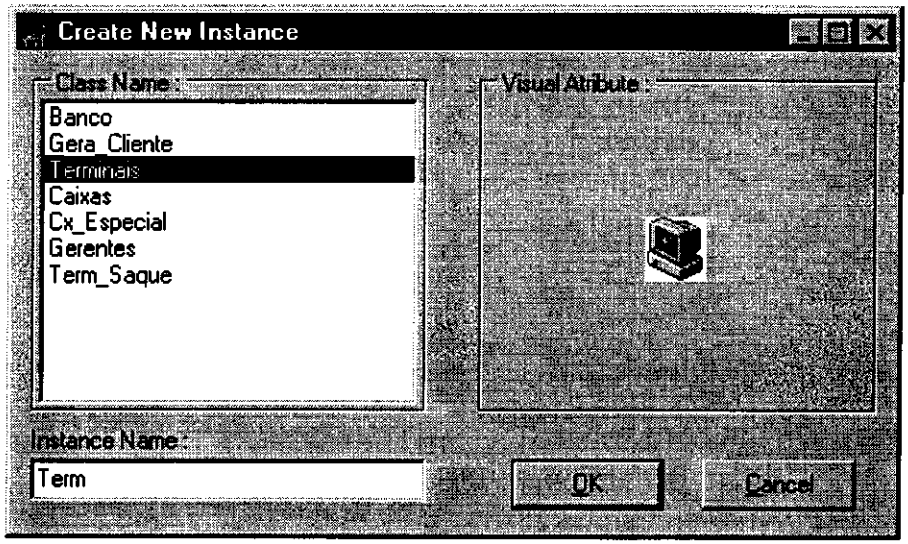


FIGURA 7.12 - Janela de criação de instâncias.

Como descrito no capítulo 6, o modelo é descrito através dos recursos que ele irá oferecer, sendo estes recursos pertencentes as classes "facility" ou "storage". Quando se define a classe da qual será criada uma instância, automaticamente já se conhece a que tipo de recurso que ela pertence. Caso seja um recurso do tipo "storage" é apresentada a janela da figura 7.13. Nesta janela deve-se especificar o número de servidores que atuarão em paralelo e a sua disposição gráfica de visualização da instância, se horizontal ou vertical. Isto se deve ao fato de que a representação visual de uma "storage" é um conjunto de ícones (representação visual da classe selecionada) dispostos lado a lado, tantos quantos forem os especificados nesta janela. Estas propriedades de uma "storage" podem ser modificadas a qualquer momento da experimentação.

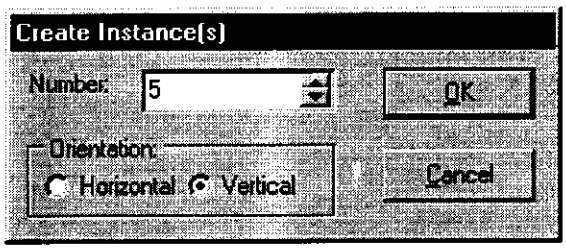


FIGURA 7.13 - Janela de especificação de um recurso "storage".

Finalmente, devem ser especificados os relacionamentos entre as instâncias definidas no cenário. Após a seleção do ícone adequado na barra de ícones, devem ser selecionadas as duas instâncias envolvidas no relacionamento. Neste ponto é realizada uma consistência para verificar se as classes das instâncias selecionadas possuem um relacionamento de conhecimento no diagrama de classes. Caso este relacionamento seja possível, deve-se especificar a trajetória (caminho) no cenário que deve ser seguida pela representação gráfica da instância (entidade temporária dinâmica). Esta trajetória é definida utilizando-se a mesma filosofia empregada para os relacionamentos no diagrama de classes e as transições no diagrama de estados.

7.6 Ferramenta de análise de resultados

A ferramenta de análise de resultados do ambiente *VISME* permite que os resultados da simulação sejam analisados em pós-processamento. Desta forma, durante a experimentação devem ser selecionadas através do recurso de monitoramento as características do modelo (variáveis e/ou atributos das entidades) que serão analisadas. Estas informações serão armazenadas na base de dados onde após o término da experimentação poderão ser analisadas tanto grafica como estatisticamente. A figura 7.14 apresenta a interface da ferramenta de análise de resultados, sendo ativada através da interface principal do ambiente *VISME*.

Esta ferramenta foi desenvolvida de forma independente de modo que possa ser utilizada por outros ambientes. Para isto, tais ambientes devem gerar um arquivo de resultados com o formato reconhecido por esta ferramenta. Este arquivo deverá conter uma lista de variáveis/atributos monitorados, estando descritos o tempo de simulação, o nome da variável/atributo e os respectivos valores para cada tempo como mostra a figura 7.15.

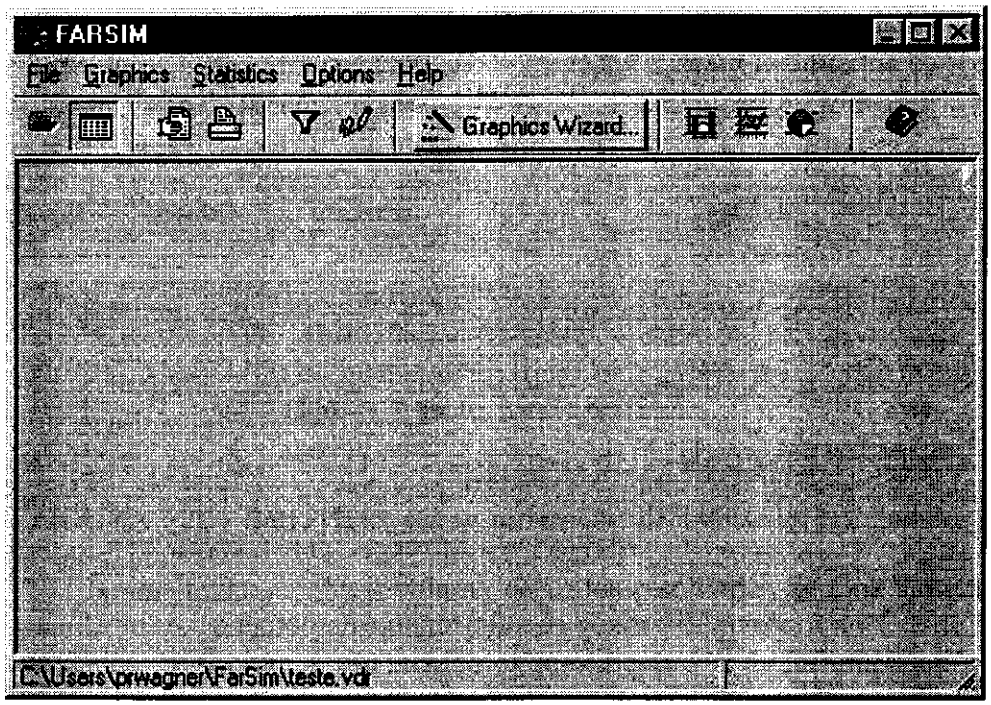


FIGURA 7.14 - Interface da ferramenta de análise de resultados.

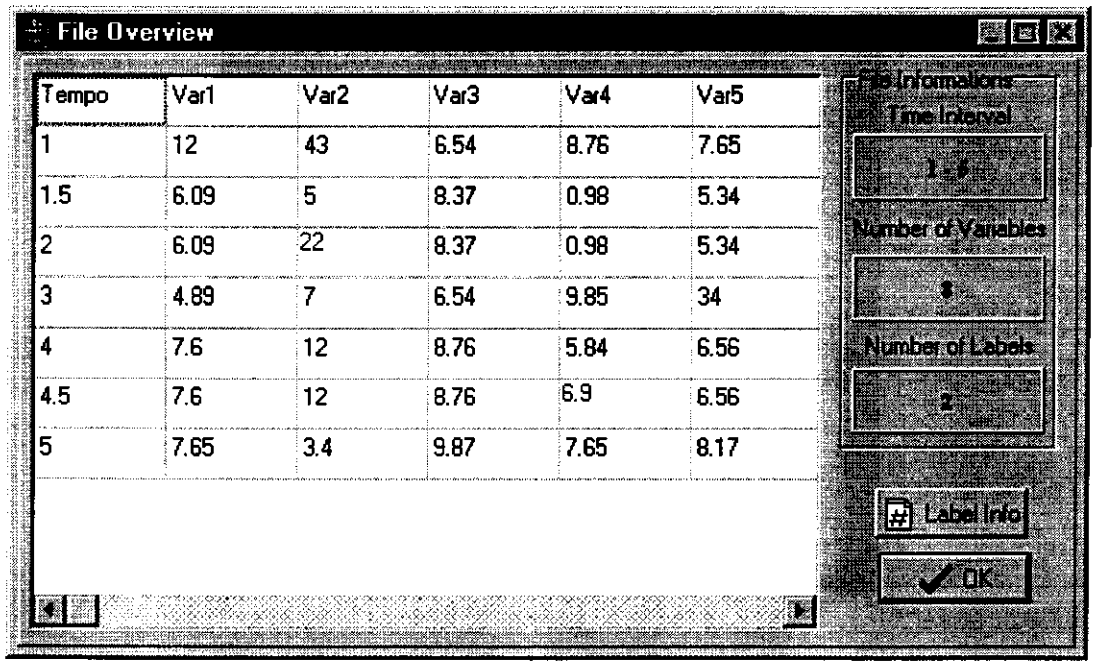


FIGURA 7.15 - Janela com o arquivo de resultados.

Inicialmente para realizar a análise de resultados deve-se selecionar na base de dados qual o arquivo de resultados que se quer analisar, uma vez que podem

ter sido criados vários arquivos de resultados para um determinado modelo. A partir deste momento os dados armazenados são carregados e ficam habilitados os recursos disponíveis na barra de ícones (figura 7.16).

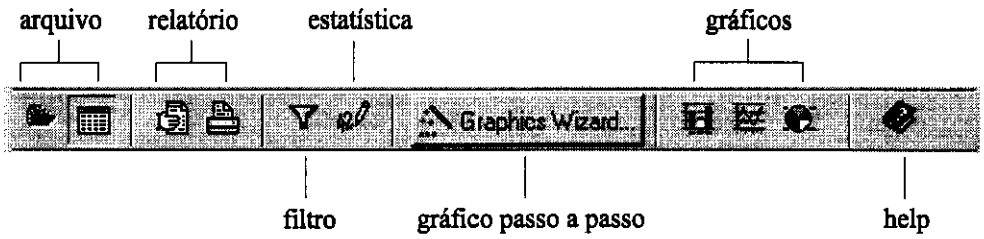


FIGURA 7.16 - Barra de ícones da ferramenta de análise de resultados.

Normalmente, como o volume de dados resultantes de uma simulação é muito grande, torna-se extremamente difícil e às vezes entediante, a análise de toda a massa de dados. Desta forma, um recurso extremamente útil da ferramenta é o filtro (figura 7.17) que permite reduzir a massa de dados através da seleção de um conjunto de entidades ou parte dos dados. O recurso do filtro permite que sejam selecionadas somente as variáveis de interesse e/ou o período de tempo da simulação no qual se deseja analisar os dados. A partir deste momento, os dados a serem analisados são aqueles resultantes deste processo de filtragem.

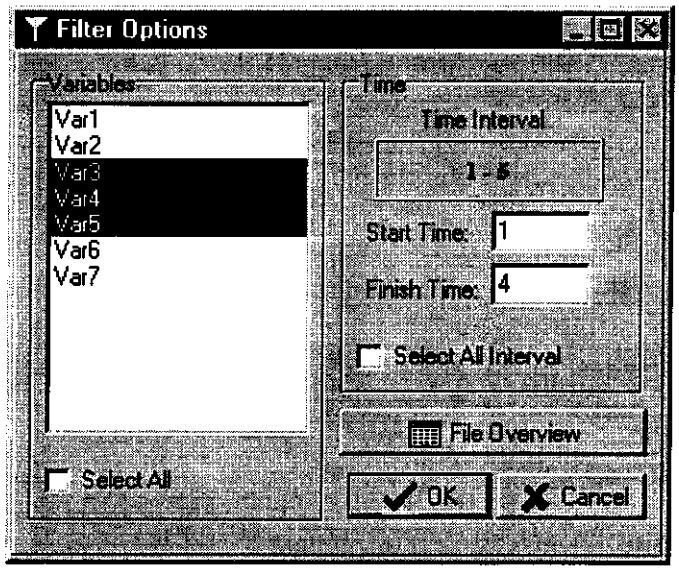


FIGURA 7.17 - Janela do recurso filtro.

A análise de resultados poderá ser realizada através de gráficos e/ou estatísticas. Para a análise gráfica, a ferramenta permite a possibilidade da geração de três tipos de gráficos: gráfico de linhas, gráfico de barras e gráfico de torta. Após a seleção dos dados que se quer analisar na massa de dados, indica-se na barra de

ícones o tipo de gráfico. Para aumentar o entendimento dos resultados, para um mesmo conjunto de dados selecionado pode-se ter diferentes gráficos, podendo desta forma realizar um confronto entre os mesmos. A figura 7.18 apresenta um gráfico de barras para um conjunto de dados selecionados. Para cada tipo de gráfico existe um conjunto atributos que permite configurar a sua forma de visualização.

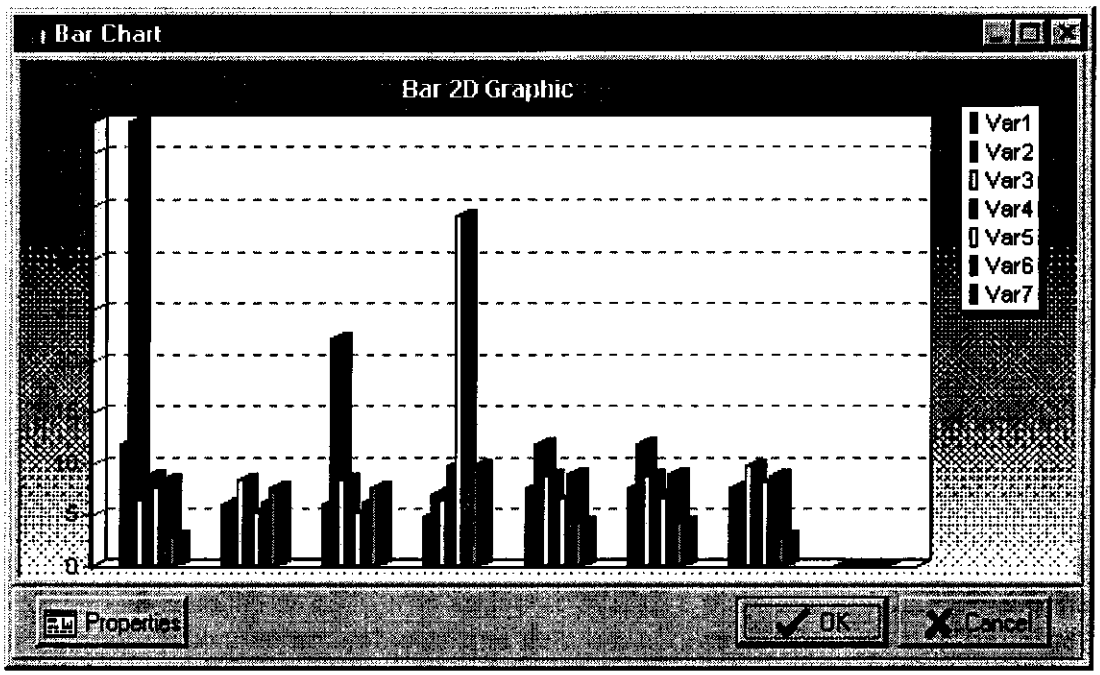


FIGURA 7.18 - Gráfico de barras para análise de resultados.

Na parte de análise estatística dos resultados a ferramenta apresenta dados como média, variância, desvio padrão, mínimos e máximos, tornando-a assim com dados suficientes para uma análise razoável dos comportamentos dos modelos. A figura 7.19 apresenta os dados estatísticos sobre o conjunto de dados selecionados.

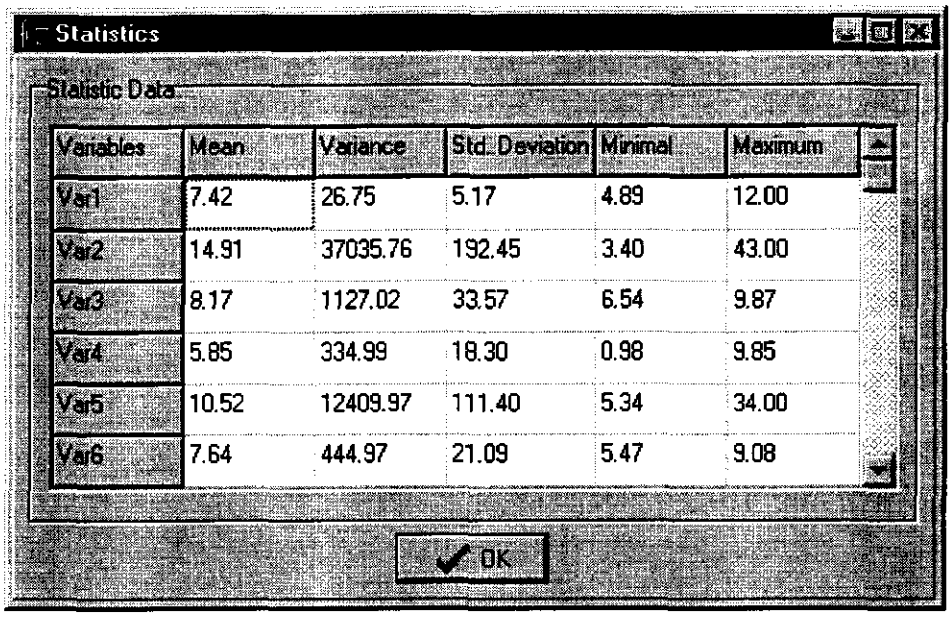


FIGURA 7.19 - Estatísticas da análise de resultados.

7.7 Geração do modelo executável

A geração de código do ambiente *VISME* é realizada automaticamente por um tradutor disponível no ambiente que gera um código de simulação na linguagem C++ a partir da especificação gráfica do modelo. O tradutor é ativado sempre que o processo de experimentação é disparado, ou seja, quando o usuário interrompe a experimentação e realiza uma modificação no modelo, um novo código será gerado. Como foi dito anteriormente, o código gerado é montado a partir das 3 principais ferramentas de modelagem: diagrama de classes, diagrama de estados e cenário.

A seguir será tratado com mais detalhes o código gerado por cada uma das ferramentas de modelagem e a biblioteca de classes da simulação. Para cada uma das ferramentas será feita uma análise dividida em duas partes: a estrutura de dados utilizada pela ferramenta e como o código é gerado a partir da mesma.

7.5.1 Diagrama de classes

7.5.1.1 Estrutura de dados

A estrutura de dados do diagrama de classes pode ser vista como uma estrutura na forma de árvore. Ela foi elaborada tendo como base o mecanismo hierárquico de descrição de classes do diagrama. Desta forma, uma classe pode conter várias outras classes, gerando uma estrutura em árvore como pode ser vista na figura 7.20.

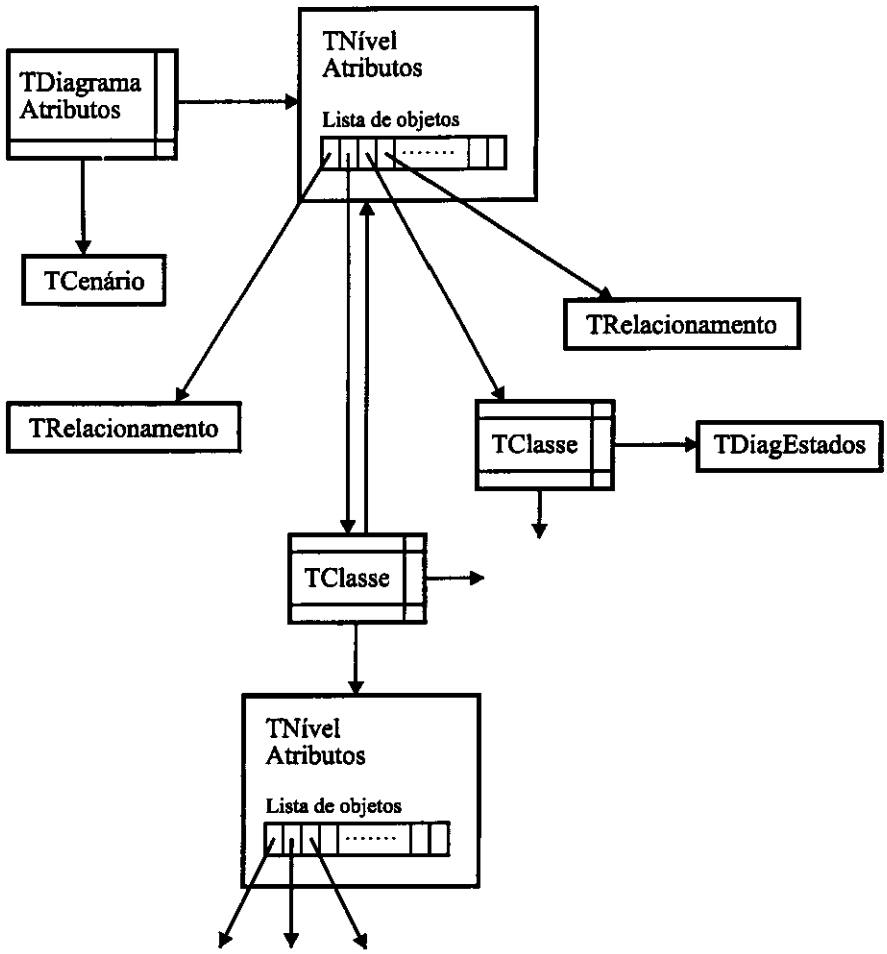


FIGURA 7.20 - Estrutura de dados para o diagrama de classes.

Para o controle de todo o fluxo do programa é utilizada a classe global TDiagrama que tem acesso a toda a estrutura do diagrama. Esta classe contém informações sobre todos os níveis, podendo obter o primeiro nível do diagrama e a referência ao nível atual, e uma série de atributos utilizados para a operação do diagrama. Cada nível é descrito através da classe TNível que contém uma lista de todos os objetos que se encontram neste nível e um conjunto de atributos necessários à descrição do nível. São considerados objetos em um nível as classes, os relacionamentos entre estas classes e os pontos de conexão das classes. Cada classe em um nível pode ser detalhada, de modo que ela deve apontar para uma classe TNível que contém seus objetos. Este detalhamento pode ser realizado para outra classe que já foi detalhada e assim por diante, fazendo com que a estrutura tenha um aspecto de árvore, com infinitos níveis. Também cada classe contém uma referência para TDiagEstados que contém a estrutura de dados do diagrama de estados que descreve o comportamento desta classe.

Para dar suporte ao mecanismo hierárquico do diagrama, são utilizadas outras estruturas de dados auxiliares. Destas, vale destacar a classe TPilha que implementa uma estrutura de pilha necessária para a implementação do detalhamento

de classes. Esta estrutura é necessária uma vez que determinadas classes (as ditas classes fantasmas) que são cópias de certas classes, podem ser colocadas em qualquer nível do diagrama. Deste modo, quando fosse necessário voltar o detalhamento, o diagrama voltaria para o nível da classe original e não o nível que se estava anteriormente no diagrama. A estrutura de pilha resolve este problema. Antes de ser detalhado, o nível corrente é colocado na pilha. Assim, quando for necessário voltar o detalhamento, basta retirar o elemento do topo da pilha para buscar o nível correto.

7.5.1.2 Código gerado

O código gerado para um diagrama de classes não é uma tarefa trivial. O gerador deve percorrer a estrutura de dados descrita na seção anterior e montar o código que pode ser dividido em 2 partes: programa principal e descrição das classes.

No programa principal primeiramente deve ser criado o gerenciador de elementos autônomos. Nele é realizado o cadastramento de todas as classes descritas no modelo (independente do nível) e todos os relacionamentos de conhecimentos possíveis no modelo. Note que também devem ser cadastrados os relacionamentos entre classes que são subclasses de superclasses que estão relacionadas. O próximo passo é criar a meta-classe, que é a classe responsável por toda a comunicação entre classes. A meta-classe tem o conhecimento de todas as classes e seus atributos. Por isso é realizado o cadastramento das classes e suas estruturas na meta-classe. Após a disponibilização da meta-classe é iniciada a simulação propriamente dita.

Antes da definição de todas as classes da simulação o gerador realiza a configuração de diversas constantes e apresenta todos os tipos definidos pelo usuário quando ele constrói o diagrama de classes. Todas as classes seguem a mesma metodologia para a geração de seu código:

1) cada classe deve ser derivada de alguma classe principal dos paradigmas de simulação propostos [COP96] (PROCMSG, EVMSG, PROCPORTA, EVPORTA);

2) são definidos os atributos declarados pelo usuário e os atributos existentes através dos relacionamentos;

3) aparece a declaração de todos os métodos existentes para a classe. Estes podem ser divididos em 3 partes:

3a) os métodos fixos: são aqueles que existem em todas as classes, a saber:

START: é o método executado quando da criação de uma instância de um objeto. Ele é composto por duas partes distintas: as inicializações que são montadas automaticamente pelo gerador e o código fornecido pelo usuário quando da definição da classe no diagrama de estados;

TRANSLATEMSG: é, na realidade, uma seqüência de estruturas de seleção, uma para cada mensagem, que a classe pode receber. Para cada uma, um método é ativado. As mensagens se dividem em fixas (uma para cada método fixo), em variáveis (uma para cada método variável) e mais aquelas que ativam os métodos definidos pelo usuário. Dentro do método TRANSLATEMSG deve ser acionado o método da classe pai, uma vez que seus atributos podem ser herdados;

RECEBEIDPONTO: fornece o identificador do ponto de conexão solicitado;

SIMOO_GET_ATTRS: informa os atributos que possui;

SIMOO_GET_MSGS: informa as mensagens que pode processar;

3b) os métodos variáveis: são os métodos referentes aos atributos definidos pelo usuário. Para cada atributo existem 2 métodos:

SIMOO_GET_atributo: obtém o valor do atributo definido no nome do método;

SIMOO_SET_atributo: coloca um valor no atributo definido no nome do método;

3c) os métodos definidos pelo usuário: são métodos que o usuário define escrevendo o seu código em C++ nas outras duas categorias.

Todos os métodos recebem uma mensagem como parâmetro (MSGEA m), tratam-na internamente utilizando o conjunto de classes básicas descrito na seção 4.2 e depois enviam uma resposta adequada a sua chamada (SEND).

7.5.2 Diagrama de estados

7.5.2.1 Estrutura de dados

Os diagramas de estado são descritos por grafos utilizando como representação uma lista de adjacência. Esta representação é formada por uma lista encadeada contendo todos os estados existentes no diagrama. Para cada um destes estados existe uma lista encadeada contendo todas as transições que partem dele, ou seja, as transições onde este estado é origem. A figura 7.21 mostra a estrutura de dados para o diagrama de estados.

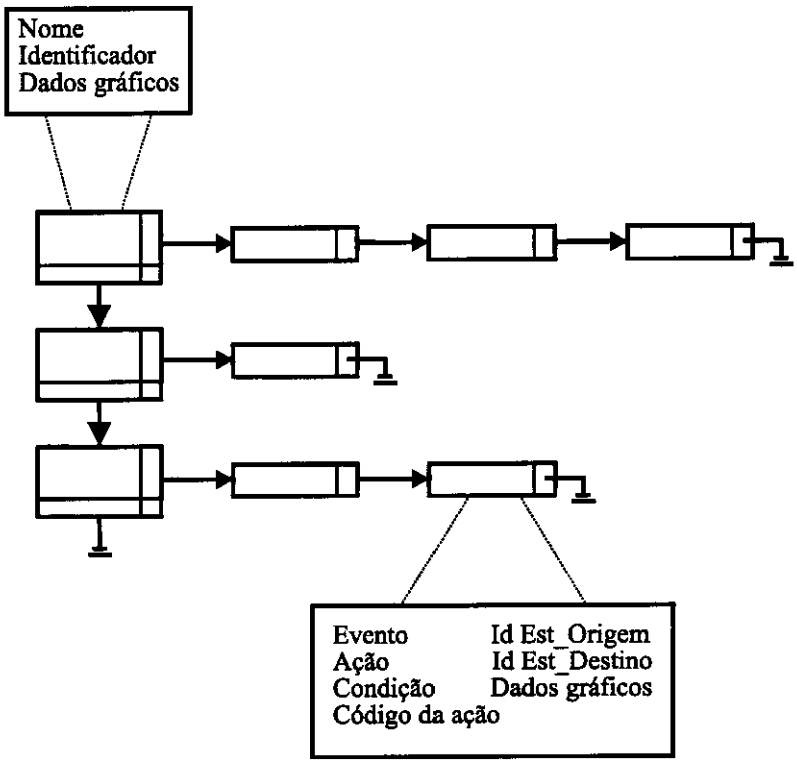


FIGURA 7.21 - Estrutura de dados para o diagrama de estados.

Cada estado descrito contém as informações de seu nome, seu identificador e outras informações relativas à parte gráfica do diagrama. Já as transições contém seu identificador, os identificadores do estado origem e do estado destino, o evento associado, qual a ação a ser executada quando a transição disparar, a condição de guarda (se houver), o código descrito em C++ da ação e outras informações úteis na parte gráfica do diagrama, como por exemplo, a lista de coordenadas x,y que representam a transição.

7.5.2.2 Código gerado

O código gerado para um diagrama de estados é bem simples e, parte da seguinte premissa: cada evento existente no diagrama tem associado uma rotina própria. Desta forma, a primeira tarefa a ser realizada é percorrer a estrutura que representa o diagrama de estados e montar uma lista com todos os eventos existentes. Para o diagrama de estados da figura 6.7 que descreve a classe gerente, pode-se notar a existência de somente 2 eventos: CHEGA_CLIENTE e FIM_ATENDIMENTO.

Isto representa que para este diagrama duas rotinas serão geradas. O nome de cada rotina é formado pela concatenação do string "TRATA$" com o nome do evento. Cada rotina possui como parâmetro a mensagem "m" do tipo MSGEA descrito na seção 4.2 que trata sobre a biblioteca de classes.

A figura 7.22 apresenta as duas rotinas geradas para o diagrama de estados da figura 6.7:

```
void TRATA$CHEGA_CLIENTE(MSGEA m)
{
  if (ESTADO == LIVRE)
    {
      inicia_processamento();
      ESTADO = OCUPADO;
    }
  else if (ESTADO == OCUPADO)
        {
          coloca_na_fila();
          ESTADO = OCUPADO;
        }
}

void TRATA$FIM_ATENDIMENTO(MSGEA m)
{
  if (ESTADO == OCUPADO) && (FILA > 0)
    {
      retira_da_fila();
      inicia_processamento();
      ESTADO = OCUPADO;
    }
  else if (ESTADO == OCUPADO) && (FILA == 0)
        {
          termina_processamento();
          ESTADO = LIVRE;
        }
}
```

FIGURA 7.22 - Rotinas geradas relativas a um diagrama de estados.

Observa-se que ambas as rotinas são compostas por uma estrutura de seleção aninhada utilizando o atributo (variável) estado, existente em todas as entidades presentes na simulação. Como estas duas rotinas retratam uma determinada instância da classe gerente, o atributo estado é o mesmo nas duas rotinas. Uma vez que ambas as rotinas possuem a mesma lógica algorítmica, será realizada somente a análise da rotina TRATA$FIM_ATENDIMENTO.

Na construção do código da rotina TRATA$FIM_ATENDIMENTO, as seguintes tarefas são executadas:

- cada ocorrência do evento FIM_ATENDIMENTO dá origem a uma seleção. Como pode ser observado na figura 6.7 existem 2 ocorrências deste evento. Desta forma o código desta rotina será composto por 2 seleções aninhadas. A condição de uma seleção é composta pela igualdade do atributo estado com o nome do estado origem da transição concatenada com um 'and' lógico com a condição da transição, se houver.

- o corpo de cada estrutura de seleção é formado por uma chamada a rotina que representa a ação da transição e uma atribuição do estado destino da transição ao atributo estado.

Estes dois passos são realizados para cada ocorrência do evento que está sendo tratado através de uma pesquisa em todas as transições da lista de adjacências apresentada na figura 7.21.

7.5.3 Cenário

7.5.3.1 Estrutura de dados

A estrutura de dados do cenário é praticamente toda voltada para a parte visual da apresentação do modelo. As principais estruturas de dados utilizadas na geração de código do cenário são duas listas encadeadas: uma contendo todas as instâncias existentes no cenário (figura 7.23) e outra contendo todos os relacionamentos realizados entre as instâncias (figura 7.24).

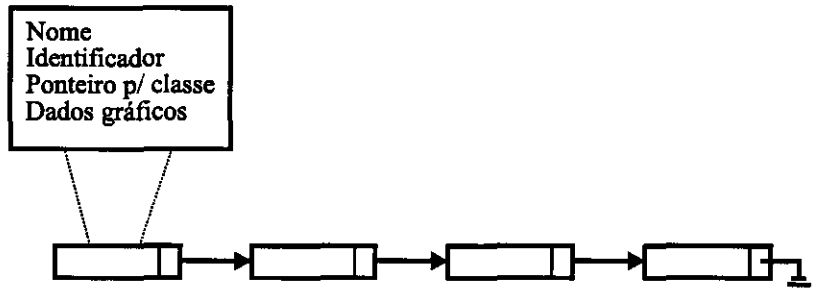


FIGURA 7.23 - Lista de instâncias do cenário.

Cada instância descrita na lista de instâncias contém as seguintes informações: nome, identificador, ponteiro para a classe no diagrama de classes da qual ela é instância, e informações gráficas como, por exemplo, o bitmap associado a ela, sua posição XY no cenário, sua orientação, etc.

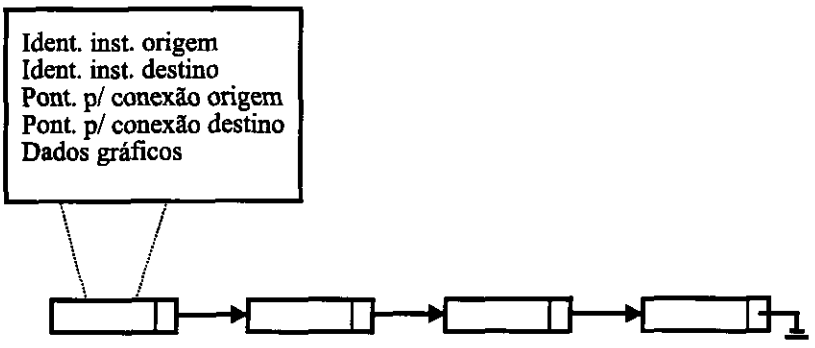


FIGURA 7.24 - Lista de relacionamentos do cenário.

A lista de relacionamentos contém as seguintes informações para cada elemento da lista: os identificadores das instâncias envolvidas no relacionamento, ponteiros para os pontos de conexão definidos em cada uma das classes das instâncias envolvidas e informações gráficas como por exemplo a lista de pontos que formam a trajetória a ser seguida pela representação visual da instância no momento da experimentação.

7.5.3.2 Código gerado

O código gerado pelo cenário está intimamente relacionado com o código gerado pelo diagrama de classes, complementado-o em muitas partes. Em primeiro lugar é realizada a inicialização de cada instância existente na estrutura do cenário da seguinte forma:

- os identificadores de cada instância são definidos no início do código gerado para a classe no diagrama de classes;

- no método START de cada classe são criadas todas as instâncias desta classe.

Também devem ser informados os relacionamentos existentes entre as instâncias através de seus pontos de conexão (de onde partem e chegam as ligações) que são inicializados e codificados através do método RECEBEIDPONTO. Estes relacionamentos já foram analisados quanto a sua consistência, uma vez que somente pode haver relacionamento entre duas instâncias caso haja um relacionamento entre as classes as quais as instâncias pertencem. É a partir do cenário que também são retirados os atributos das instâncias que são inicializados nas classes.

De uma forma geral, pode-se afirmar que é a partir das instâncias descritas no cenário que as classes realmente ganham corpo. É nas instâncias que se encontram os atributos da classe, a possibilidade ou não de troca de mensagens entre uma classe e outra, as constantes para a simulação, a inicialização dos identificadores dos pontos de conexão e dos relacionamentos, etc.

8 Comparação com outros ambientes de simulação

Este capítulo apresenta um conjunto de ambientes de simulação que possuem características de VIS e de VIM. O objetivo é realizar uma análise comparativa entre estes ambientes descritos e o ambiente *VISME* que foi proposto. As seções 8.1 a 8.6 apresentam resumidamente a descrição de cada ambiente. A seção 8.7 apresenta uma tabela comparativa juntamente com algumas considerações a respeito de cada item analisado.

8.1 VMSS

O ambiente VMSS (Visual Model-building and Simulation System) [KAM96] é um sistema de simulação para sistemas flexíveis de manufatura e está focalizado no controle operacional de veículos de controle automático (AGV - Automated Guided Vehicle). Este ambiente integra os conceitos de modelagem interativa visual (VIM) e simulação interativa visual (VIS).

VMSS é composto por três módulos principais: Escritor (Writer), Teatro (Theater) e Diretor (Director), e por um banco de dados (Cenário) que é constituído pelas propriedades de cada equipamento, o caminho do AGV, etc. O módulo Escritor é o módulo de VIM que permite a criação do Cenário. Este módulo é constituído por um editor gráfico, onde através de operações de seleção e posicionamento das representações gráficas das entidades, pode-se construir o cenário. As entidades em VMSS são chamadas de equipamentos cujos ícones associados ficam à disposição em uma palheta para que o usuário faça a seleção. Cada equipamento é descrito por uma classe que possui parâmetros de instanciamento que devem ser fornecidos pelo usuário quando de seu posicionamento no cenário.

O módulo Teatro é o módulo de experimentação baseado nas carcterísticas de VIS. Este módulo é constituído de várias janelas contendo diversas informações a respeito do modelo e uma janela principal onde é apresentada a animação baseada no Cenário descrito através do módulo Escritor. A animação é simples de ser realizada uma vez que o conjunto de elementos que pode ser instanciado é fixo e são representados por ícones cujo comportamento é conhecido. Os resultados da simulação são incorporados ao Cenário, permitindo desta forma continuar uma simulação a partir de um cenário previamente salvo.

O módulo Diretor é um módulo de controle que permite o funcionamento integrado dos módulos Escritor e Teatro. Através de uma interface amigável baseada em cardápios, o usuário pode especificar um comando de controle para a simulação pela seleção de uma lista de comandos (eventos, equipamentos, AGVs e partes). Existe um assistente inteligente que permite que os comandos especificados incompletamente sejam traduzidos para comandos completos e corretos. O comando de controle especificado é transferido para o módulo que controla o Cenário.

8.2 VSE

O ambiente VSE (Visual Simulation Environment) [BAL97] é um ambiente de simulação de propósitos gerais orientado a objetos que permite o desenvolvimento e execução de modelos visuais hierárquicos. O ambiente VSE possui uma biblioteca de classes que implementa as classes das quais todos os componentes do modelo são derivados. Este ambiente é composto por três ferramentas principais: Editor, Simulador e Analisador.

O Editor é uma ferramenta de VIM utilizada para criar o modelo visual e automaticamente traduzí-lo para um código executável. O modelo é especificado através de dez partes estruturais pré-definidas onde, por meio de uma série de janelas o usuário especifica as classes de objetos, hierarquias de herança e de agregação que compõem o modelo. Os modelos em VSE podem conter componentes estáticos que permanecem parados durante toda a simulação e, componentes dinâmicos que podem movimentar-se dentro do modelo. Cada componente pode ser detalhado em subcomponentes e, os componentes que não são folhas da hierarquia podem ter uma representação visual associada.

O Simulador é um ambiente separado utilizado para executar, animar e experimentar os modelos de simulação criados através da ferramenta Editor. O Simulador oferece vários dos recursos esperados por um ambiente de VIS. O usuário pode alterar o conteúdo de atributos das instâncias e realizar um experimento através de diferentes condições. Durante o experimento o usuário pode abrir e fechar janelas de forma a poder acompanhar a animação dos aspectos do modelo que ele julgar mais importantes.

O Analisador é o responsável pela análise estatística dos resultados da experimentação. São oferecidos recursos para o cálculo de médias, intervalos de confiança, valores mínimos e máximos, etc. Ao final da experimentação os resultados são armazenados em arquivos para uma posterior análise. Também são oferecidos recursos para o tratamento dos dados de entrada, determinação de distribuições de probabilidade entre outros.

8.3 ARENA

O ambiente de simulação ARENA [MAR96] suporta todos os passos básicos de um processo de simulação. ARENA apresenta um ambiente integrado para análise dos dados de entrada, construção do modelo de simulação, experimentação interativa, animação e análise de resultados. É um sistema gráfico de modelagem e animação baseado nos conceitos de orientação a objetos e de modelagem hierárquica, sendo construído utilizando-se da plataforma SIMAN/Cinema.

ARENA é um ambiente baseado em janelas, onde cada etapa do processo de simulação é desenvolvida em uma janela distinta. Um modelo em ARENA é construído através do posicionamento e conexão de módulos na área de trabalho da janela de modelos. Estes módulos são selecionados a partir de um "template" .

Quando um módulo é selecionado e posicionado (uma instância dele é criada), o usuário deve especificar seu comportamento e os valores de seus parâmetros e atributos. A grande força de ARENA está no seu suporte a utilização e construção de "templates" para diversas áreas de aplicação. O "template" padrão de ARENA contém módulos que permitem uma modelagem de mais alto nível que os módulos padrões de SIMAN. Dois tipos de módulos estão disponíveis: os módulos lógicos que são usados para descrever o fluxo lógico do sistema e, os módulos de dados que descrevem as características dos componentes do sistema.

Na fase de experimentação ARENA apresenta uma animação do modelo e permite que o usuário faça alterações nos parâmetros do modelo e visualize os dados através de gráficos e estatísticas. Quanto a parte de análise de dados, ARENA possui análise dos dados de entrada que permite a escolha da melhor distribuição de probabilidade que se ajusta ao conjunto de dados e, análise de resultados em pós-processamento através de diversos gráficos e de estatísticas.

8.4 MICRO SAINT

Micro Saint [DRU94] é uma ferramenta para modelagem e simulação de sistemas de diferentes tipos, tais como: manufatura, reengenharia de processos de negócio e serviços industriais. Esta ferramenta utiliza a metodologia de modelo de redes de tarefas, onde as atividades são representadas em um diagrama como nodos e as setas entre os nodos representam a seqüência na qual as atividades são executadas. Este método permite desenvolver modelos usando a mesma técnica utilizada para definir um diagrama de fluxo de atividade. Cada atividade, seja ela uma atividade humana ou uma atividade de sistema é definida usando o mesmo método. Isto reduz a complexidade da interface do usuário e elimina a necessidade de rotinas de programação específicas para uma aplicação.

Um modelo em Micro Saint é composto por redes que podem ser seqüências de tarefas para serem realizadas por uma pessoa, uma máquina em um sistema de manufatura ou, uma série de processos que definem uma organização. As redes são compostas por redes de níveis mais baixos e por tarefas que representam o nível mais baixo no modelo. O processo de construção de um modelo em Micro Saint envolve dois passos distintos, porém inter-relacionados. Em primeiro lugar, deve ser definida a estrutura da rede de tarefas. Isto é feito através da seleção dos objetos (tarefa, caminho, fila, etc) na barra de ferramentas e seu posicionamento na área de trabalho. Num segundo momento, após a definição da rede de tarefas, deve-se descrever cada atividade que compõe esta rede. Selecionando-se a tarefa, é ativada uma janela onde são configuradas as informações específicas para aquela atividade, tais como: tempo médio que a tarefa leva para ser executada, condições lógicas para controlar a execução das tarefas, comandos a serem executados sempre que ocorre uma mudança no estado do sistema.

Para melhor representar as situações do sistema real, existe um mecanismo que permite determinar a seqüência de execução das tarefas. Desta forma, quando existe mais de uma tarefa a ser executada na seqüência de tarefas, as seguintes

decisões podem ser tomadas: probabilística (apenas uma tarefa é executada de acordo com as probabilidades relativas a cada tarefa), múltipla (todas as tarefas que tiverem suas condições satisfeitas são executadas) e tática (entre as tarefas com condições satisfeitas é executada a que possui o valor mais alto). Também podem ser associadas filas às atividades, facilitando desta forma o monitoramento de variáveis de desempenho associado a estas filas.

Na fase de experimentação algumas informações podem ser obtidas pela visualização da execução do modelo. A capacidade de animação simbólica do Micro Saint fornece uma visão animada do diagrama de rede, podendo ser observadas as tarefas em execução no momento, o fluxo das entidades através da rede e as filas de espera. Além desta forma de animação, Micro Saint também fornece uma animação icônica que permite que seja construída uma imagem mais realística do sistema. Durante a execução do modelo é possível coletar dados (variáveis do modelo) para análise. Estes dados podem ser visualizados em uma janela que atualiza dinamicamente seus valores durante a simulação. Os resultados da simulação podem ser analisados através de gráficos (gráfico de linhas, barras, etc) e estatísticas (médias, desvio padrão, máximos e mínimos) existentes no Micro Saint.

8.5 WITNESS

WITNESS [THO94] é um sistema de simulação interativo visual destinado à modelagem e simulação de sistemas de manufatura. WITNESS é basicamente um sistema de simulação com uma interface gráfica baseada em cardápios que permite construir um modelo hierárquico através de representações gráficas para as entidades. O processo de modelagem em WITNESS pode ser dividido em 3 fases: definição, exibição e detalhamento.

A fase de definição do modelo é destinada à criação e denominação dos elementos de modelagem necessários ao modelo. WITNESS contém 22 elementos pré-definidos que podem ser agrupados em 3 categorias: a) elementos físicos, como 'parts', 'machines', 'tracks', 'vehicles', 'conveyors', 'buffers'; b) elementos lógicos, como 'variables', 'attributes', 'functions' e 'files'; c) elementos de informação, como 'histograms' e 'timeseries'. A seguir é realizada a exibição dos elementos, ou seja, o posicionamento da representação gráfica dos elementos definidos na fase anterior em uma das janelas existentes no modelo. Com isto, o usuário é capaz de criar o cenário do modelo de simulação. Já na fase de detalhamento são especificados os parâmetros necessários para a execução do modelo. Para cada tipo de elemento é apresentada uma janela com as suas propriedades que devem ser especificadas. Também nesta fase são especificadas como as entidades se movimentam no cenário durante a simulação. Para isto, devem ser definidas as regras de movimentação de cada entidade. WITNESS possui 8 regras pré-definidas, mas permite que sejam definidas outras regras mais complexas através de estruturas lógicos.

O usuário não possui recursos para interagir com o modelo durante fase de experimentação em WITNESS. Para experimentar um modelo, deve-se criar um experimento, ou seja, definir todos os parâmetros necessários (qual o cenário, tempo

de simulação, período de 'warm-up', entidades a serem analisadas, qual o arquivo de resultados, etc). Após o final da simulação pode-se determinar estatisticamente a performance do modelo.

8.6 TAYLOR II

Taylor II [KIN96] é um sistema de simulação para sistemas discretos onde haja formação de filas. As maiores áreas de aplicação de Taylor são manufatura e logística, mas pode-se simular qualquer tipo de sistema onde as entidades discretas possam ser processadas, transportadas e armazenadas. Taylor contém um conjunto de funções (ferramentas) que permitem o controle de todo o processo de simulação.

A primeira atividade em Taylor é a construção do modelo, sendo esta toda dirigida por cardápios. Um modelo consiste de 4 entidades fundamentais: "elements", "jobs", "routings"e "products". Existem 3 operações básicas chamadas de tarefas: processamento, transporte e armazenagem. As entidades utilizadas nas tarefas são chamadas de produtos. O primeiro passo na construção do modelo é a definição do cenário. Este cenário consiste no posicionamento dos diferentes tipos de elementos na área de trabalho, cada um representando uma função no sistema real. Os seguintes elementos estão disponíveis: "buffer", "machine", "tranport", "conveyor", "path", "AGV", "warehouse", "reservoir", "aid" e "inout". Os produtos ocupam um elemento e são enviados para outro elemento quando sua operação é encerrada. Desta forma, é necessária a descrição de como os produtos fluem através do modelo. Esta descrição é armazenada nas "routings" sendo feita através da seleção dos elementos um após o outro, formando desta forma o caminho que o produto segue no modelo. O próximo passo na construção do modelo é o seu detalhamento, ou seja, a definição dos parâmetros que descrevem o comportamento do modelo. Os elementos e as tarefas possuem um conjunto próprio de parâmetros que os definem.

A fase de experimentação permite que o usuário pare a simulação, faça modificações e continue a simulação. Um modelo pode ser simulado por um certo período de tempo ou até que uma condição seja alcançada. Taylor utiliza uma macro linguagem, chamada TLI (Taylor Language Interface), que permite modificar o comportamento do modelo através de variáveis pré-definidas específicas de simulação e de variáveis definidas pelo usuário. Esta linguagem é utilizada para elementos, tarefas e caminhos. Durante a simulação pode-se observar os dados através de gráficos de diversos tipos e estatísticas que são atualizados dinamicamente. Uma das principais características de Taylor é a sua capacidade de animação. O usuário pode visualizar o modelo através de uma animação 2D ou 3D, realizar rotações, mudar o ângulo de visão, realizar zoom, etc. Na parte de análise dos dados, Taylor apresenta tanto análise dos dados de entrada, onde pode sugerir a melhor distribuição de probabilidade para o conjunto de dados, como análise de resultados de saída através de gráficos e estatísticas.

8.7 PROMODEL

ProModel [BAI94] é uma ferramenta de simulação que permite modelar sistemas de manufatura, sendo baseada em uma interface gráfica e construções de modelagem orientadas a objetos que eliminam a necessidade de programação. Mesmo assim, para uma maior flexibilidade do usuário, ProModel permite que sejam vinculadas dinamicamente subrotinas escritas em uma linguagem de programação durante a simulação.

Um modelo desenvolvido em ProModel é totalmente gráfico e orientado a objetos. Através de uma interface baseada em cardápios o usuário define os elementos de modelagem que representam o sistema modelado. A construção do modelo pode ser dividida em dois passos distintos: descrição do cenário através do posicionamento dos elementos físicos (entidades, máquinas e recursos) e a definição da seqüência de processamento e fluxo lógico das entidades entre os elementos físicos descritos. Esta definição lógica pode ser realizada através de distribuições, funções, subrotinas, expressões lógicas, etc. A animação está integrada com a definição do modelo, ou seja, o cenário da animação é aquele que está sendo montado na construção do modelo.

Na fase de experimentação o usuário pode modificar parâmetros do modelo (capacidades, tempos de operação, etc) sem a necessidade de realizar um novo experimento. O modelo pode ser simulado por um tempo específico ou até que todas as entidades sejam processadas. Durante a simulação pode-se consultar o estado de cada recurso ou o valor de qualquer elemento lógico. A parte de análise de resultados permite a visualização de estatísticas e de diversos tipos de gráficos (histogramas, torta, linhas, etc).

8.8 SIMOBJECT

SIMOBJECT [GOB94] é um ambiente gráfico orientado a objetos para construção de modelos hierárquicos. SIMOBJECT foi baseado em MODSIM II [BEL90] e sua principal área de aplicação é redes de comunicação. SIMOBJECT contém uma coleção de objetos pré-definidos permitindo tanto a construção de modelos de sistemas que envolvam fluxo através de redes como a construção de interfaces gráficas para construir e configurar estes modelos. Em conjunto com SIMOBJECT existe o ambiente OBJECT.MGR que permite a criação de novos objetos e modificação dos objetos já existentes.

O modelo de simulação em SIMOBJECT é construído através da seleção dos objetos em uma palheta de objetos e seu posicionamento na área de trabalho. Estes objetos são conectados para representar os caminhos e a seqüência lógica dos elementos. Os parâmetros de cada objeto são configurados através de uma janela de parâmetros existente para cada um. SIMOBJECT possui uma coleção básica pré-definida de objetos, que são: nodos (modelam processos, recursos e hierarquia), tokens (modelam os elementos que se movimentam através do sistema) e arcos (formam o caminho através do qual os tokens fluem).

Durante a experimentação SIMOBJECT apresenta uma animação do modelo de simulação. O usuário pode interromper a simulação, realizar modificações nos parâmetros e imediatamente observar o efeito destas mudanças continuando a simulação. Também pode-se obter dados estatísticos dos objetos durante a simulação. Uma vez que o modelo é hierárquico, um subsistema completo pode ser visualizado como um simples ícone, facilitando desta forma a visualização da animação. É possível a mudança de um subsistema por outro com diferente comportamento. Para a análise de resultados, diversos tipos de gráficos são oferecidos.

8.9 Análise comparativa

A análise de diferentes características para cada um dos ambientes estudados permitiu que fosse construída a tabela 8.1 apresentada abaixo. Os ítens analisados referem-se aos recursos oferecidos nos aspectos de modelagem, experimentação e visualização.

VISME é o único ambiente, dos ambientes analisados, que utiliza representação gráfica (diagrama de estados) para a descrição do comportamento das entidades. Todos os demais ambientes e sistemas utilizam a mesma abordagem na descrição do comportamento: configuração de parâmetros e atividades a serem executadas em janelas específicas para cada tipo de entidade.

Todos os ambientes analisados oferecem a possibilidade de construir o modelo através de um editor gráfico específico. Em todos, o usuário descreve a estrutura do modelo como uma rede de ícones que representam as entidades do modelo. Normalmente estes ambientes apresentam "templates" de entidades onde cada uma contém um ícone associado que o usuário seleciona em uma palheta de ícones. Desta forma, a estrutura gráfica do modelo é o próprio cenário (diagrama de instâncias), utilizado muitas vezes para a animação. O ambiente *VISME* apresenta uma abordagem diferente, separando a estrutura estática do modelo de seu diagrama de instâncias. O usuário possui um editor gráfico (diagrama de classes) que permite descrever a estrutura estática do modelo, facilitando a compreensão do mesmo na medida em que pode-se visualizar as entidades (classes) presentes no modelo e os diferentes tipos de relacionamentos que envolvem as entidades (inclusive os de herança). Já o diagrama de instâncias (cenário) do ambiente *VISME* utiliza a mesma filosofia de construção que os demais ambientes.

TABELA 8.1 - Análise comparativa.

	VISME	VMSS	VSE	ARENA	MICRO SAINT	PROMODEL	WITNESS	SIMOBJECT	TAYLOR II
Projeto hierárquico	x	-	x	x	-	-	x	x	-
Representação gráfica do comportamento das entidades	x	-	-	-	-	-	-	-	-
Representação gráfica da estrutura estática do modelo	x	-	-	-	-	-	-	-	-
Representação gráfica do cenário do modelo	x	x	x	x	x	x	x	x	x
Recursos de interação entre o usuário e o modelo	x	x	x	x	x	x	x	x	x
Recursos de controle sem interromper o experimento	x	-	-	-	-	-	-	-	-
Recursos de monitoramento visualização de resultados	x	x	x	x	x	x	x	x	x
Recursos de animação	x	x	x	x	x	x	x	x	x
Recursos de pré-processamento	-	-	x	x	-	-	-	-	x
Recursos de análise em pós-processamento	x	x	x	x	x	x	x	x	x
Recursos para coleta de estatísticas	x	x	x	x	x	x	x	x	x
Criação de bibliotecas de entidades	x	-	x	x	-	x	x	x	x

Quanto aos recursos de interação entre o usuário e o modelo, a totalidade dos ambientes analisados oferece recursos para conduzir o experimento (tais como limite de tempo ou até que uma condição seja alcançada) e recursos que permitem alterar parâmetros/atributos das entidades e variáveis de controle da simulação. Igualmente, todos os ambientes apresentam algum tipo de recurso que permite o monitoramento e visualização do andamento da simulação e de resultados parciais, seja através de gráficos ou através de ícones associados a entidades ou seus atributos.

Uma característica de *VISME*, não encontrada nos demais ambientes analisados, são os recursos de controle oferecidos durante a experimentação sem a necessidade de interromper o experimento. Todos os ambientes permitem que sejam alterados parâmetros e variáveis durante o experimento, mas estas alterações não

modificam o modelo em si. Já o ambiente *VISME*, além destes recursos, permite que seja alterada a estrutura estática e dinâmica do modelo através de operações como: inclusão/remoção de classes e instâncias de classes, modificação de relacionamentos e do comportamento das entidades.

Em relação aos recursos de coleta de estatísticas, todos os ambientes analisados oferecem recursos básicos para geração de distribuições de probabilidade, cálculos de médias e desvio padrão. Também quanto a análise em pós-processamento todos os ambientes apresentam gráficos de diversos tipos e estatísticas. Já a análise em pré-processamento, assim como adotado por ARENA, VSE e Taylor, é um recurso não disponível em *VISME* que deve ser implementado em uma futura extensão.

Os recursos de animação apresentados em *VISME* estão limitados à movimentação das representações gráficas das entidades no cenário construído. Diversos ambientes possuem, além desta forma de animação, recursos que permitem a representação do modelo de uma forma mais realística. Pode-se visualizar o modelo através de animações 2D ou 3D, realizar rotações, realizar zoom, etc. Como todos os ambientes e sistemas analisados possuem recursos gráficos para construção e experimentação do modelo, a grande maioria destes incentivam a criação de bibliotecas de entidades.

A partir desta análise, pode-se concluir que o ambiente *VISME* se destaca em relação aos demais ambientes pelos seguintes aspectos:

- pela forma gráfica de representar o comportamento das entidades;
- pela forma de representar o modelo separando a estrutura estática de seu diagrama de instâncias;
- pelo conjunto de recursos de controle que permitem modificar o modelo sem a necessidade de interromper a experimentação.

9 Conclusões

9.1 Resumo e contribuição do trabalho

A principal contribuição deste trabalho foi a identificação e a exploração das potencialidades e recursos advindos da integração de VIM e VIS em um único ambiente. Desta forma, este trabalho propôs uma nova abordagem, chamada de VISM (Visual Interactive Simulation and Modeling), que permite num único ambiente a interação do usuário tanto com o modelo como com o experimento em tempo de simulação/experimentação. Esta abordagem de disponibilizar simultaneamente as facilidades de VIM e VIS durante a experimentação permite a interação do usuário diretamente sobre a representação visual do modelo.

A idéia de um único ambiente é que o usuário pode durante a experimentação alterar o modelo que está sendo simulado conforme a análise por ele realizada. Desta forma, o usuário não necessita encerrar o experimento que está realizando. Com isto ele interrompe (suspende) a experimentação, altera o modelo segundo suas observações e reinicia a experimentação do ponto em que foi interrompida. Uma alteração do modelo vai desde a modificação de parâmetros/atributos de uma instância até a modificação da estrutura e comportamento do modelo (com a mesma ferramenta utilizada para a construção do modelo de simulação).

Também deve ser salientado que esta abordagem tem por objetivo a integração total das características e recursos oferecidos por VIS e VIM, de forma que o usuário tenha um controle completo sobre todo o processo de simulação. Sua contribuição principal não são novas ferramentas criadas para modelagem e experimentação, mas a solução das questões que surgem com a disponibilidade simultânea de facilidades de VIS e VIM durante a experimentação do modelo.

Para que se chegasse a esta nova abordagem, foi realizado estudo daqueles ambientes de simulação que contemplassem as fases de modelagem e experimentação através de recursos visuais e interativos. Com base neste estudo, pode-se definir de forma bem clara, conceitos como VIS e VIM que são utilizados com freqüência na área de simulação, mas que por muitas vezes ficam confusos devido à diversidade de entendimento e conceitos.

A partir da análise de diversos ambientes de simulação, procurou-se enquadrá-los dentro da classificação proposta por Marshall [MAR90]. Através dos recursos e facilidades disponíveis nos ambientes de simulação analisados, pode-se concluir quais seriam as características desejáveis para ambientes de VIS. Da mesma forma, foram analisados os recursos destinados à modelagem em diversos ambientes permitindo que fossem identificados os conceitos e características para ambientes de VIM. Nestes ambientes procurou-se salientar a construção do modelo de simulação através de recursos gráficos-interativos.

Para a experimentação e conseqüente validação desta nova abordagem foi construído o ambiente *VISME* (Visual Interactive Simulation and Modeling Environment). *VISME* é um ambiente de simulação discreta que possui uma interface única que fornece todos os recursos necessários de modelagem e experimentação. Ele foi projetado e construído para que tenha um alto grau de interação entre o usuário e o modelo durante todo o processo de simulação, desde a fase de construção do modelo até a fase de análise de resultados. O ambiente *VISME* se caracteriza pela natureza gráfica e interativa com que seus recursos são oferecidos ao usuário.

A filosofia do ambiente *VISME*, observando suas características e recursos oferecidos, permite que se faça uma diferenciação entre dois tipos de usuários: o modelador e o usuário final. O modelador, ou seja, o responsável pela construção do modelo de simulação deve ser um especialista em informática, pois deve conhecer programação, o mecanismo de orientação a objetos e o funcionamento da biblioteca de classes para simulação [COP97]. A forma de desenvolvimento de modelos através do ambiente *VISME*, permite ao modelador a construção de modelos claros e estruturados. Isto significa que este modelo pode ser facilmente compreendido por outros modeladores e até mesmo por usuários que não são especialistas em informática.

Já o usuário final não necessita ser um especialista em informática, uma vez que sua atividade principal é a de configurar o modelo através dos recursos oferecidos no cenário conforme o sistema que ele deve analisar. Com isto, quando o usuário for operar no ambiente ele já tem a sua disposição uma biblioteca de objetos relacionada com o sistema que está modelando e o comportamento e atributos de cada objeto. Sua tarefa é, a partir da especificação do sistema, construir a apresentação visual do mesmo através dos recursos oferecidos pelo cenário. Isto não significa, entretanto, que caso o usuário final seja mais experiente, ele não possa modificar o modelo, realizando tarefas como a criação de novas classes e a modificação do comportamento de uma classe.

Esta abordagem se caracteriza pelos recursos de natureza gráfica e interativa durante todo o processo de simulação, desde a construção do modelo até a análise de resultados. Devido à disponibilidade de recursos desta natureza, é que durante a fase de experimentação, os efeitos de pequenas alterações realizadas no modelo podem ser observados com maior clareza por parte do usuário. Deste modo, é extremamente útil que o usuário tenha recursos interativos mais diretos para conduzir adequadamente seu experimento, ou seja, sua participação em termos de operar com o ambiente deve ser a menor possível.

Uma vantagem desta abordagem está voltada para o ensino da simulação. Com um ambiente visual-interativo pode-se obter um entendimento maior do sistema modelado através da observação das diversas modificações realizadas. Claro que algumas modificações para determinadas aplicações podem não fazer sentido nenhum e ocasionariam erros irremediáveis, mas isto também faz parte do aprendizado.

9.2 Perspectivas e continuidade do trabalho

Apesar do protótipo do ambiente *VISME* já se encontrar funcionando, ele somente foi usado e testado de forma mais concreta para a aplicação exemplo apresentada no trabalho. Desta forma, é conveniente que a partir deste momento o ambiente seja utilizado para outros projetos/aplicações para que muitos aspectos dele possam ser aprimorados. Dentro deste contexto, já existem dois trabalhos de mestrado em andamento que utilizarão o ambiente *VISME* para a construção de seus modelos.

A definição da estrutura estática do modelo, realizada através do diagrama de classes, possibilita uma descrição hierárquica do modelo, permitindo seu detalhamento em diferentes níveis de acordo com a necessidade. Com isto, para cada nível do diagrama de classes, um cenário poderia ser especificado, o que levaria a um cenário hierárquico. Cada cenário seria visualizado em uma janela de visualização distinta, permitindo uma animação simultânea durante a experimentação. A aplicação exemplo utilizada no trabalho permitiu que o modelo fosse descrito em apenas dois níveis, considerando-se o nível mais alto da hierarquia. Desta forma, não houve a necessidade de construção de um cenário hierárquico, mas tal recurso deve ser acrescentado ao ambiente para permitir um modelo mais realístico.

As ferramentas de modelagem do ambiente *VISME* foram desenvolvidas de forma modular, evitando ao máximo a dependência entre elas. Devido a isto, uma extensão do ambiente seria o desenvolvimento de outras formas diagramáticas para descrever o comportamento das classes. O usuário durante o desenvolvimento do modelo poderia configurar para cada classe a forma de descrição que julgar mais conveniente. A geração de código não seria afetada, uma vez que o código gerado pelo diagrama de estados é totalmente independente das demais ferramentas. Desta forma, deve-se desenvolver um novo módulo de geração de código para esta nova forma de descrição. Os códigos gerados para uma mesma classe, mas descrita de forma diferente devem ser equivalentes.

Quanto à análise estatística, o ambiente *VISME* apresenta somente recursos para uma análise em pós-processamento, não possuindo recursos para análise dos dados de entrada, determinação de distribuições de probabilidade, etc. Tais recursos devem ser acrescentados aos já existentes.

Finalmente, para fornecer uma maior garantia dos dados e consistência dos mesmos, seria interessante a utilização de um banco de dados para armazenamento dos modelos. Atualmente o ambiente *VISME* utiliza um sistema de gerência de arquivos para armazenar seus modelos e experimentos.

Bibliografia

[BAI94] BAIRD, S. P.; LEAVY, J. J. Simulation modeling using Promodel for Windows. In: WINTER SIMULATION CONFERENCE, 1994, Lake Buena Vista, Florida. **Proceedings...** Lake Buena Vista: ASA/IEEE/ACM/SCS, 1994. p.527-532.

[BAL90] BALCI, Osman. Guidelines for successful simulation studies. In: WINTER SIMULATION CONFERENCE, 1990, New Orleans, Louisiana. **Proceedings...** New Orleans: IEEE/ACM/SCS, 1990. p.25-32.

[BAL92] BALCI, O.; NANCE, R.E. The simulation model development environment: an overview. In: WINTER SIMULATION CONFERENCE, 1992, Arlington, Virginia. **Proceedings...** Arlington: IEEE/ACM/SCS, 1992. p.726-736.

[BAL97] BALCI, O. et al. The visual simulation environment. In: EUROPEAN SIMULATION MULTICONFERENCE, 11., 1997, Istanbul, Turkey. **Proceedings...** [S.l. : s.n.], 1997, p.61-68.

[BAN94] BANKS, J. Software for simulation. In: WINTER SIMULATION CONFERENCE, 1994, Lake Buena Vista, Florida. **Proceedings...** Lake Buena Vista: ASA/IEEE/ACM/SCS, 1994. p.26-33.

[BEL87] BELL, P.C.; O'KEEFE, R. M. Visual Interactive Simulation - history, recent developments, and major issues. **Simulation**, San Diego, v.49, n.3, p.109-116, Sept. 1987.

[BEL90] BELANGER, Ron. MODSIM II - A modular, object-oriented language. In: WINTER SIMULATION CONFERENCE, 1990, New Orleans, Louisiana. **Proceedings...** New Orleans: IEEE/ACM/SCS, 1990. p.118-122.

[BIR73] BIRTWISTLE, G. M. **Simula Begin**. New York: Lund Student Litterature, 1973.

[BIS91] BISCHAK, D. P.; ROBERTS, S. D. Object oriented simulation. In: WINTER SIMULATION CONFERENCE, 1991, Phoenix, Arizona. **Proceedings...** Phoenix: IEEE/ACM/SCS, 1991. p.194-203.

[BIS90] BISHOP, J. L.; BALCI, O. General purpose visual simulation system: a functional description. In: WINTER SIMULATION CONFERENCE, 1990, New Orleans, Louisiana. **Proceedings...** New Orleans: IEEE/ACM/SCS, 1990. p.504-512.

[BOO94] BOOCH, G. **Object-oriented analysis and design**. New York: The Benjamin/Cummings Publish Company, 1994.

[BOR90] BORGES, J. A.; JOHNSON, R. E. Multiparadigm visual programming language. In: WORKSHOP ON VISUAL LANGUAGES, 1990, Skokie, Illinois. **Proceedings...** Los Alamitos: IEEE, 1990. p.233-240.

[BRO92] BRODLIE, K. W. et al. **Scientific visualization - techniques and applications**. Berlin: Springer-Verlag, 1992.

[CAC91] CACI PRODUCTS COMPANY. **SIMGRAPHICS: User's guide and casebook**. La Jolla: CACI, 1991.

[CER94] CERIC, V. Hierarchical abilities of diagrammatic representations of discrete event simulation models. In: WINTER SIMULATION CONFERENCE, 1994, Lake Buena Vista, Florida. **Proceedings...** Lake Buena Vista: ASA/IEEE/ACM/SCS, 1994. p.589-594.

[CHA87] CHANG, S. K. Visual languages: a tutorial and survey. **IEEE Software**, New York, v.4, n.1, p.29-39, Jan. 1987.

[COP96] COPSTEIN, B.; PEREIRA, C. E.; WAGNER, F. R. The object oriented approach and the event simulation paradigms. In: EUROPEAN SIMULATION MULTICONFERENCE, 10., 1996, Budapest, Hungary. **Proceedings...** [S.l. : s.n.], 1996, p.57-61.

[COP97] COPSTEIN, B. **SIMOO: Plataforma orientada a objetos para simulação discreta multi-paradigma**. Porto Alegre: CPGCC da UFRGS, 1997. 137p. Tese de doutorado.

[COX87] COX, S. Interactive graphics in GPSS/PC. **Simulation**, San Diego, v.49, n.3, p.117-122, Sept. 1987.

[DIJ96] Van DIJK, J. N. et al. Visual interactive modelling with SimView for organizational improvement. **Simulation**, San Diego, v.67, n.2, p.106-120, Aug. 1996.

[DRU94] DRURY, C. E.; LAUGHERY, K. R. Fundamentals of simulation using Micro Saint. In: WINTER SIMULATION CONFERENCE, 1994, Lake Buena Vista, Florida. **Proceedings...** Lake Buena Vista: ASA/IEEE/ACM/SCS, 1994. p.546-551.

[DYE90] DYER, D. S. A dataflow toolkit for visualization. **IEEE Computer Graphics and Applications**, Los Alamitos, v.10, n.4, p.60-69, July 1990.

[EAR90] EARLE, N. J.; BRUNNER, D. T.; HENRIKSEN, J. O. Proof: The general purpose animator. In: WINTER SIMULATION CONFERENCE, 1990, New Orleans, Louisiana. **Proceedings...** New Orleans: IEEE/ACM/SCS, 1990. p.106-108.

[EAR92] EARNSHAW, R. A.; WISEMAN, N. **An introductory guide to scientific visualization**. Berlin: Springer-Verlag, 1992.

[ELD91] ELDER, M. D. Visual Interactive Modelling. MUNFORD, A. G.; BAILEY, T. C. (Eds.). **Operational Research Tutorial Papers**. [S.l. : s.n.], 1991. p.1-16.

[ELL88] ELLSON, R.; COX, D. Visualization of injection molding. **Simulation**, San Diego, v.51, n.5, p.184-188, Nov. 1988.

[ELZ86] ELZAS, M. S. The applicability of artificial intelligence techniques to knowledge representation in modelling and simulation. In: MODELLING AND SIMULATION METHODOLOGY IN THE ARTIFICIAL INTELLIGENCE ERA. 1986, Amsterdam, The Netherlands. **Proceedings...** Amsterdam: [s.n.], 1986. p.19-40.

[EMS70] EMSHOFF, J. R.; SISSON, R. L. **Design and use of computer simulation models**. London: MacMillan, 1970.

[FID81] FIDDY, E.; BRIGHT, J. G.; HURRION, R. D. SEE-WHY: Interactive simulation on the screen. **Proceedings of the Institute of Mechanical Engineers**, [S.l.], v.293, n.81, p.167-172, 1981.

[FRE91] FREITAS, C. M. D. S. **Um estudo sobre a aplicação dos conceitos de orientação a objetos e de recursos interativos visuais a linguagens e sistemas de simulação**. Porto Alegre: CPGCC da UFRGS, 1992. 55p.

[FRE92] FREITAS, C. M. D. S. **Um estudo sistemático sobre linguagens visuais**. Porto Alegre: CPGCC da UFRGS, 1992. 86p. (RP-197).

[FRE93] FREITAS, C. M. D. S.; WAGNER, F.R. A methodology for selecting visual representations in scientific and simulation applications. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 6., 1993, Recife, PE. **Anais...** Recife: SBC/UFPE, 1993. p89-97.

[FRE94a] FREITAS, C. M. D. S.; WAGNER, F. R. Análise exploratória visual orientada a ferramentas. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 7., 1994, Curitiba, PR. **Anais...** Curitiba: SBC/UFPR, 1994. p197-203.

[FRE94b] FREITAS, C. M. D. S. **Uma abordagem unificada para a análise exploratória e simulação interativa visual**. Porto Alegre: CPGCC da UFRGS, 1994. 146p. Tese de doutorado.

[GOB94] GOBLE, J. G. Simobject: from rapid prototype to finished model – a breakthrough in graphical model building. In: WINTER SIMULATION CONFERENCE, 1994, Lake Buena Vista, Florida. **Proceedings...** Lake Buena Vista: ASA/IEEE/ACM/SCS, 1994. p.437-442.

[GOR69] GORDON, M. G. **System simulation**. Englewood Cliffs, N. J.: Prentice Hall, 1969.

[GOR90] GORDON, R. F. et al. Hierarchical modeling in a graphical simulation system. In: WINTER SIMULATION CONFERENCE, 1990, New Orleans, Louisiana. **Proceedings...** New Orleans: IEEE/ACM/SCS, 1990. p.499-503.

[GOR91] GORDON, R. F. et al. Error detection and display for graphical modeling environments. In: WINTER SIMULATION CONFERENCE, 1991, Phoenix, Arizona. **Proceedings...** Phoenix: IEEE/ACM/SCS, 1991. p.1139-1145.

[HAR97] HAREL, D.; GERY, E. Executable object modeling with statecharts. **Computer**, New York, p.31-42, July 1997.

[HIL96] HILL, D. R. C. **Object-oriented analysis and simulation**. England: Addison-Wesley, 1996.

[HUR76] HURRION, R. D. **The design, use, and required facilities of an interactive visual computer simulation language to explore production planning problem**. Londres: Univ. of London, 1976. PhD Thesis.

[HUR86] HURRION, R. D. Visual interactive modelling. **European Journal of Operational Research**, [S.l.], v.23, n.3A, p.281-287, 1986.

[JOH88] JOHNSON, M. E.; POORTE, J. P. A hierarchical approach to computer animation in simulation modeling. **Simulation**, San Diego, v.50, n.1, p.30-36, Jan. 1988.

[KAM96] KAMIGAKI, T.; NAKAMURA, N. An object-oriented visual model-building and simulation system for FMS control. **Simulation**, San Diego, v.67, n.6, p.375-385, Dec. 1996.

[KEL98] KELTON, W.D; SADOWSKI, R.P.; SADOWSKI, D.A. **Simulation with Arena**. Boston: McGraw-Hill, 1998.

[KIE94] KIENBAUM, G.; PAUL, R. J. H-ACD: hierarchical activity cycle diagrams for object-oriented simulation modelling. In: WINTER SIMULATION CONFERENCE, 1994, Lake Buena Vista, Florida. **Proceedings...** Lake Buena Vista: ASA/IEEE/ACM/SCS, 1994. p.600-610.

[KIN96] KING, C. B. Taylor II manufacturing simulation software. In: WINTER SIMULATION CONFERENCE, 1996, Coronado, California. **Proceedings...** Coronado: IEEE/ACM/SCS, 1996. p.569-573.

[LAW91] LAW, A. M.; KELTON, W.D. **Simulation modeling & analysis**. New York: McGraw-Hill, 1991.

[LIN95] LINDSTAEDT, E.; WAGNER, F. R. Um ambiente de simulação visual interativa. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 12., 1995, Canela, RS. **Anais...** Canela: SBC/CLEI/II-UFRGS, 1995. p.1365-1375.

[LOM91] LOMOW, G.; BAEZNER, D. A tutorial introduction to object-oriented simulation and Sim++. In: WINTER SIMULATION CONFERENCE, 1991, Phoenix, Arizona. **Proceedings...** Phoenix: IEEE/ACM/SCS, 1991. p.157-163.

[MAR68] MARTIN, Francis F. **Computer modeling and simulation**. New York: John Wiley & Sons, 1968.

[MAR90] MARSHALL, R. et al. Visualization methods and simulation steering for a 3D turbulence model of Lake Erie. **Computer Graphics**, New York, v.24, n.2, p.89-97, Mar. 1990.

[MAR96] MARKOVITCH, N. A.; PROFOZICH, D. M. Arena software tutorial. In: WINTER SIMULATION CONFERENCE, 1996 Coronado, California, **Proceedings...** Coronado: IEEE/ACM/SCS, 1996. p.437-440.

[MCC87] McCORMICK, B. H.; DeFANTI, T. A.; BROWN, M. Visualization in scientific computing. **Computer Graphics**, New York, v.21, n.6, Nov. 1987.

[MEL85] MELAMED, B.; MORRIS, R. J. T. Visual simulation: the Performance Analysis Workstation. **Computer**, Los Alamitos, v.18, n.8, p.87-94, Aug. 1985.

[MEY88] MEYER, B. **Object-oriented software construction**. New York: Prentice Hall, 1988.

[MYE90] MYERS, B. Taxonomies of visual programming and program visualization. **Journal of Visual Languages and Computing**, [S.l.], n.1, p.97-123, 1990.

[NOL91] NOLAN, P. J.; LANE, G. M.; FEGAN, J. M. ISI - An environment for the engineering use of general purpose simulation languages. **Simulation**, San Diego, v.56, n.1, p.41-47, Jan. 1991.

[OZD91] OZDEN, M. F. Graphical programming of simulation models in an object-oriented environment. **Simulation**, San Diego, v.56, n.2, p.104-116, Feb. 1991.

[PAU94] PAUL, R. J.; HLUPIC, V. The CASM environment revisited. In: WINTER SIMULATION CONFERENCE, 1994, Lake Buena Vista, Florida. **Proceedings...** Lake Buena Vista: ASA/IEEE/ACM/SCS, 1994. p.641-648.

[PID92] PIDD, M. **Computer simulation in management science**. Chichester, England: John Wiley & Sons, 1992.

[POO90] POORTE, J. P.; DAVIS, D. A. Computer animation with CINEMA. In: WINTER SIMULATION CONFERENCE, 1990, New Orleans, Louisiana. **Proceedings...** New Orleans: IEEE/ACM/SCS, 1990. p.123-127.

[POS94] POST, F. H. et al. Visualization thechniques for vector fields with aplication to flow data. **CWI Quarterly Report**, Amsterdam, v.7, n.2, p.131-146, June 1994.

[REI87] REISS, S. P. Graphical program development with Pecan program development system. **Sigplan Notices**, New York, v.19, n.5, p.30-41, Nov. 1987.

[ROO91] ROOKS, M. A unified framework for visual interactive simulation. In: WINTER SIMULATION CONFERENCE, 1991, Phoenix, Arizona. **Proceedings...** Phoenix: IEEE/ACM/SCS, 1991. p.1146-1155.

[ROO93] ROOKS, M. A user-centered paradigm for interactive simulation. **Simulation**, San Diego, v.60, n.3, p.168-177, Mar. 1993.

[RUM97] RUMBAUGH, J. et al. **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Campus, 1997.

[SAD91] SADOWISKI, R. P. Avoiding the problems and pitfalls in simulation. In: WINTER SIMULATION CONFERENCE, 1991, Phoenix, Arizona. **Proceedings...** Phoenix: IEEE/ACM/SCS, 1991. p.48-55.

[SAR91] SARGENT, R. G. Simulation model verification and validation. In: WINTER SIMULATION CONFERENCE, 1991, Phoenix, Arizona. **Proceedings...** Phoenix: IEEE/ACM/SCS, 1991. p.37-47.

[SEI91] SEILA, A. F. Output analysis for simulation. In: WINTER SIMULATION CONFERENCE, 1991, Phoenix, Arizona. **Proceedings...** Phoenix: IEEE/ACM/SCS, 1991. p.28-36.

[SCH92] SCHRUBEN, L. W. Graphical model structures for discrete event simulation. In: WINTER SIMULATION CONFERENCE, 1992, Arlington, Virginia. **Proceedings...** Arlington: IEEE/ACM/SCS, 1992. p.241-245.

[SHA75] SHANNON, R. E. **Systems simulation - the art and science**. Englewood Cliffs, N. J.: Prentice-Hall, 1975.

[SIL91] SILICON GRAPHICS. **Iris Explorer User's Guide**. Mountain View: Silicon Graphics, 1991.

[SPE87] Special Report. Visualization in scientific computing - a synopsis. **IEEE Computer Graphics and Applications**, Los Alamitos, v.7, n.7, p.61-70, July 1987.

[SPR92] SPRINGMEYER, R. R.; BLATTNER, M. M.; MAX, N. L. A characterization of the scientific data analysis process. In: IEEE CONFERENCE ON VISUALIZATION, 1992, Boston. **Proceedings...** Boston: IEEE, 1992. p.235-242.

[STA90] STANDRIDGE, C. R. Interactive simulation (panel). In: WINTER SIMULATION CONFERENCE, 1990, New Orleans, Louisiana. **Proceedings**. New Orleans: IEEE/ACM/SCS, 1990. p.453-458.

[SWI94] SWIDER, C. L.; BAUER Jr., K. W.; SCHUPPE, T. F. The effective use of animation in simulation model validation. In: WINTER SIMULATION CONFERENCE, 1994, Lake Buena Vista, Florida. **Proceedings...** Lake Buena Vista: ASA/IEEE/ACM/SCS, 1994. p.633-640.

[THE90] THESEN, A.; TRAVIS, L. Introduction to simulation. In: WINTER SIMULATION CONFERENCE, 1990, New Orleans, Louisiana. **Proceedings...** New Orleans: IEEE/ACM/SCS, 1990. p.14-21.

[THO90] THOMASMA, T.; MAO, Y.; ULGEN, O. Manufacturing simulation in smalltalk. In: MULTICONFERENCE ON OBJECT ORIENTED SIMULATION, 1990, San Diego, CA. **Proceedings...** San Diego: SCS, 1990. p.93-97.

[TOB91] TOBIAS, A. M. Verification, validation and experimentation with visual interactive simulation models. **Journal of the Operational Research Society**, [S.l.], p.85-104, 1991.

[UPS89] UPSON, C. et al. The Application Visualization System: a computational environment for scientific visualization. **IEEE Computer Graphics and Applications**, Los Alamitos, v.9, n.4, p.30-42, July 1989.

[VUJ90] VUJOSEVIC, R. Object oriented visual interactive simulation. In: WINTER SIMULATION CONFERENCE, 1990, New Orleans, Louisiana. **Proceedings...** New Orleans: IEEE/ACM/SCS, 1990. p.490-498.

[WAG88] WAGNER, P. R. **Editor gráfico REDES**. Porto Alegre: CPGCC da UFRGS, 1988. 60p.

[WAG92] WAGNER, P. R.; WAGNER, F. R. O ambiente de simulação do sistema AMPLO. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 7., 1992, Rio de Janeiro, RJ. **Anais...** Rio de Janeiro: SBC/NCE/UFRJ, 1992. p.255-269.

[WAG96a] WAGNER, P. R.; FREITAS, C. M. D. S.; WAGNER, F. R. A new paradigm for visual interactive modeling and simulation. In: EUROPEAN SIMULATION SYMPOSIUM, 8., 1996, Genoa, Italy. **Proceedings...** Genoa: SCS, 1996. p.142-145.

[WAG96b] WAGNER, P. R.; FREITAS, C. M. D. S.; WAGNER, F. R. Um novo paradigma para modelagem e simulação interativa visual. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 9., 1996, Caxambu, MG. **Anais...** Caxambu: SBC/UFMG, 1996. p.87-94.

[THO94] THOMPSON, W. B. A tutorial for modeling with the WITNESS visual interactive simulator. In: WINTER SIMULATION CONFERENCE, 1994, Lake Buena Vista, Florida. **Proceedings...** Lake Buena Vista: ASA/IEEE/ACM/SCS, 1994. p.517-521.

[TUK77] TUKEY, J. W. **Exploratory data analysis**. Reading: Addison-Wesley, 1977.

Informática  *UFRGS*

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

"Uma Nova Abordagem para Modelagem e Simulação Interativa Visual"
por

Paulo Rech Wagner

Tese apresentada aos Senhores:

Prof. Dr. Jorge Muniz Barreto (UFSC)

Prof. Dr. Luis Henrique Rodrigues (PPGEP/UFRGS)

Prof. Dr. Carlos Eduardo Pereira (IEE/UFRGS)

Vista e permitida a impressão.
Porto Alegre, 09/08/99.

Profa. Dra. Carla Maria Dal Sasso Freitas,
Orientadora.

Prof. [illegible]
Coordenador do Curso de Pós-Graduação em Computação - UFRGS
Instituto de Informática - UFRGS