222543-0

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ambiente de Apoio ao Projeto de Circuitos Integrados baseado no World Wide Web

por

LEANDRO SOARES INDRUSIAK

Dissertação submetida à avaliação, como
requisito parcial para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Ricardo Augusto da Luz Reis
Orientador

Porto Alegre, abril de 1998

UFRGS
INSTITUTO DE INFORMÁTICA

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Indrusiak, Leandro Soares

Ambiente de Apoio ao Projeto de Circuitos Integrados baseado no World Wide Web/ por Leandro Soares Indrusiak.- Porto Alegre: CPGCC da UFRGS, 1998.

104f.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul. Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, BR-RS, 1998. Orientador: Reis, Ricardo A.L.

1. Microeletrônica. 2. CAD. 3. Ambientes Distribuídos. 4. Internet. 5. World Wide Web. 6. Apoio ao Projeto de Circuitos Integrados. 7. Java. 8. Realidade Virtual. 9. Execução Transparente de Processos. I. Reis, Ricardo A. L. II. Título.

Engenharia elétrica ~~(CPD)~~
Microeletrônica
CAD: Microeletrônica
Internet
WWW
Realidade virtual
Projeto: Circuitos integrados

UFRGS
INSTITUTO DE INFORMÁTICA
BIBLIOTECA

N.º CHAMADA	N.º REG.:
621.38-181.4(043) I41A	34747
	DATA: 10/11/98
ORIGEM: D / DATA: 26/10/98	PREÇO: R$ 30,00
FUNDO: II	FORN.: II

CNPq 3.04.03.00-6

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profa. Wrana Panizzi
Pró-Reitor de Pós-Graduação: Prof. José Carlos Ferraz Hennemann
Diretor do Instituto de Informática: Prof. Roberto Tom Price
Coordenadora do CPGCC: Profa. Carla Maria Dal Sasso Freitas
Bibliotecária-Chefe do Instituto de Informática: Zita Prates de Oliveira

Sumário

Lista de Abreviaturas

API - *Application Programming Interface*
ASIC - *Application Specific Integrated Circuit*
BNF - *Backus-Naur Form*
CAD - *Computer-Aided Design*
CGI - *Common Gateway Interface*
CIF - *Caltech Intermediate Format*
HTML - *Hypertext Markup Language*
IP - *Internet Protocol*
JDK - *Java Development Kit*
JWS - *Java Web Server*
SSI - *Server Side Includes*
TCP - *Transmission Control Protocol*
URL - *Universal Resource Locator*
VHDL- *VHSIC Hardware Description Language*
VRML- *Virtual Reality Modelling Language*
WWW- *World Wide Web*

Lista de Figuras

Lista de Tabelas

Resumo

Atualmente, o uso de ferramentas de apoio ao projeto de circuitos integrados é indispensável, devido à complexidade desses circuitos que aumenta incessantemente. O presente trabalho discute um modelo para integração de ferramentas em um ambiente único - formando um *framework* - com o objetivo de acelerar o processo de concepção dos circuitos através da automatização de tarefas, livrando o projetista de tarefas como a administração de recursos distribuídos, o armazenamento de arquivos e assim por diante.

O *framework* proposto é baseado em um ambiente amplamente conhecido: o World Wide Web. Ao utilizar o World Wide Web como base para o ambiente de integração de ferramentas, muito trabalho é poupado, uma vez que grande parte da interface gráfica e do controle de rede do *framework* já está implementada. A facilidade de acesso ao WWW também é uma grande vantagem, no caso de uma equipe de projeto distribuída.

A integração das ferramentas segue dois modelos. O primeiro é utilizado em ferramentas de maior interação com o usuário. Nesse caso, a ferramenta deve ser re-escrita para ser integrada ao ambiente na forma de applets - programas escritos com a linguagem Java que podem ser anexados a documentos WWW. O segundo modelo é utilizado em ferramentas com pouca ou nenhuma interação com o usuário. Essas ferramentas são integradas através de entradas e saídas de dados.

Usando applets Java, a funcionalidade e a interface gráfica da ferramenta são independentes de plataforma e podem ser anexadas a documentos WWW, o que faz com que a ferramenta possa ser executada na máquina do projetista. reduzindo a carga de processamento do servidor do *framework*. Já as ferramentas integradas usando o segundo modelo devem ser executadas no servidor devido à compatibilidade, já que são dependentes de plataforma.

Objetivando fundamentar e validar a proposta do *framework* baseado no WWW, uma revisão bibliográfica é apresentada, em ambos os temas: World Wide Web e CAD *Frameworks*. A partir dessa revisão e da especificação proposta, implementou-se um protótipo integrando ferramentas usando ambos os mecanismos descritos. A descrição do protótipo e suas características são apresentadas, bem como alguns pontos críticos que devem ser alvo de pesquisa em trabalhos futuros.

Palavras-chave: Microeletrônica, CAD, Ambientes Distribuídos, Internet, World Wide Web, Apoio ao Projeto de Circuitos Integrados, Java, Realidade Virtual, CAD *Frameworks*

TITLE: "A WORLD WIDE WEB BASED INTEGRATED CIRCUITS DESIGN ENVIRONMENT "

Abstract

Nowadays the use of design automation tools for integrated circuits is more necessary than ever, due to the always increasing complexity of such circuits. This work discusses a model for tool integration in a framework, in order to speed up the design flow, saving the designer from tasks such as distributed resources and file administration.

This framework is based on a well known environment: the World Wide Web. When using the World Wide Web as the base for the framework, a lot of work is saved since most of the user's graphic interface and the network management is already done. The availability of the WWW is also interesting, in the case of a distributed design team.

The integration of the tools follows two models. The first is that of interactive tools. In this case, the tool must be re-written to be integrated to the environment as applets - applications written using the Java language that can be attached to WWW documents. The second model is used on poorly or non-interactive tools. In this case, the tool is integrated by its input and output streams.

Using Java applets, the tools functionality and graphical interface are platform independent and may be attached to a WWW hyperdocument. Thus, the tool may run at the user's machine. Using this architecture, it is possible to divide the processing task among the framework server and the designer's machines. The tools that are integrated using the second model must run on the framework server due to compatibility issues, since they are platform dependent.

In order to validate the proposed web based design framework, a literature review is presented in both themes: World Wide Web and CAD Frameworks. From the literature review and the proposed specification, a prototype was implemented, integrating tools using both the mechanisms described.

The description of the prototype and its features are presented, as well as some critical points that need to be improved in future works.

Keywords: Microelectronics, CAD, Distributed Systems, Internet, World Wide Web, Integrated Circuits Design, Java, Virtual Reality, CAD Frameworks

1 Introdução

1.1 Motivação

1.1.1 Apoio ao Projeto de Circuitos Integrados

Avanços tecnológicos têm permitido a construção de circuitos integrados cada vez mais complexos. Assim, o desenvolvimento de novos ambientes de apoio ao projeto é fator determinante para que o aumento progressivo da complexidade na concepção dos circuitos integrados não torne esta tarefa inviável. Para que seja possível a automatização de etapas do processo de concepção, novas técnicas devem ser pesquisadas para que o projeto seja exeqüível dentro dos limites de tempo impostos.

Um ambiente de projeto de circuitos integrados é formado por diversas ferramentas, cada uma delas associada a uma ou mais etapas da concepção do circuito. Essas ferramentas podem ter grande interação com o usuário ou podem executar tarefas sem que o usuário as perceba. Assim, a arquitetura de integração dessas ferramentas e a forma como elas serão utilizadas é ponto de suma importância ao se estudar melhorias na eficiência do processo de concepção de circuitos. Baseado nisso, o presente trabalho propõe uma nova abordagem para a integração de ferramentas de apoio ao projeto de circuitos integrados, visando poupar o projetista da administração da distribuição das ferramentas e do uso de interfaces heterogêneas.

1.1.2 Recursos Distribuídos e Interface Gráfica

O trabalho de um projetista de circuitos integrados necessita, muitas vezes, do uso de ferramentas de diferentes origens, implementadas em diferentes plataformas. Isso obriga o projetista a alternar o ambiente de trabalho entre diferentes plataformas de hardware e/ou software, usando diferentes modelos de interface (gráfica ou não), o que torna o fluxo de trabalho mais lento. A integração das ferramentas de apoio ao projeto é então necessária para um fluxo de projeto mais eficiente.

O uso de sistemas em rede para a integração das ferramentas diminuiu o tempo perdido pelo projetista, mas não o eliminou. Ainda se faz necessária a implementação de um ambiente que poupe o projetista de administrar os recursos distribuídos, bem como ofereça a ele uma interface gráfica simples e eficiente para que possa desempenhar o projeto propriamente dito.

1.2 Objetivos

O presente trabalho propõe uma nova arquitetura para a integração de ferramentas de apoio ao projeto de circuitos integrados, visando reduzir o tempo perdido pelo projetista com a administração dos recursos distribuídos, bem como com o aprendizado do uso das interfaces de integração. O World Wide Web foi a base escolhida para esta arquitetura de integração, por sua simplicidade e abrangência.

A proposta foi abordada em dois diferentes âmbitos no decorrer do trabalho: global e local.

No nível global, tratou-se de criar um modelo para esse ambiente de desenvolvimento de circuitos integrados baseado no WWW. Utilizando redes de computadores e recursos de hipermídia, este ambiente tem como conceitos básicos a transparência da distribuição de recursos, a operação multiplataforma, a padronização de interface e a extensibilidade.

A meta atingida em âmbito local, por sua vez, pode ser considerada um estudo de caso, já que emprega e verifica a validade do modelo de ambiente de desenvolvimento de circuitos integrados citado anteriormente. O modelo é utilizado como base para a integração de uma suíte de ferramentas de apoio ao projeto desenvolvidas paralelamente ao trabalho de projeto do ambiente, bem como algumas das ferramentas já desenvolvidas dentro do Grupo de Microeletrônica da UFRGS.

1.3 Plano de Apresentação

A Figura 1.1 a seguir ilustra a organização do texto da dissertação, associando as etapas da pesquisa aos capítulos.

O Capítulo 2 traz uma revisão bibliográfica sobre ambientes de integração de ferramentas de apoio ao projeto, conceito por vezes chamado de *CAD Frameworks* e que abrange o tema do presente trabalho. Através do estudo da evolução desse conceito nos últimos anos, são analisadas as bases da estrutura e da funcionalidade de um *CAD Framework*.

O Capítulo 3 trata sobre a arquitetura cliente-servidor da rede Internet, enfocando com mais detalhe o World Wide Web e sua estruturação. São apresentados também a linguagem Java e a interface CGI.

O Capítulo 4 discute o potencial do World Wide Web de ser base de um ambiente de apoio ao projeto de circuitos integrados. Também são analisadas abordagens anteriores a este trabalho nesse assunto.

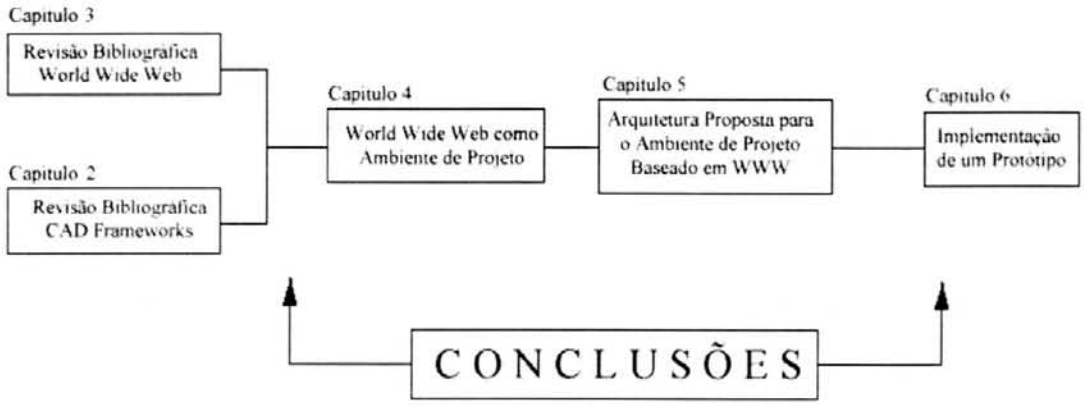

FIGURA 1.1 - Organização da Dissertação

O Capítulo 5 descreve as arquiteturas propostas para a integração de ferramentas, visando a distribuição transparente e racional de recursos, bem como a arquitetura de padronização de interface gráfica.

O Capítulo 6 descreve a implementação da primeira versão do *framework*, bem como da integração das primeiras ferramentas.

O capítulo 7 encerra a dissertação apresentando conclusões e contribuições da pesquisa, assim como levantando questionamentos e necessidades para trabalhos futuros.

2 Ambientes de Apoio ao Projeto

2.1 Introdução

Um ambiente de apoio ao projeto - também conhecido pela expressão da língua inglesa *CAD Framework* - engloba os recursos disponibilizados aos desenvolvedores de ferramentas de CAD, ao usuário final destas ferramentas e ao integrador ou administrador do ambiente de CAD em si [BAR92]. Estes recursos visam facilitar os processos efetuados pelas três classes de usuários e reduzir o tempo gasto por eles nestes processos. Assim, um CAD *Framework* deve conter mecanismos para integrar ferramentas, para auxiliar no desenvolvimento de ferramentas a serem integradas a ele e para permitir ao usuário final que utilize essas ferramentas de forma simples e flexível.

Para que isso seja possível, detalhamos nas seções seguintes as entidades formadoras de um ambiente de apoio ao projeto de circuitos integrados e a evolução desses ambientes através da funcionalidade dessas entidades, enfocando representação e administração de dados de projeto, administração do fluxo de projeto, integração e encapsulamento de ferramentas e interface com o usuário.

2.2 Estruturação dos Ambientes de Apoio ao Projeto

Segundo [BAR92], a estruturação de um *framework* segue a representação mostrada na Figura 2.1. Como pode ser visto, o ambiente consiste de vários níveis de abstração. Na maioria dos casos, os desenvolvedores de ferramentas e os administradores do ambiente têm acesso a todos esses níveis, mas ao usuário final permite-se o acesso aos níveis mais altos apenas. Detalharemos a seguir esses níveis de abstração.

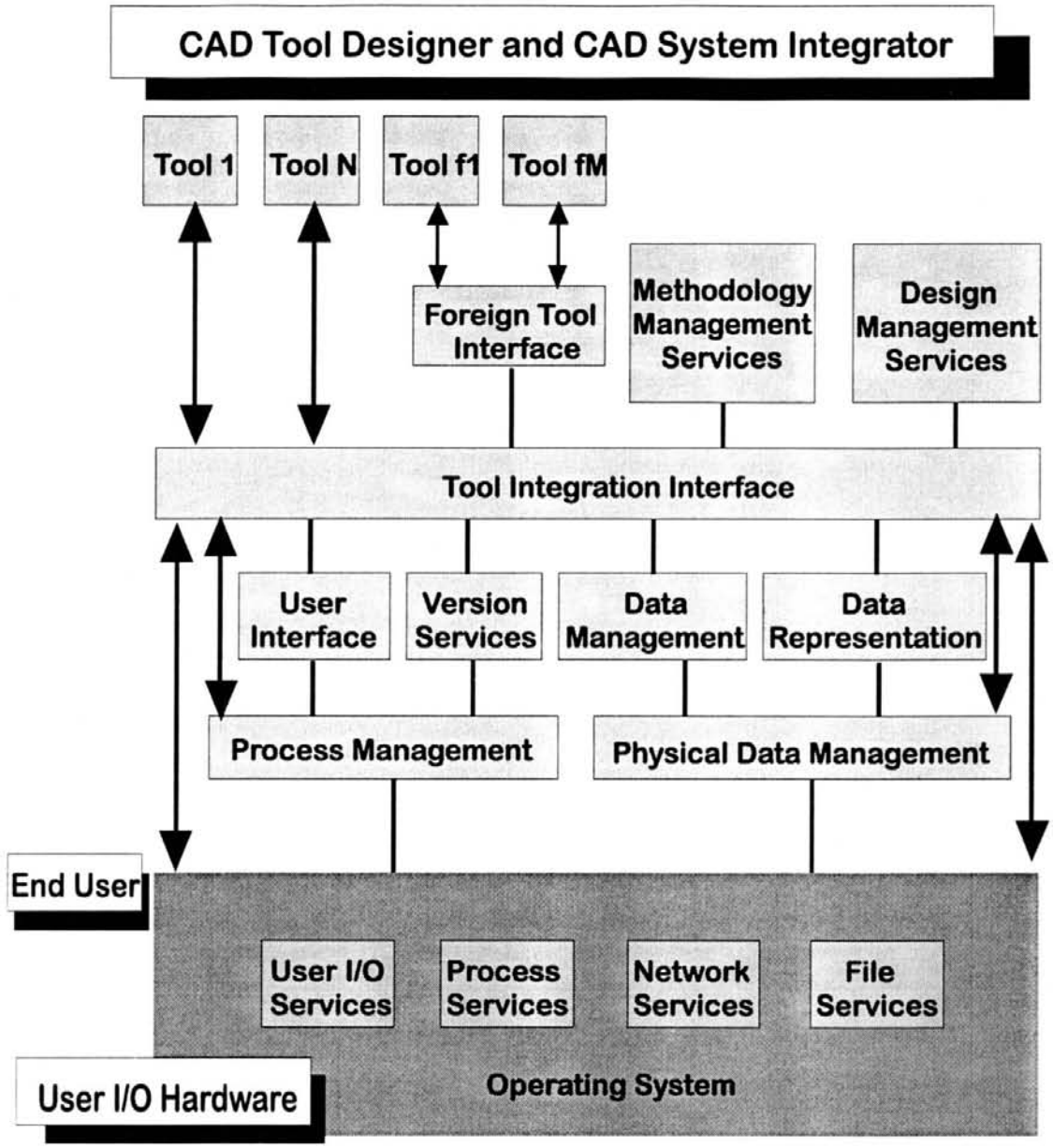

FIGURA 2.1 - Componentes de um framework

2.2.1 Serviços de Sistemas Operacionais

O ambiente de projeto é apoiado na estrutura existente criada pelo sistema operacional. Entre as funções do sistema operacional utilizadas pelo ambiente estão a manipulação e o armazenamento organizado de arquivos (*File Services*), a execução de programas (*Process Services*), a comunicação com outros sistemas (*Network Services*) e a comunicação com o usuário e o mundo externo (*User I/O Services*).

Já que nem todos os sistemas operacionais oferecem a mesma funcionalidade, deve-se implementar uma camada de interface entre o sistema operacional e o *framework*, para que as demais partes do *framework* usem um conjunto padronizado de operações, independente do sistema operacional no qual o *framework* está sendo executado. Esse conjunto de operações envolve basicamente a administração física dos dados (*Physical Data Management*) e a execução de programas (*Process Management*).

Esse nível da estrutura do *framework* deve ser utilizado por desenvolvedores de ferramentas e administradores de ambiente apenas em casos específicos. Na maioria das vezes, alterações nas configurações de serviços de sistema operacional devem ser feitas utilizando interfaces de mais alto nível.

2.2.2 Serviços de Administração de Ferramentas

Recursos para a integração de ferramentas ao *framework* também devem ser disponibilizados a desenvolvedores e administradores. Esses recursos dividem-se em (1) construtores de interface (*User Interface Services*); (2) gerenciadores de dados de projeto e administradores de acesso aos dados por ferramentas e projetistas (*Data Management Services*); (3) gerenciadores da evolução do projeto (*Version Services*); e (4) organizadores de dados e relacionamentos (*Data Representation Services*).

As ferramentas do ambiente (*Tool 1 ... Tool N*) são então integradas com o uso destas facilidades. Ferramentas projetadas para outras aplicações ou ambientes (*Tool f1 ... Tool fM*) também podem ser anexadas ao *framework*. Para isso, é necessário um outro nível - para garantir a compatibilidade - denominado *Foreign Tool Interface*.

2.2.3 Serviços de Administração de Projeto

As ferramentas que realizam as tarefas de administração de projeto são muitas vezes chamadas de meta-ferramentas, isto é, ferramentas que não manipulam os dados de projeto propriamente ditos, mas auxiliam o trabalho do projetista quando a manipulação é feita por ele. Como exemplos, temos seqüências pré-programadas para a execução de tarefas repetitivas, serviços de metodologia de projeto (*Methodology Management Services*) e serviços de visualização e avaliação do status do fluxo de projeto (*Design Management Services*).

2.3 Representação e Administração dos Dados de Projeto

As primeiras gerações de ambientes de apoio ao projeto de circuitos integrados não tinham preocupações com a administração dos dados de projeto, devido à pequena complexidade dos circuitos projetados e às características rudimentares das plataformas de hardware disponíveis na época.

Com o aumento da complexidade dos circuitos, estruturas de dados textuais e binárias - normalmente associadas a uma única ferramenta - foram criadas para representar etapas específicas do fluxo de projeto, como máscaras de layout ou netlists a nível de portas lógicas. Como as ferramentas eram implementadas independentemente umas das outras, e cada uma delas tinha seu próprio formato de dados, a maneira encontrada para permitir a integração foi com o uso de conversores (Figura 2.2).

Assim, um sistema que integrasse um número razoável de ferramentas deveria ter uma série de tradutores. Com o aumento do número de formatos de dados - resultado do aumento de ferramentas - o número de conversores passou a crescer e cada um deles passou a tratar com diversas versões de um único formato. Manter esse conjunto de conversores tornou-se então inviável, o que resultou na padronização de um formato de dados que seria entendido por todas as ferramentas. Com isso, os conversores passaram a ser internos a cada ferramenta (Figura 2.3).

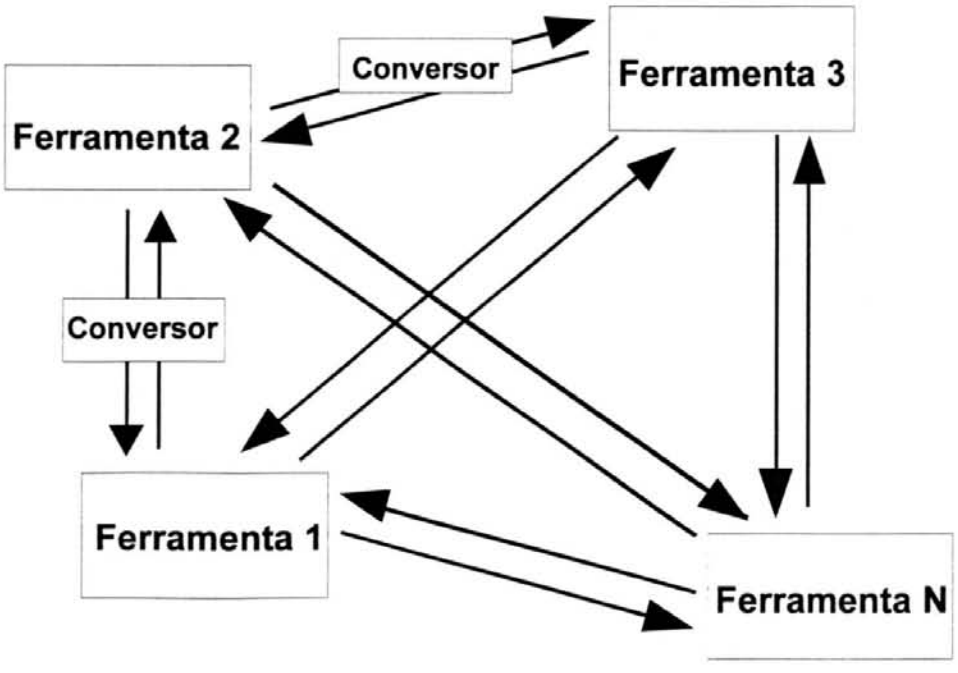

FIGURA 2.2 - Integração de ferramentas com o uso de conversores

Linguagens como a CIF (Caltech Intermediate Format) [SHE93], utilizada para descrever layout a nível de máscaras, e a SDL (Stanford Design Language) [SDL76], para descrições a nível de portas lógicas, e VHDL (VHSIC Hardware Description Language) [COM9?] são exemplos de linguagens amplamente utilizadas como formato de intercâmbio entre ferramentas.

Algumas questões em torno da administração dos dados de projeto ainda não haviam sido solucionadas com o uso de formatos padronizados de intercâmbio, como: (1) o gerenciamento de versões [WAG92]; (2) referências ativas às instâncias das partes no projeto - se uma célula for alterada, em quais circuitos essa alteração repercutirá; (3) e a consistência dos dados de projeto caso haja uma interrupção forçada no processo de edição. O uso de formatos complexos para descrição dos dados - como o EDIF e o CFI abordados em [KAH94] - e de bancos de dados para a administração dos dados de projeto passaram então a ser ponto constante nos *frameworks*.

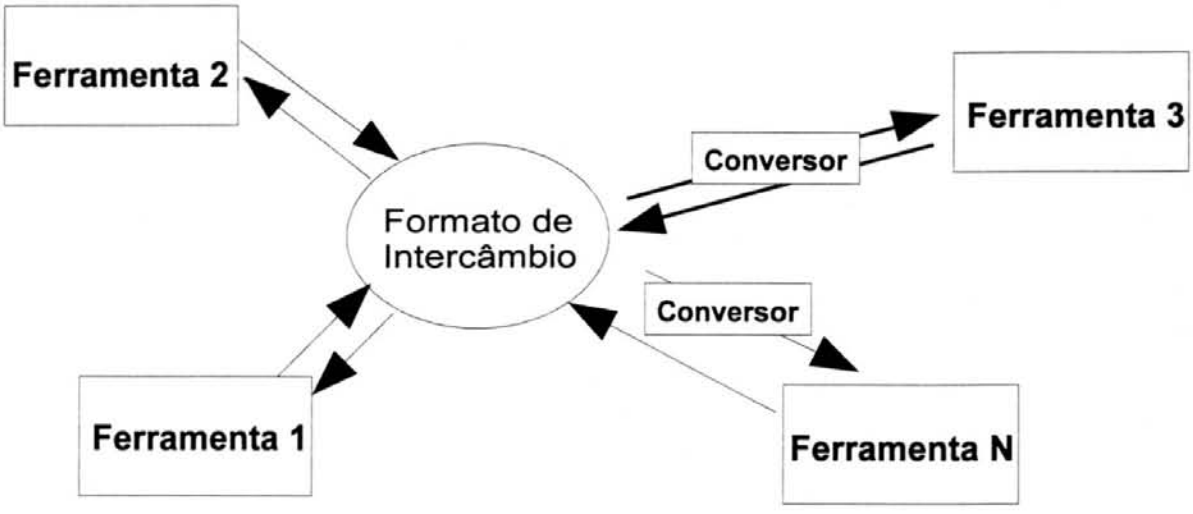

FIGURA 2.3 - Modelo de integração com formato comum de intercâmbio

2.4 Administração de Fluxo de Projeto

O fluxo de projeto de um circuito integrado é constituído pelas etapas ordenadamente cumpridas pelo projetista no processo de concepção. As ferramentas de administração de fluxo de projeto tentam capturar essas etapas e automatizar a transição entre elas, gerenciando o fluxo e o armazenamento dos dados de projeto. Por outro lado, essas etapas tem sido alteradas com o avanço da tecnologia, o que faz com que as ferramentas de controle de fluxo sejam também alteradas ou adaptadas para as novas condições de projeto. Assim, quanto mais genérica for a implementação do

modelo de fluxo de projeto, mais simples será a sua adaptação às mudanças impostas pela tecnologia [KWE95].

A princípio, o funcionamento das ferramentas de administração de fluxo de projeto se baseia em gerenciar o processo de utilização das ferramentas acessíveis através do ambiente de CAD, associando-as a cada etapa do projeto e tratando do encaminhamento dos dados de uma ferramenta a outra, fazendo as conversões quando necessárias.

Dentre os benefícios trazidos pela modelagem do fluxo de projeto em uma estrutura a ser percorrida pelo projetista está uma maior rapidez do processo de concepção, evitando deixar ao projetista a tarefa repetitiva de inicializar as ferramentas e passar os dados de uma para a outra. Outro benefício é a padronização de procedimentos, fazendo com que toda a equipe de projeto esteja seguindo uma mesma metodologia, permitindo o intercâmbio dos dados de projeto e simplificando o processo de substituição de projetistas.

Em oposição aos benefícios, uma série de problemas tiveram que ser resolvidos ou otimizados pelos projetistas de ferramentas de controle de fluxo de projeto, como: (1) controle de acesso por vários projetistas aos recursos compartilhados, evitando conflitos de acesso e mantendo a consistência dos dados [BRE95]; (2) manter o estado atual do projeto registrado e acessível, e apresentar opções ao projetista quanto às próximas etapas, permitindo inclusive a automação de algumas delas [BOS95]; (3) modelagem e administração de metodologias de projeto [JAC95]; (4) e comunicação e compartilhamento de resultados entre projetistas trabalhando cooperativamente [ROC95].

2.5 Integração e Encapsulamento de Ferramentas

Ferramentas podem ser incorporadas a um ambiente de duas formas: por integração ou por encapsulamento [CÂM95]. A integração das ferramentas pressupõe mudanças internas ao código da ferramenta, enquanto o encapsulamento opera apenas externamente.

A administração da integração e do encapsulamento de ferramentas de CAD envolve a caracterização e o controle dessas ferramentas. Para se valer do poder de automatização de tarefas oferecido pelas ferramentas, o projetista precisaria dominar uma série de requisitos, tais como a localização da estrutura de armazenamento da ferramenta, o tipo de ambiente de execução, a forma de inicialização, os formatos de dados de entrada e saída, bem como conversores desses formatos e assim por diante. Para um número pequeno de ferramentas, não é necessária nenhuma caracterização, pois o próprio projetista é capaz de assimilar as particularidades de cada uma. Mas no caso de ambientes complexos, compostos por

grande número de ferramentas, essa caracterização se faz necessária. A seguir, listamos uma série de parâmetros que são necessários à caracterização:

1.nome da ferramenta;

2.versão da ferramenta;

3.localização física da ferramenta;

4.documentação on-line da ferramenta;

5.requisitos de recursos computacionais;

6.formatos de entrada e saída e conversores;

7.condições de inicialização e de pós-execução.

De posse desses parâmetros, o ambiente de projeto pode encarregar-se do controle de inicialização, troca de dados e término da execução das diversas ferramentas, criando para o projetista uma série de comandos semelhante para todas elas. Esse controle pode ser feito tanto com a integração quanto com o encapsulamento e permite que o projetista veja todas as ferramentas do ambiente de uma forma padronizada.

Além do controle da execução das ferramentas, o ambiente pode estender sua abrangência às tarefas de balanceamento da carga [SCH95] e controle do número de usuários, autenticação de acesso, manutenção de consistência e indicação de status das ferramentas.

Normalmente, todas essas atividades são centralizadas em uma única entidade do ambiente, chamada de Tool Manager. Além das tarefas de encapsulamento das ferramentas existentes - que traz grande vantagem ao projetista - o Tool Manager pode oferecer facilidades ao desenvolvedor de ferramentas e ao administrador do ambiente, para que novas ferramentas possam ser criadas e/ou integradas ao ambiente com maior facilidade.

2.6 Interface com o Projetista

O estudo e o desenvolvimento de interfaces entre usuário e computador vêm ocorrendo desde os primórdios da computação [THI90], pois da eficiência desta interface pode depender a satisfação do usuário em relação à tarefa efetuada. Essa eficiência, por sua vez, está ligada aos recursos de software e hardware disponíveis para a implementação dessa interface [LAU90].

A evolução das interfaces entre ambientes de projeto de circuitos integrados e o projetista foi bem mais tardia do que a evolução das formas de representação e administração de dados de projeto, vista anteriormente. Isso ocorreu

dessa forma graças a limitação dos dispositivos de visualização disponíveis na época da introdução dos ambientes de apoio ao projeto.

Inicialmente, com recursos de visualização gráfica limitada, o processo de concepção de circuitos era feito sem manipulação direta dos dados do projeto. Com o advento de dispositivos de visualização mais complexos - inicialmente monocromáticos e posteriormente em cores - a visualização e a edição gráfica dos dados de projeto passaram a ser possíveis. Inicialmente, as tarefas de entrada gráfica dos dados de projeto eram feitas por pessoal especializado, pois as ferramentas não eram intuitivas e as estações de trabalho com recursos gráficos eram restritas a essa função. Percebendo que esse era um gargalo no fluxo de projeto, ferramentas de entrada gráfica mais simples foram desenvolvidas e, com a redução dos custos do hardware de suporte a gráficos, disponibilizadas aos projetistas em suas estações de trabalho. O uso de redes de computadores também foi um fator importante na difusão das estações de trabalho de saída gráfica, pois permitiu que vários projetistas estivessem conectados a um único projeto, compartilhando dados.

A introdução de interfaces em múltiplas janelas, o uso de dispositivos de ponteiro - mouse - e a interação através de menus, botões, e caixas de texto permitiu que a interatividade do projetista com o ambiente fosse maior, diminuindo o tempo de aprendizado para o uso da interface, bem como o tempo na execução de tarefas de projeto. Com isso, houve uma compensação do aumento do tempo de projeto devido ao incessante aumento na complexidade dos circuitos.

Com o amplo uso de edição gráfica de circuitos, o desenvolvimento de estruturas de dados para modelar dados em 2 dimensões, assim como de rotinas para tratar com esses dados, foi estimulado já que o layout físico de um circuito integrado passou a ser visto como um conjunto enorme de polígonos, referenciados pela posição em relação aos eixos cartesianos e a dimensão de suas arestas [SHE93, TRI90].

Atualmente, a interface do ambiente de projeto deve fornecer recursos ao usuário para manipular os dados de projeto na forma de especificações de alto nível, representação gráfica, documentação de procedimentos, entre outras. Deve também permitir a visualização de novas formas de representação - formatos padrão de intercâmbio, estatísticas de processos, informações sobre o andamento de projetos e estado das ferramentas - além dos dados finais para fabricação.

Paralelamente a isso, as interfaces devem facilitar a integração de ferramentas que, embora não manipulem diretamente os dados de projeto, tornam mais eficiente o fluxo de projeto. Ferramentas de comunicação entre projetistas, agendamento de prazos para tarefas e geração de documentação são alguns exemplos.

A arquitetura das interfaces pode ser vista como diferentes níveis de software, cada um provendo um nível mais alto de serviços de manipulação gráfica, iniciando no hardware de visualização. Basicamente, partindo do nível de controle do

dispositivo de saída gráfica, temos um nível de materialização de elementos básicos de interface (painéis, janelas, containers), um nível materializando a interface propriamente dita e finalmente uma linguagem de controle de interface gráfica a ser usada pelas ferramentas do ambiente.

Recentes pesquisas em interfaces para ambientes de projeto têm abordado conceitos como hipermídia, multimídia e integração flexível com redes de computadores - padrões informalmente estabelecidos tanto na indústria como na academia. Esses conceitos permitem a descentralização de recursos do *framework*, distribuindo-os entre máquinas interligadas por rede, tornando essa distribuição transparente para o usuário. Para esses casos, várias arquiteturas de interface estão sendo estudadas e é nesse âmbito que se inclui o presente trabalho.

3 O Ambiente World Wide Web

3.1 Introdução

O estudo dos conceitos de redes de computadores aplicados à implementação de ferramentas de apoio ao projeto - um conceito que pode ser chamado de Network CAD [IND96] - motivou o desenvolvimento de ferramentas que permitissem ao projetista de circuitos integrados a utilização destas facilidades. A integração dessas ferramentas é ponto crítico para um bom fluxo de trabalho - conforme mencionado no primeiro capítulo deste texto. Portanto, um ambiente que integre eficientemente as ferramentas distribuídas entre os diferentes computadores da rede se faz necessário.

Desconsiderando a implementação das ferramentas em si, dois grandes problemas devem ser solucionados durante a etapa de especificação do ambiente de integração de ferramentas: a implementação da interface com o usuário e a administração dos recursos distribuídos na rede.

A administração da rede requer o estudo de um protocolo já implementado - ou ainda a criação de um novo - que satisfaça as exigências do ambiente. Também se faz necessário o estudo de interfaces que compatibilizem as diferenças entre os sistemas de armazenamento e compartilhamento de arquivos das diferentes plataformas de hardware e software encontradas na rede.

A definição de interface com o usuário é também um desafio, já que deve ser simples o suficiente para ser facilmente assimilada pela equipe de projeto, bem como eficiente o suficiente para prover a equipe com os recursos necessários para executar todas as tarefas de projeto sem perda de tempo.

Esses problemas podem ser minimizados com a adoção de uma única solução: a rede Internet. A implementação de administração de rede não seria mais necessária, pois já está disponível. O protocolo TCP/IP [TAN96] provê todos os recursos necessários para o controle de transferência de dados e aplicações pela rede. Já a interface com o usuário é feita através da World Wide Web [BER94], que já é familiar à maioria dos usuários de computador.

Nas próximas seções, analisaremos as características da rede Internet, mais especificamente as do World Wide Web. Detalharemos também a linguagem de programação Java e a interface CGI, que são as maneiras que utilizaremos para integrar aplicações - no presente caso as ferramentas de apoio ao projeto - ao ambiente WWW.

3.2 Arquitetura Cliente-Servidor

Sem dúvida alguma um dos grandes motivos da atual importância dada às redes de computadores foi o advento da arquitetura cliente-servidor [TAN96]. O que anteriormente era idealizado como uma maneira de interconectar recursos computacionais - com o objetivo de aumentar a confiabilidade através da redundância - passou por um processo evolutivo que acabou por hierarquizar esses recursos.

Um computador de grande porte (ou um grupo de computadores de grande porte, conforme o caso) centraliza os recursos, que são solicitados conforme a necessidade por computadores de menor porte, conforme Figura 3.1.

Uma série de possibilidades se abriu para os desenvolvedores de aplicações, que passaram a utilizar amplamente o modelo. Compartilhamento de recursos, processamento distribuído e ambientes multiusuário foram alguns pontos cuja implementação foi impulsionada pelo uso desse modelo. A rede Internet é um exemplo típico de aplicação, pois tem a maioria dos seus serviços baseados na arquitetura cliente-servidor.

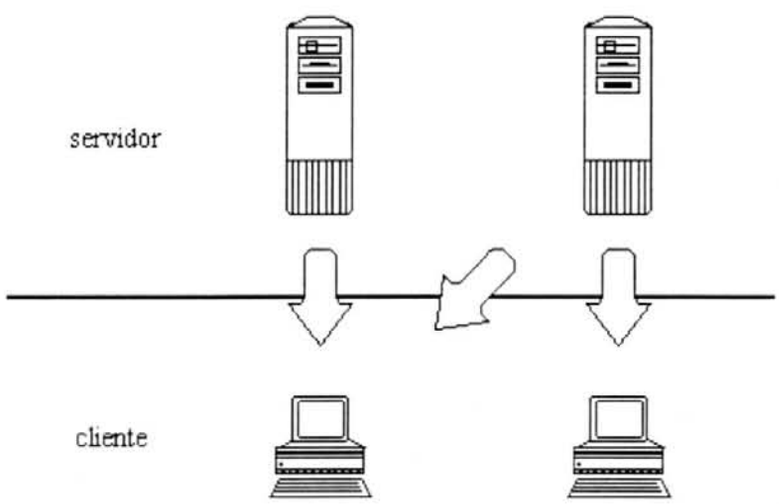

FIGURA 3.1 - Arquitetura Cliente-Servidor

3.3 World Wide Web

3.3.1 Conceito

O World Wide Web é um serviço Internet que permite recuperação flexível da informação distribuída na rede. Baseado no modelo cliente-servidor, o WWW engloba características dos serviços Internet implementados antes dele, como o FTP [REA94] e o Gopher [ALB93]:

- promover o compartilhamento de arquivos, sejam programas ou dados;
- motivar a utilização de computadores remotos;
- tornar transparentes ao usuário diferenças existentes entre sistemas de arquivos associados a estações de uma rede;
- transferir dados de maneira eficiente e confiável entre dois sistemas.

O diferencial do WWW em relação aos seus predecessores está na interface com o usuário. Para permitir um acesso eficiente à grande quantidade de dados existente na Internet, o WWW é estruturado de forma a reunir os recursos existentes na Internet em uma série de páginas interligadas - hipermídia [NIE90] - e possibilitar o uso de diferentes mídias - multimídia [WOD93] - para a apresentação desses recursos para o usuário. Detalharemos a seguir estes dois conceitos.

3.3.1.1 Hipermídia

A aplicação da hipermídia no WWW, na forma de hipertexto, trouxe simplicidade na busca de informação. Documentos contendo tópicos ou informações correlatas podem ser interligados mesmo que estejam em repositórios geograficamente afastados. Através de palavras destacadas no texto de um documento, pode-se acessar outros documentos relacionados com essas palavras. Isso permite a assimilação não linear da informação, já que cada usuário seguirá seu próprio caminho através da rede de documentos disponíveis na rede.

3.3.1.2 Multimídia

O uso de alternativas ao texto para a representação da informação mostrou que a assimilação pode ser mais eficiente. Usando formas de representação mais familiares ao usuário, o World Wide Web conseguiu atingir um púbico amplo e heterogêneo, permitindo a exploração de forma mais ampla e eficiente dos recursos

dos computadores em rede. Nas aplicações baseadas em WWW, são utilizados representações dos tipos:

- áudio : música, efeitos sonoros, voz humana
- imagens : fotos, gráficos, diagramas
- vídeos : filmes, animações computadorizadas
- realidade virtual : universos 3D interativos

3.3.2 Estrutura

O WWW teve seu início na combinação de um programa de hipertexto com a rede Internet. O objetivo era permitir que documentos em hipertexto transitassem pela rede. Esse trabalho foi liderado por Tim Berners-Lee, no CERN (Centro Europeu de Pesquisas Nucleares), e resultou na definição de um protocolo de comunicação que possibilita a transferência de imagens, sons e textos pela rede Internet: o HTTP - Hypertext Transfer Protocol [BER95].

A forma de acesso ao WWW utiliza um programa navegador, o qual interpreta e apresenta no cliente os documentos de hipertexto (escritos usando a linguagem HTML, posteriormente detalhada), e permite o acesso a diferentes servidores de informação e serviços, nas mais diversas formas: FTP (File Transfer Protocol) [REA94], Gopher (processo de busca de informações) [ALB93], HTTP (Hypertext Transfer Protocol) [BER95], Telnet (acesso remoto de computadores) [BRA94], WAIS (Wide Area Information Server) [GRA96], file (acesso a um arquivo local), etc.

A Figura 3.2 apresenta um diagrama em blocos da divisão de recursos entre cliente e servidor e a integração entre eles, compondo o WWW.

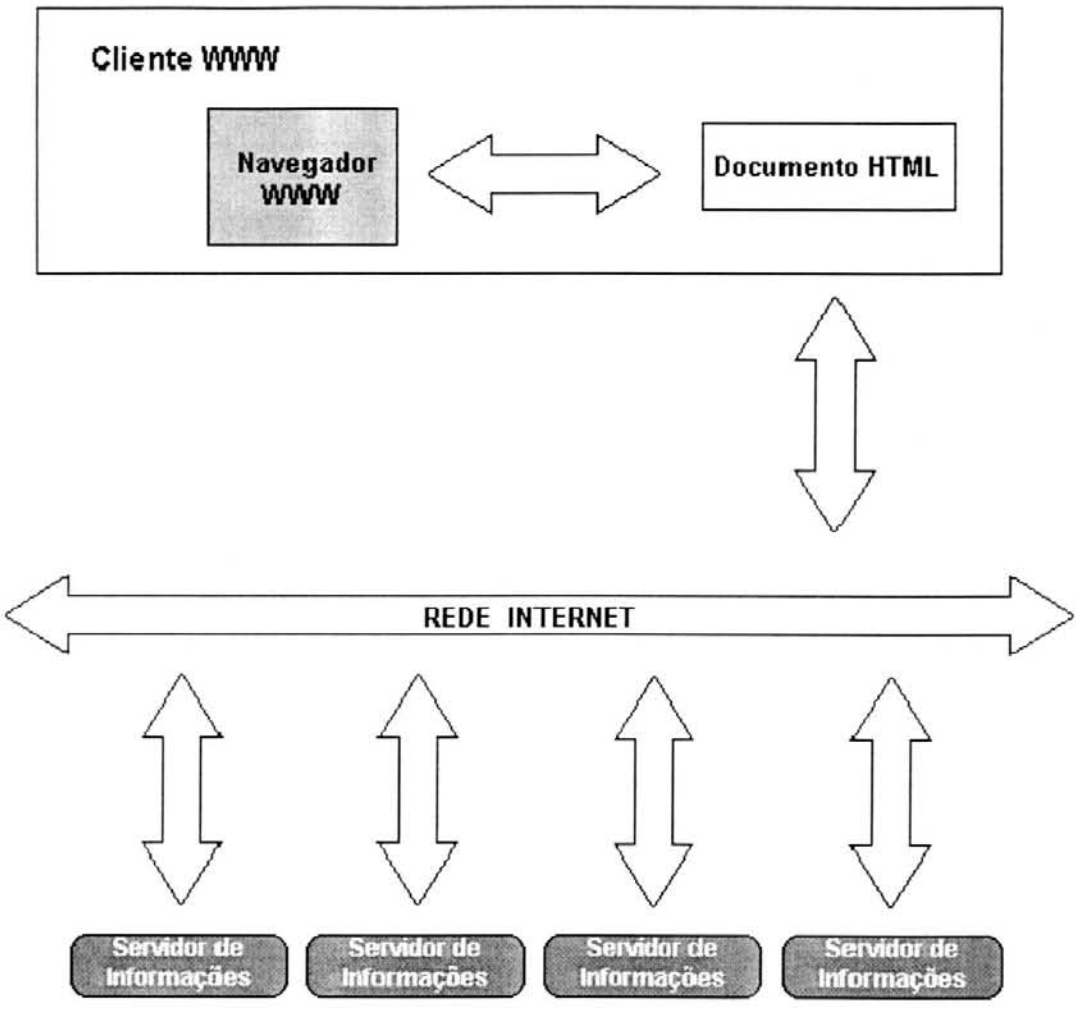

FIGURA 3.2 - Estrutura cliente-servidor do WWW

3.4 URL - Uniform Resource Locator

A diversidade de elementos que compõem a rede Internet é grande: plataformas de computadores, protocolos de comunicação, tipos de informações e documentos, programas, etc. Entretanto, a interação dos diferentes recursos existentes na rede é facilitada com o tratamento padrão existente para o endereçamento dos objetos: o *Uniform Resource Locator*.

Um objeto, para ser identificado na Internet, exige que a estrutura do seu endereço físico possua quatro partes [BER93b]:

1. identificador do protocolo de comunicação;
2. nome do servidor;
3. informação a ser passada para o servidor;

4. informação a ser utilizada pela aplicação que solicitou o acesso ao objeto.

O URL tem como primeiro elemento o identificador do protocolo de comunicação ou o método de acesso. O identificador é seguido por um separador, representado por ":". Se o endereço continuar com "//", significa que o nome do domínio do objeto na Internet está sendo especificado, isto é, o seu servidor. A seguir, é descrito o caminho (path) e o nome do arquivo que o cliente quer acessar no servidor.

Geralmente, cada URL especifica um protocolo, um servidor e um identificador, utilizado para a recuperação de um documento, por exemplo "http://www.inf.ufrgs.br/~lsi/index.html ". Nesse caso, está especificado o protocolo **http**, a máquina servidora **www.inf.ufrgs.br**, o diretório **~lsi** e o arquivo **index.html**.

3.5 HTML - Hypertext Markup Language

O HTML é uma linguagem de formatação de hipertexto que é interpretada no cliente WWW por um software de navegação (browser). Um documento HTML é um arquivo texto que é formatado através de comandos HTML. Os comandos HTML identificam e definem a estrutura e os vários componentes de um documento WWW:

- título do documento;
- estrutura hierárquica do documento com níveis de cabeçalho e nomes de seções;
- listas;
- pontos de inserção para arquivos de outras mídias;
- destaques especiais para palavras ou frases;
- áreas pré-formatadas;
- ligações de hipertexto usando URL.

Através do documento HTML, que é fornecido ao cliente pelo servidor WWW segundo o protocolo HTTP, são definidos a formatação do texto, as ligações de hipertexto que direcionam a outros documentos correlatos através de URLs e as transferências dos outros arquivos que possam compor o documento WWW, sejam imagens, vídeos, sons ou quaisquer outro tipo de arquivo que o browser suporte.

Enquanto a transferência de todos esses arquivos é feita, o browser organiza a informação e monta a interface, definida previamente pelo programador do documento HTML, com a qual o usuário vai interagir.

3.6 HTTP - Hypertext Transfer Protocol

A linguagem HTML e o protocolo HTTP trabalham em conjunto permitindo o uso de documentos hipertexto em um sistema de informações distribuído. O protocolo HTTP permite a comunicação entre clientes e servidores WWW. Esse protocolo é baseado no paradigma request/response [BER95].

A transferência de um arquivo de acordo com o protocolo HTTP é detalhada a seguir [RAS94]:

1. o cliente recebe uma URL do usuário ou de um hyperlink contido em um arquivo HTML;
2. o cliente conecta-se no servidor WWW indicado na URL;
3. o servidor WWW identifica na URL o path e o arquivo que está sendo solicitado pelo cliente;
4. o servidor WWW envia os dados requisitados pelo cliente;
5. o servidor WWW encerra a conexão.

3.7 CGI - Common Gateway Interface

O CGI é um mecanismo padrão para a execução de programas externos ao servidor Web sob requisição do usuário através do WWW através de programas gateway [MCC93]. Esse padrão estabelece como passar informações do cliente WWW para o programa gateway, e do programa gateway para o cliente WWW. O gateway é um programa executável, que pode ser escrito em qualquer linguagem, e é disparado, pelo servidor WWW, toda vez que o cliente o solicita através do seu URL. O gateway recebe solicitações e retorna as informações no formato hipertexto.

Um formulário HTML é uma boa interface para a coleta de dados no cliente WWW [RAG9?]. As informações do formulário, preenchidas pelo usuário, podem ser transmitidas para o gateway, passando pelo servidor WWW, de duas maneiras:

1. através de variáveis de ambiente - método GET;
2. através da entrada padrão - método POST.

O gateway processa as informações de entrada e pode gerar uma saída, na forma de um documento HTML, escrevendo na saída padrão. As informações escritas na saída padrão são recebidas pelo servidor WWW e passadas para o cliente. Dessa forma, os programas gateways permitem que diferentes aplicações e recursos como bancos de dados, geradores de documentos e demais aplicações locais possam

ser utilizados através do WWW. O programa gateway é requisitado pelo cliente através de uma URL do tipo:

http://www.servidor.dominio/diretorio/programa.exe?parametro1+parametro2+...

Segundo [ADI97], o padrão CGI define um protocolo de comunicação entre o servidor Web e o programa gateway:

1. o servidor recebe a URL requisitada pelo cliente;
2. o servidor inicia um novo processo no sistema, definindo as variáveis de ambiente correspondentes aos parâmetros da conexão;
3. o servidor executa o programa gateway requisitado pelo cliente no processo iniciado, e passa a esse programa os parâmetros porventura recebidos.

3.8 Linguagem Java

A linguagem Java, criada pelo grupo liderado por James Gosling na Sun Microsystems [GOS95], é uma linguagem computacional completa, independente de plataforma e com uma série de facilidades para a integração ao WWW.

Por ser uma linguagem orientada a objetos, grande parte da funcionalidade necessária para estabelecer conexões de rede [HOR96], integrar com hiperdocumentos [FLA96] e construir interfaces gráficas [ZUK97] pode ser obtida a partir do reuso do código disponibilizado pela Sun Microsystems no Java Development Kit [KRA97], conhecido como Java API. Isso faz com que aplicações escritas em Java tenham um tempo de desenvolvimento bem menor do que aplicações escritas utilizando linguagens convencionais. Além de código para ser reutilizado, o JDK contém também ferramentas rudimentares de desenvolvimento de aplicações. Essas ferramentas serviram como apoio para programadores na primeira fase da linguagem. Posteriormente, ambientes de desenvolvimento mais complexos foram disponibilizados pelas grandes *softwarehouses* [MIC96, SYB97, BOR97] .

As aplicações Java desenvolvidas para serem integradas a hiperdocumentos e materializadas em web browsers são chamadas applets. Os applets ficam armazenados em servidores web juntamente com hiperdocumentos. Quando o cliente requisita esses hiperdocumentos ao servidor, os applets são transmitidos pela rede, se instalam e são executados na máquina cliente. O applet se comunica com o sistema operacional do cliente através do Java Runtime Environment [KRA96], que é parte da maioria dos web browsers. Para cada plataforma de software/hardware existe uma versão do Java Runtime Environment, permitindo que o código Java a ser executado em qualquer uma dessas plataforma seja o mesmo.

Por serem carregados pela rede, os applets são submetidos a uma série de testes e obedecem uma série de restrições para serem executados [GON97]. A menos que o usuário explicitamente permita, o applet não pode acessar o sistema de arquivos da máquina onde está sendo executado, nem acessar outro servidor senão aquele de sua procedência. Acessos diretos à memória são completamente proibidos. Esse modelo de segurança é conhecido como *sandbox*, pois delimita um conjunto de recursos dentro do qual o applet tem permissão de ser executado. Além dos recursos limitados, o applet tem seu código analisado, para certificar que não houve nenhuma modificação nociva no código gerado pelo compilador.

Aplicações independentes de web browsers também podem ser programadas em Java, desde que o sistema operacional da máquina na qual a aplicação vai ser executada tenha a camada de interpretação do Java Runtime Environment. As aplicações Java não têm as restrições de segurança impostas aos applets.

A interface gráfica de aplicações e applets Java foi especialmente projetada para ser materializada em diferentes sistemas operacionais. Ao desenvolver uma interface, o programador especifica os elementos e o posicionamento relativo desses elementos utilizando o Abstract Window Toolkit - uma série de classes da Java API. Quando da materialização da interface, o Java Runtime Environment faz uso dos elementos de interface nativos do sistema no qual está sendo executado para mapear os elementos do Abstract Window Toolkit. Assim, os programas Java assumem a identidade visual do ambiente no qual são executadas, mantendo a familiaridade do usuário.

A linguagem Java é vista em mais detalhes no Anexo A-1 do presente texto.

4 O World Wide Web como ambiente de projeto

4.1 Introdução

Conforme visto no capítulo anterior, o World Wide Web é um ambiente independente de plataforma, o que já o qualifica como suporte para um ambiente de projeto que pretende englobar recursos distribuídos em diferentes tipos de hardware. Além disso, o World Wide Web conseguiu, nos últimos anos, uma vasta abrangência, alcançando todos os grandes centros de desenvolvimento tecnológico. Isso deve-se, em grande parte, à sua simplicidade de uso, que permite fácil interação usuário-máquina-rede.

Por já contar com a familiaridade dos usuários, o World Wide Web diminui a sobrecarga cognitiva do ambiente de projeto: as tarefas são mais eficientemente e rapidamente realizadas pelo usuário, pois este não precisa aprender a lidar com a interface do ambiente antes de realizar o trabalho propriamente dito.

Usando o WWW, a construção da estrutura do ambiente de projeto também é bastante simples, pois esta estrutura é toda baseada em hipermídia, permitindo ao usuário um acesso flexível à informação armazenada em uma rede de documentos interligados [BAL93]. Esses documentos são criados com a amplamente difundida linguagem HTML, detalhada na Seção 3.5 do presente texto.

Outra vantagem da estruturação em documentos de hipermídia - também chamados hiperdocumentos - está na facilidade de integração de material educacional e informativo com as ferramentas de projeto propriamente ditas, pois esta integração pode ser implementada através de simples ligações hipertextuais. Isso facilita o trabalho do projetista, que pode alternar entre ambiente de trabalho, help on-line e tutoriais utilizando uma única interface.

O browsers WWW suportam programas na linguagem Java, bem como interfaces para programas em outras linguagens através de CGI (Common Gateway Interface), detalhados no capítulo anterior. Assim, integrando através do ambiente WWW as ferramentas de apoio ao projeto - implementadas em Java ou acessadas via CGI - o usuário interagirá apenas com documentos HTML, usando basicamente a interface de um browser WWW à qual já está familiarizado.

A arquitetura do World Wide Web permite que se torne transparente para o usuário a distribuição de recursos em rede. Isso quer dizer que o usuário pode desconhecer em qual máquina conectada à rede - e em que tipo de plataforma de hardware - estão os recursos por ele acessados. Em se tratando de um ambiente de apoio ao projeto de circuitos integrados isso é bastante útil, pois no fluxo de

desenvolvimento de um projeto, etapas computacionalmente diferentes - como por exemplo simulação elétrica (requer processamento numérico intenso) e edição de layout (requer processamento gráfico) - podem ser processadas em diferentes máquinas da rede.

Além destas características, o World Wide Web agrega ainda ao ambiente de projeto as possibilidades de trabalho colaborativo e acesso remoto.

As próximas seções mostram os elementos necessários para a construção de um ambiente de projeto baseado no WWW que apresente as características até aqui citadas. Além disso, são apresentadas as formas de estruturação destes elementos, propostas por outros grupos de pesquisa, cuja documentação serviu de ponto de partida para a abordagem proposta pelo presente trabalho, a ser detalhada no capítulo seguinte.

4.2 Elementos de um ambiente de projeto baseado no World Wide Web

As entidades componentes de um ambiente de projeto baseado no World Wide Web podem ser divididas nos seguintes grupos:

Servidor Principal: para cadastro e autenticação de usuários e ferramentas;

Servidor de Dados: para armazenar os dados de trabalho e de projeto dos usuários;

Meta Servidor: para prover aos outros elementos a localização na rede dos dados de usuários e de ferramentas;

Servidor HTTP: web server tradicional;

Ferramenta: aplicações ou serviços cadastrados no servidor central e acessíveis via rede;

Interface Usuário-Rede: web browser com suporte a Java ou outro ambiente de execução de aplicações independentes de plataforma.

As 4 primeiras entidades listadas são detalhadas na Seção 4.2.1, pois por simplicidade podem ser englobadas pelo termo Servidor do Ambiente. A estruturação dos dados, ferramentas e documentação do ambiente é vista na Seção 4.2.2; as ferramentas são vistas na Seção 4.2.3; e a interface entre usuário e ambiente - o cliente WWW - é vista na Seção 4.2.4.

4.2.1 Servidor de Ambiente

O servidor de ambiente é um centralizador de recursos presente no ambiente de projeto. Por ser um ambiente de projeto baseado no World Wide Web, o software servidor de documentos WWW - que passamos a denominar web server - é a peça principal do servidor de ambiente. As extensões adicionadas a esse web server suprem a funcionalidade necessária para que ele complete as funções destinadas ao servidor de ambiente.

Assim, o servidor de ambiente é ponto de suma importância na estruturação do ambiente e na performance total do fluxo de projeto. Cabe a ele o armazenamento e o controle de acesso dos hiperdocumentos, das ferramentas e dos demais arquivos necessários ao funcionamento do ambiente. No caso de ferramentas a serem executadas na máquina servidora, também cabe ao web server a conexão com estas aplicações.

Detalhamos a seguir os requisitos de maior importância a serem cumpridos pelo servidor de ambiente nos aspectos de armazenamento e controle de acesso.

4.2.1.1 Armazenamento

O armazenamento de arquivos dentro de um web server segue o modelo de sistemas de arquivos, com separação dos arquivos em diretórios. A principal diferença é que os arquivos armazenados no web server são os nodos de uma rede de hiperdocumentos.

O principal cuidado que deve se ter ao definir o modelo de armazenamento de arquivos para o ambiente de projeto está na padronização. Para permitir que todas as ferramentas possam acessar o help on-line do ambiente, de outras ferramentas, arquivos de configuração, bem como trocar dados com outras ferramentas, é necessário que haja uma padronização a ser seguida em toda a estrutura de armazenamento. Essa padronização deve estar bem clara na documentação para desenvolvedores, para que sejam seguidas também nas ferramentas a serem incorporadas ao ambiente.

Assim, propõe-se que sejam armazenados no diretório raiz do web server os hiperdocumentos que dão a estruturação básica ao ambiente de projeto, bem como a ajuda on-line do ambiente e a documentação para desenvolvedores. Elementos que são usados em todos os blocos do ambiente, como barras de navegação e ícones, também podem ser armazenados nesse diretório.

Avançando na hierarquia de diretórios, devemos ter subdiretórios para as ferramentas propriamente ditas. Nesses subdiretórios é que a padronização deve ser mais rigorosa, pois é se baseando nela que as ferramentas conduzirão o usuário aos vários arquivos utilizados no decorrer do fluxo de projeto. Aos desenvolvedores de aplicações, que pretendem integrar novas ferramentas ao ambiente, essa padronização também é indispensável, para que possam se valer dos recursos já disponíveis nas ferramentas á implementadas.

Assim, cada subdiretório de ferramenta deve conter hiperdocumentos de nomes padronizados para help on-line específico da ferramenta, créditos de implementação, contato para suporte técnico e documentação para desenvolvedores, para que esses documentos possam ser acessados diretamente por outras ferramentas.

Esta padronização pode se estender tão amplamente quanto se queira, padronizando nomes de arquivos de configuração, nomes de subdiretórios de bibliotecas de arquivos, etc. A complexidade da implementação do ambiente é o principal fator determinante da complexidade desta padronização, pois quanto mais complexo for o ambiente, maior terá de ser a padronização, para tornar possível a integração das ferramentas.

Além do armazenamento físico dos hiperdocumentos, espera-se que o software da máquina servidora permita que sejam mantidos consistentes os links de hipertexto. Teste de links e gerenciamento da estrutura de hipermídia são exemplos de atividades que podem ser automatizadas, evitando a intervenção intensiva do administrador do servidor.

4.2.1.2 Controle de Acesso

Comparado com o armazenamento, o controle de acesso feito pelo web server é uma contribuição bem mais importante para o ambiente de projeto. O armazenamento de arquivos dentro do web server não difere muito do armazenamento que há em sistemas de arquivo convencionais. O controle de acesso feito pelo web server, por sua vez, deve ser mais eficiente que os sistemas de arquivos convencionais no que diz respeito à segurança. Os sistemas de arquivo limitam o acesso aos arquivos a usuários autenticados por senha. Já o web server deve ter uma camada superior de autenticação sem a intervenção do usuário, já que num sistema em rede deve-se ter o menor tráfego possível de dados . Assim, antes de ser exigida do usuário uma senha para autenticação de acesso, o web server deve autenticar a máquina cliente de onde o usuário está se conectando ao web server. Com isso, só é exigida a senha dos usuários conectados através de máquinas autorizadas a serem clientes do ambiente de projeto.

O administrador do ambiente de projeto deve então cadastrar no web server as máquinas clientes autorizadas a acessar o web server, referenciando-as por seu endereço IP numérico (por exemplo 143.54.9.2) ou pelo nome da máquina e seu

domínio Internet (por exemplo coral.inf.ufrgs.br). Também deve ser feito o cadastro de senhas e nomes de usuários, completando assim o cadastro dos dois níveis de autenticação.

Esse cadastro alimenta uma pequena base de dados, na qual o web server efetua uma procura a cada requisição de acesso a arquivo que recebe.

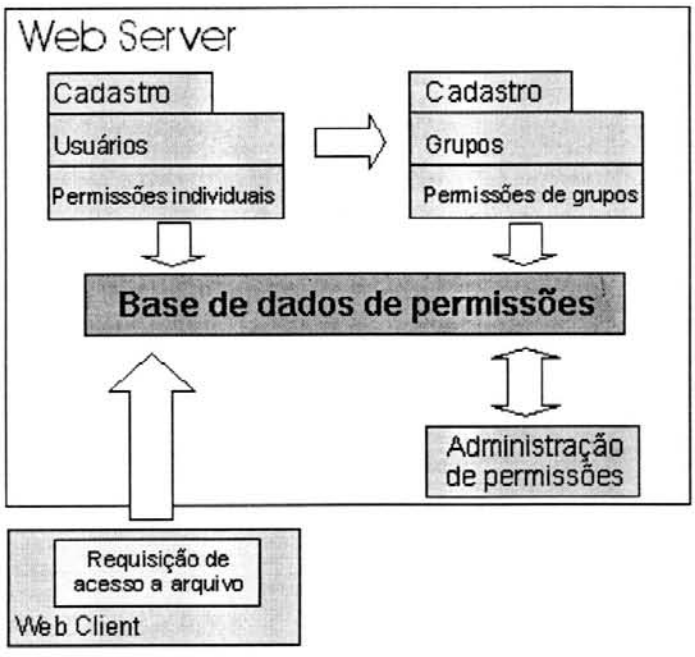

FIGURA 4.1 - Permissões de acesso ao Web Server

Como se trata de um web server para controlar o acesso de projetistas a um ambiente de apoio ao projeto, se faz necessária a definição de diferentes exigências de autenticação para diferentes arquivos e diretórios. Por exemplo, um determinado grupo de usuários deve utilizar um certo número de ferramentas, mas não pode ter autorização de utilizar uma certa biblioteca. Assim, o web server deve permitir uma administração fácil dessas permissões individuais de acesso, para que a organização criada pela coordenação da equipe de projetistas possa ser implementada no web server.

Para facilitar a administração do conjunto de permissões de acesso a arquivos, o conceito de permissão a grupos se faz importante. Com este conceito, o administrador do ambiente de projeto pode definir grupos de usuários ou máquinas entre os que já constam no cadastro e estabelecer permissões extra a esses grupos. Essas permissões, relativas a alguma tarefa a ser executada por esse grupo no decorrer do fluxo de projeto, seriam adicionadas ao conjunto individual de permissões de cada grupo. O mecanismo das permissões de acesso é mostrado na Figura 4.1.

Além de assegurar a segurança e a privacidade dos dados que trafegam no ambiente de projeto restringindo o acesso de usuários estranhos, o controle de acesso pode ser utilizado para controlar o tempo de acesso de um usuário aos recursos do ambiente de projeto. Assim, permissões temporárias podem ser dadas a usuários ou grupos, por exemplo nos casos em que o usuário paga pelo uso do ambiente de projeto apenas por um determinado período, ou ainda em casos em que é necessário um escalonamento de usuários para balancear a carga no servidor.

4.2.1.3 Conexão com Aplicações Externas

Cabe ao web server a conexão dos clientes às ferramentas a serem executadas nas máquinas servidoras. O web server recebe do cliente os dados de entrada dessas ferramentas, inicia suas execuções e direciona ao cliente os resultados. Essa conexão pode ser feita utilizando CGI - conforme visto no capítulo 3 - ou utilizando servlets, que são maneiras de se estender a funcionalidade de web servers utilizando aplicações Java [WUT97].

4.2.2 Estruturação de Hipermídia

Por ser um ambiente baseado no WWW, propomos que a estruturação do ambiente de projeto deve ser toda baseada em hiperdocumentos. Esses hiperdocumentos devem: (1) suprir ao usuário os modos de acesso a todos os recursos do ambiente; (2) servir de interface entre os usuários e as ferramentas; (3) conter toda a documentação do ambiente e integrá-la aos contextos aos quais cada parte do texto se refere.

Integrar os arquivos armazenados no servidor do ambiente através dos links de hipertexto - característica principal dos hiperdocumentos - é o mecanismo utilizado no ambiente para interligar ferramentas, documentação e demais recursos.

Além disso, os hiperdocumentos têm grande responsabilidade sobre a padronização da interface gráfica do ambiente, conforme será visto na Seção 5.2.2. Por isso é necessário o cuidado para que sua especificação seja simples e de fácil manutenção. O uso de SSI (Server Side Includes) permite que um arquivo HTML seja incluído dinamicamente dentro de outros através de comando específico. Com isso, é possível descrever uma estrutura padrão, a ser utilizada em todos os hiperdocumentos do ambiente, em um arquivo HTML e então incluir um comando de inclusão desse arquivo em todos os outros hiperdocumentos do ambiente. Esse recurso diminui a quantidade de descrições HTML redundantes armazenadas no servidor e permite uma manutenção bem mais simples, pois para editar essa estrutura padrão é necessário editar apenas um arquivo.

4.2.3 Ferramentas

Ferramentas são os blocos do ambiente que administram e manipulam os dados de projeto, auxiliando o usuário no processo de concepção de circuitos integrados, conforme visto nas seções 2.3 e 2.4. As ferramentas são armazenadas no servidor de ambiente e a interface com o usuário é feita pelo hiperdocumento e materializada pelo web browser.

4.2.4 Cliente WWW

O software que agrega toda a funcionalidade necessária à máquina cliente é o web browser. Ele é o elemento que faz toda a interface do ambiente de projeto com o usuário. Isso mostra a importância deste elemento e leva à análise dos web browsers atualmente disponíveis no desempenho da função de interface para o projetista. Mudanças da representação gráfica, extensibilidade, disponibilidade e performance são alguns pontos críticos nesta análise.

Para evitar a perda da grande vantagem do ambiente baseado em WWW - a familiaridade do usuário com o ambiente - o web browser deve sofrer o menor número de alterações possível durante o processo de adaptação para a nova função. A interface gráfica já consagrada deve ser mantida. Deve-se permitir ao projetista as parametrizações típicas para aumentar a área reservada aos documentos do ambiente, reduzindo o número de elementos ativos da interface do browser. Elementos como lista de URLs Internet freqüentemente acessadas, caixa de texto com a URL do documento atual e botões para leitura de e-mail e news podem ser desativados, já que não são pertinentes ao ambiente de projeto.

A extensibilidade também é um fator decisivo na escolha de um web browser. Entende-se por extensibilidade a possibilidade de estender a funcionalidade do web browser para que suporte elementos adicionados aos hiperdocumentos por ele materializados. Inicialmente, toda a funcionalidade de um web browser estava na materialização de hipertexto e imagens. As novas gerações de browsers passaram a suportar uma série de outros recursos, tais como inclusão de aplicativos Java e interpretação de scripts, bem como permitir ao usuário que adicione suporte a elementos específicos.

Para a especificação de um ambiente de projeto baseado do WWW, necessita-se de maneiras de integração entre as ferramentas e os documentos de hipermídia. Essa integração é detalhada no capítulo 5, mas de antemão podemos afirmar que o suporte do web browser a certos recursos é indispensável. Recursos que permitam ao usuário que interaja com as ferramentas, faça entrada de dados e visualize dados complexos serão necessários, e o suporte a esses recursos será determinante na escolha de um web browser adequado à tarefa.

No presente caso, o uso da interface CGI e de aplicações Java será intenso. Isso determina que o web browser a ser utilizado deva suportar ao menos HTML forms - recursos de formatação de hiperdocumentos que materializam formas de entradas de dados para serem enviadas ao servidor via CGI - e ao Java Runtime Environment.

A disponibilidade e a performance dos web browsers são aspectos vinculados especificamente ao produto. Isso quer dizer que a decisão deve ser tomada com base nas informações sobre as versões disponíveis no momento do uso. Por isso, esses aspectos serão abordados apenas no estudo de caso, no capítulo 6, onde é analisada uma situação específica.

4.3 Modelos de integração de ferramentas

Por ser uma área de pesquisa bastante recente, a integração de ferramentas de CAD utilizando o WWW possui poucos casos de ambientes já implementados que tenham sido publicamente documentados.

4.3.1 Sistema Henry

Em [SIL95], temos uma proposta de um CAD *Framework* baseado no WWW, para superar as limitações dos ambientes convencionais:

- uma única entidade controla o processo de projeto;
- toda a informação é disponibilizada localmente à equipe de projeto;
- o projeto é o produto de uma única organização.

O *framework* proposto comporia um ambiente de projeto baseado em informação, onde poderia interagir com outros *frameworks* semelhantes. Assim, o fluxo completo de projeto poderia ser realizado em mais de um *framework*.

Como modelo de integração de ambientes, a proposta é de grande valor, e certamente servirá como base para estudos futuros na definição da integração do ambiente proposto pelo presente trabalho com outros *frameworks*.

Já na arquitetura de integração de ferramentas de um ambiente, o modelo propõe que o *framework* seja um servidor de informações. Assim, o ambiente tem a abrangência quase que restrita aos dados de projeto, fazendo com que a maioria das ferramentas para edição desses dados sejam instaladas em cada máquina cliente.

As ferramentas instaladas nas máquinas servidoras são acessíveis remotamente, mas permitem uma interação muito pobre com o projetista.

O fluxo de projeto nessa proposta é mostrado na Figura 4.2. A máquina cliente tem acesso aos dados de projeto e às bibliotecas através de documentos, a serem trabalhados nas ferramentas instaladas localmente. O cliente tem acesso remoto às ferramentas instaladas no cliente, que devolvem os resultados também em forma de documentos.

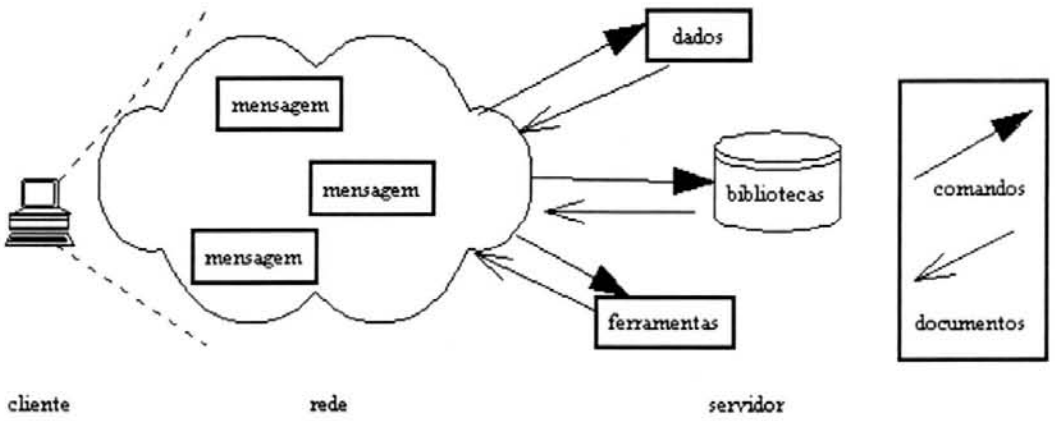

FIGURA 4.2 - Fluxo de informações no Sistema Henry

4.3.2 Sistema PPP

Em [BEN96] é proposto um novo paradigma para a integração de ferramentas de apoio ao projeto de circuitos integrados rodando em diferentes máquinas servidoras sob uma única interface com o usuário. A abordagem desse trabalho baseia-se no uso de execução remota das tarefas do projeto em um servidor, utilizando uma máquina cliente interligada a ele pela rede Internet usando o WWW.

A maior diferença entre esta abordagem e a abordagem citada na seção anterior está no fato de que o sistema PPP não necessita a instalação de nenhum software de projeto na máquina cliente. Todo a interface entre o projetista e o ambiente de projeto é feita pelo web browser (Figura 4.3).

Os ciclos de projeto no ambiente PPP são iniciados pelo projetista, que requisita no web browser a ferramenta que deseja trabalhar. O servidor envia o documento HTML que recebe os parâmetros necessários para a execução da ferramenta e a entrada de dados do projeto. O projetista alimenta esse documento com os dados necessários e comanda o envio desses dados para o servidor. De posse dos dados, o servidor executa a ferramenta e gera dinamicamente um documento contendo os resultados. Esse documento é enviado à máquina cliente, onde ocorre a análise dos

dados pelo projetista. Este ciclo ocorre tantas vezes quanto forem necessárias para que o projetista fique satisfeito com os resultados.

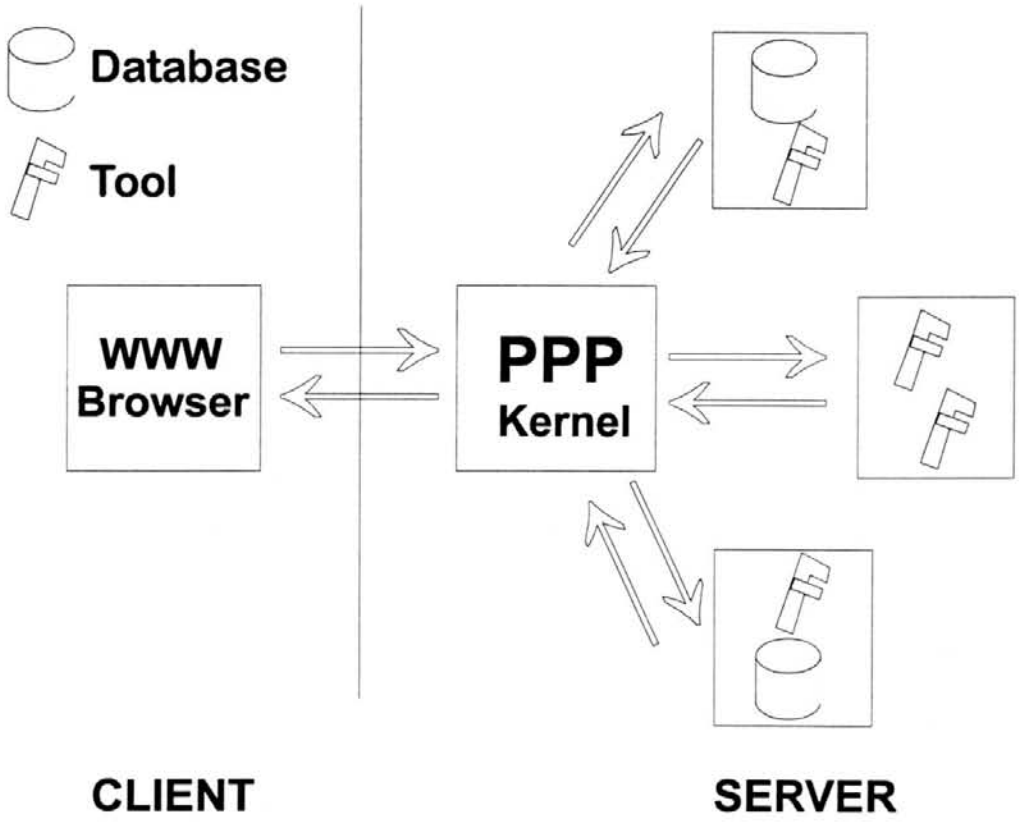

FIGURA 4.3 - Arquitetura cliente-servidor do sistema PPP [BEN96]

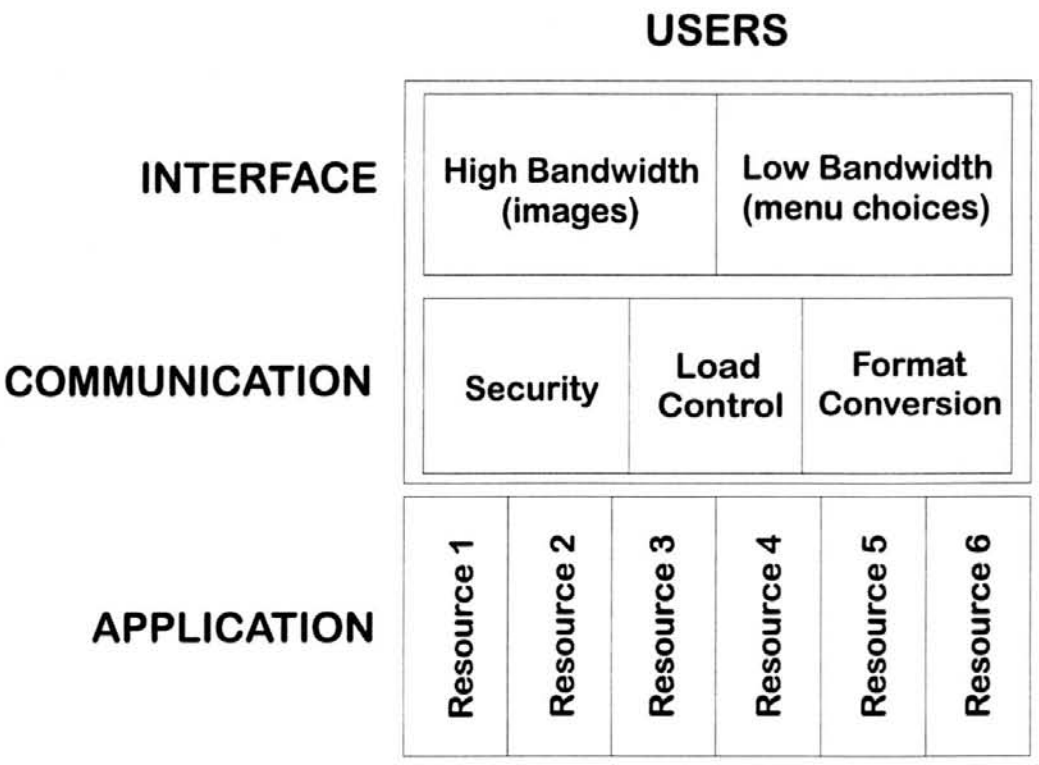

FIGURA 4.4 - Arquitetura em níveis do ambiente PPP [BEN96]

Os resultados são mostrados diretamente no web browser. Para permitir que formas de resultados mais complexas - como formas de onda, máscaras de layout, etc. - sejam visualizadas, o PPP possui uma camada de conversão entre a saída das ferramentas e o web browser do projetista, conforme Figura 4.4. Essa camada de conversão faz o mapeamento das formas de representação mais complexas para as formas que os recursos do web browser permitem visualizar.

4.3.3 Sistema WELD

Desenvolvido pela equipe de pesquisa em CAD da Universidade da California em Berkeley, o sistema WELD é um projeto ambicioso que pretende criar uma estrutura de apoio ao projeto de circuitos integrados acessível via Internet [NEW96]. Assim como o PPP, ele é constituído de máquinas cliente de pequeno porte e servidores distribuídos.

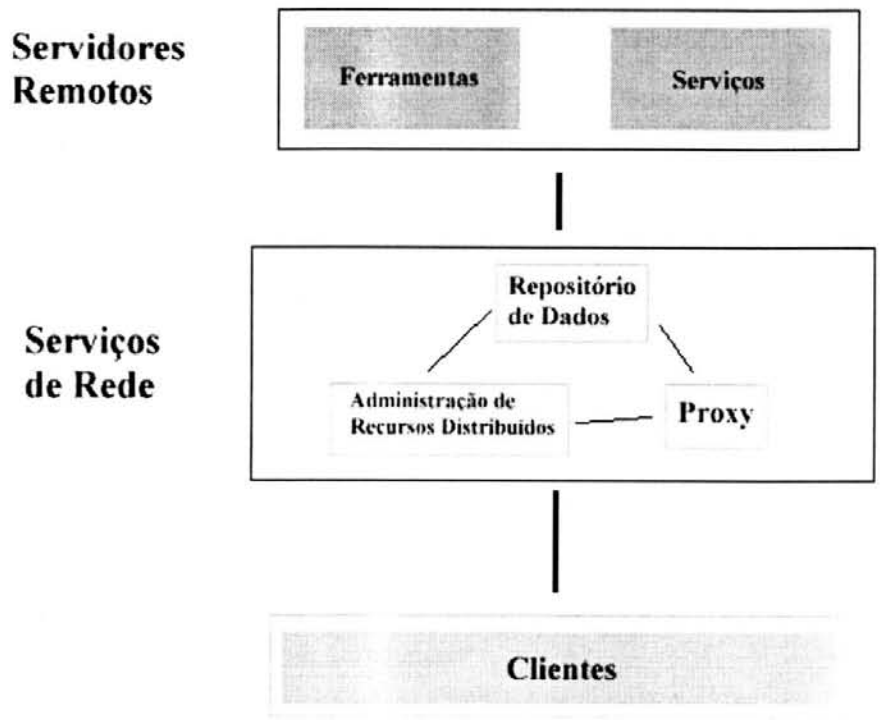

FIGURA 4.5 - Arquitetura do Sistema WELD

A arquitetura do sistema WELD detalhada em [CHA97] envolve 3 níveis: Clientes, Servidores Remotos e Serviços de Rede (Figura 4.5). Os Clientes, já citados, são máquinas de pequeno porte através das quais é acessado via rede o ambiente de projeto. Os Servidores Remotos englobam as ferramentas e os serviços disponibilizados para acesso via rede. Os Serviços de Rede, finalmente, assistem e apoiam as atividades de clientes e servidores remotos. Os mecanismos de comunicação entre essas entidades podem ser divididos em: protocolos de comunicação cliente-servidor e protocolos de comunicação cliente-repositório de dados.

A principal vantagem dessa arquitetura é a inclusão do nível de Serviços de Rede. Nesse nível, são efetuadas as tarefas de administração de recursos distribuídos, mecanismos de administração de dados de projeto, administração de versões, motores de procura de recursos entre outros. A funcionalidade adicionada pela inclusão de servidores de proxy nesse nível inclui a filtragem e conversão de arquivos, cache de dados frequentemente utilizados, restrições de segurança, controle de acesso e procura dinâmica de servidores no caso de falha.

5 Arquitetura proposta do Ambiente de Projeto baseado no WWW

5.1 Introdução

A forma de integração dos elementos componentes do ambiente de projeto tem grande contribuição na performance total do ambiente. Neste capítulo será detalhada uma nova abordagem para a estruturação dos elementos componentes do ambiente de projeto baseado no WWW, fazendo uma distribuição mais flexível de recursos entre cliente e servidor. Esta nova abordagem é, sem dúvida, uma das maiores contribuições do presente trabalho.

A arquitetura proposta vale-se do mesmo conceito visto na abordagem de [BEN96], Seção 4.3.2: o único software necessário para que o projetista acesse o ambiente de projeto deve ser o web browser. Com isso, a nova arquitetura pretende ser simples e abrangente, eliminando dificuldades iniciais para o projetista. Para o primeiro acesso ao ambiente, apenas o cadastro no web server se faz necessário.

No ambiente PPP de [BEN96], a simplicidade do software na máquina cliente teve um preço: a arquitetura teve de ser baseada na centralização do processamento das ferramentas na máquina servidora.

Para que seja possível utilizar apenas o web browser como software de interface do ambiente de projeto na máquina cliente, sem que se perca a flexibilidade de distribuição de recursos como ocorrido no caso anterior, o presente trabalho estende aquela abordagem propondo o uso de ferramentas implementadas usando a linguagem Java, que permite que as ferramentas sejam facilmente carregadas pela rede, possibilitando que a execução das mesmas ocorra também nas máquinas cliente.

Com isso, propomos explorar o modelo cliente-servidor de uma maneira diferente das atualmente encontradas em ambientes de apoio ao projeto de circuitos integrados, estendendo ao cliente a tarefa de processamento da informação. A Figura 5.1 mostra a abordagem proposta. As setas representam as conexões de rede, através das quais os dados e o código executável das ferramentas serão transferidos entre servidor e cliente. Visando simplicidade de entendimento, nessa ilustração é mostrado um único cliente conectado a um único servidor, mas o modelo permite que um cliente troque dados com mais de um servidor, bem como um servidor sirva a quantos clientes sua capacidade de processamento permitir.

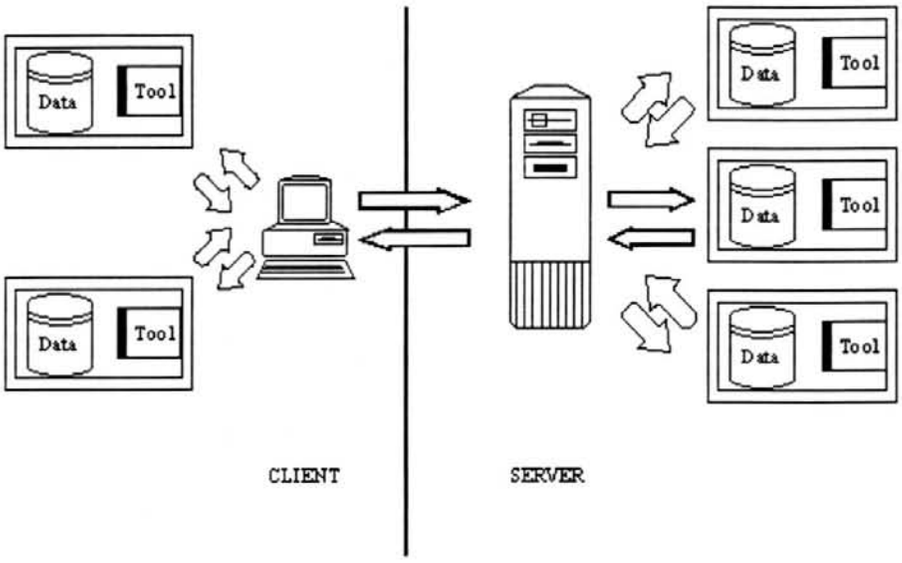

FIGURA 5.1 - Modelo cliente-servidor proposto para o ambiente de projeto

Essa nova maneira de integrar ferramentas distribuídas entre as máquinas clientes e servidoras no ambiente WWW repercute em vários âmbitos na modelagem do ambiente de projeto. Objetivando detalhar o funcionamento das principais diferenças do paradigma proposto em relação aos seus predecessores, detalharemos nas seções seguintes a arquitetura de interface entre o projetista e o ambiente, a ser materializada nas máquinas cliente, e a arquitetura de integração de ferramentas, que permite a coesão entre os recursos distribuídos.

5.2 Arquitetura de Interface entre Projetista e Ambiente de Projeto

Conforme já citado anteriormente, o projeto da interface do ambiente de projeto com o usuário é um dos pontos mais importantes do presente trabalho. A interface deve ser simples e eficiente, para ser facilmente assimilada pela equipe de projetistas e para dar a eles recursos para realização das tarefas de projeto.

Seguindo a tendência atual, optou-se pelo uso intensivo de interfaces gráficas, com janelas, menus, ícones e botões, deixando apenas o absolutamente necessário na forma de entrada textual por parte do usuário.

A seguir, definimos a arquitetura em 3 níveis - ilustrada na Figura 5.2 - para a interface entre projetista e ambiente de projeto:

5.2.1 Nível 1: Web Browser

Como o ambiente de projeto proposto é baseado no World Wide Web, podemos aproveitar a principal interface entre o usuário e o WWW: o web browser. Utilizando essa ferramenta de navegação e visualização disponível comercialmente, poupamos o trabalho de criar uma base de sustentação para o restante da interface entre o projetista e o ambiente de projeto. Somado a isso, temos também a vantagem da eliminação de boa parte da sobrecarga cognitiva - a dificuldade de se obter familiaridade - que o projetista teria ao se deparar pela primeira vez com a interface gráfica do ambiente de projeto, já que a interface do web browser provavelmente já lhe seja familiar, como é à maioria da comunidade de informática mundial nos dias de hoje.

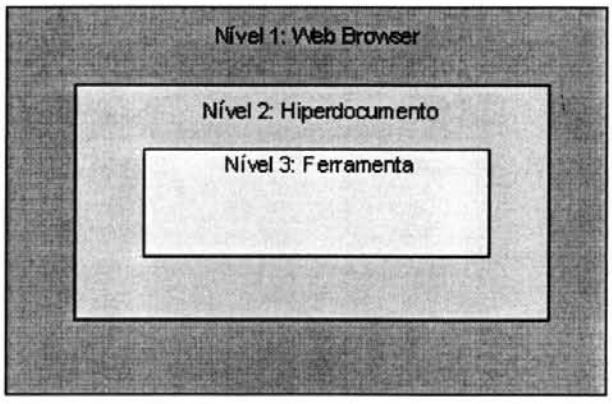

FIGURA 5.2 - Arquitetura da interface entre projetista e ambiente de projeto

5.2.2 Nível 2: Hiperdocumento

Definido o web browser como a base da interface, passamos ao segundo nível, que permite ao usuário o acesso a todas as partes do ambiente de projeto. Esse nível é implementado nos hiperdocumentos que estruturam o ambiente de projeto como um todo. Esses hiperdocumentos passam informação ao primeiro nível de interface, o web browser, sobre as ligações entre cada bloco do ambiente de projeto. Entretanto, nem sempre essas informações são formatadas de forma adequada para o ponto de vista do usuário. Tomemos o seguinte exemplo: o ambiente de projeto tem blocos distribuídos entre diversas máquinas servidoras, com aplicações anexas a um grande número de hiperdocumentos. Ao contrário do web browser, ao usuário não interessa o endereço Internet (URL) de cada uma das máquinas servidoras, tampouco os diretórios e arquivos de hiperdocumentos e aplicações. Assim, é necessário prover aos hiperdocumentos formas de facilitar ao usuário a navegação pelo ambiente de projeto sem a necessidade de envolvimento com o ambiente de rede e sistemas de arquivos.

Assim, optou-se pelo uso de barras de navegação anexas aos hiperdocumentos, bem como o uso intenso de links hipertextuais, integrando conceitualmente os blocos do ambiente de projeto. As barras de navegação são arquivos gráficos nos formatos GIF ou JPEG [MUR96] - suportados por todos os web browsers - com áreas sensibilizadas conforme um mapa descrito no corpo do hiperdocumento ao qual o gráfico é anexado. Cada área da figura leva a um outro hiperdocumento associado ao conceito mostrado graficamente ou textualmente nessa área da figura. Com isso, a real localização dos recursos distribuídos no ambiente de rede fica oculta, facilitando o trabalho do projetista.

Uma barra de navegação principal deve ser colocada na parte superior de cada hiperdocumento usando SSI - visto na Seção 4.2.2 - o que padroniza a interface e facilita o acesso aos blocos principais do ambiente de projeto. Outras barras de navegação com objetivos mais específicos podem ser utilizadas em cada um dos blocos do ambiente de projeto, sempre mantendo a identidade visual compatível com a barra de navegação principal.

5.2.3 Nível 3: Ferramenta

O nível mais alto da interface com o usuário, finalmente, é definido pela interface das aplicações propriamente ditas: as ferramentas de apoio ao projeto. Essas interfaces são inevitavelmente diversas, por motivos relacionados à sua funcionalidade e à sua arquitetura de integração com o ambiente de projeto. Assim, a padronização nesse nível de interface é mais sutil, restringindo-se ao uso dos mesmos componentes de interface (botões, menus, etc.) e a preservação em cada ferramenta das funções comuns a todas as ferramentas do ambiente, como: on-line help, créditos e contatos para suporte técnico.

5.3 Arquitetura de Integração de Ferramentas

A arquitetura de integração de ferramentas de apoio ao projeto ao ambiente de projeto é baseada na divisão dessas ferramentas em dois grupos - aplicações dos métodos de integração de ferramentas White Box e Black Box citados em [KRN91]. Essa divisão é baseada no nível de interação do projetista com a ferramenta.

5.3.1 Grupo 1: Alto grau de interação

O primeiro grupo engloba ferramentas que requerem do projetista o uso intenso de interface gráfica, como editores de esquemáticos, editores de layout e ferramentas gráficas de síntese de alto nível. Essas ferramentas precisam ser

implementadas usando linguagem Java - ou re-implementadas, o que caracteriza a relação com o modelo de integração White Box - e são anexadas a um hiperdocumento.

Quando o usuário do ambiente requisita uma ferramenta desse grupo ao servidor através de uma URL (Figura 5.3a), tanto a ferramenta quanto o hiperdocumento são transmitidos pela rede à máquina cliente e materializados na tela através do web browser (Figura 5.3b). O web browser fará a interface da ferramenta com o sistema de hardware/software da máquina cliente, que fará todo o processamento requerido pela ferramenta, havendo completa desvinculação com o servidor. Novas conexões de rede poderão ser abertas, caso a ferramenta necessite arquivos de configuração, bibliotecas ou outros arquivos. O modelo permite o armazenamento dos dados de projeto tanto no cliente quanto no servidor (Figura 5.3c).

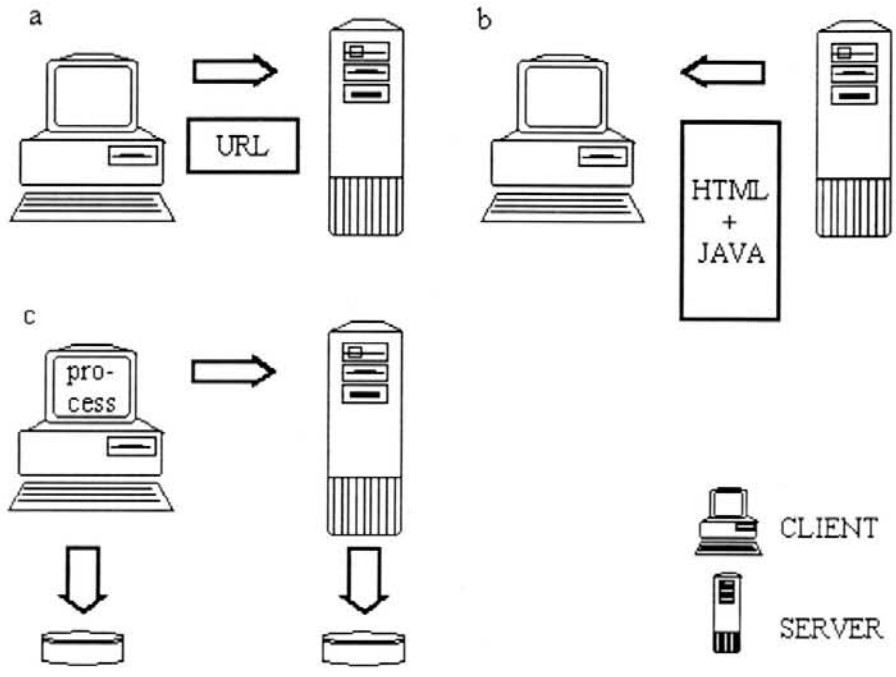

FIGURA 5.3 - Distribuição de recursos entre cliente e servidor no Grupo 1

5.3.2 Grupo 2: Baixo grau de interação

Já o segundo grupo engloba ferramentas que não exigem uma interação muito grande com o usuário. Nesse caso, o usuário age como provedor de dados e analisador de resultados, enquanto o processamento desses dados é feito pela máquina servidora. A arquitetura desse grupo aproxima-se da proposta pelo modelo de execução remota usado no ambiente PPP, visto na Seção 4.3.2, e consequentemente do modelo de integração Black Box.

Quando o usuário requisita uma ferramenta do segundo grupo (Figura 5.4a), o servidor envia um formulário onde o usuário entra com os dados a serem processados e parâmetros a serem passados para a ferramenta (Figura 5.4b). Esses dados e parâmetros são enviados pela rede ao servidor através da interface CGI (Figura 5.4c), que os processa de acordo com os parâmetros e retorna os resultados para o usuário, que os analisa no web browser (Figura 5.4d). O ciclo de interatividade é bem mais longo, já que para qualquer modificação na entrada de dados é necessária mais uma série de conexões de rede. Assim como no grupo anterior, o armazenamento dos dados pode se dar tanto no servidor quanto no cliente.

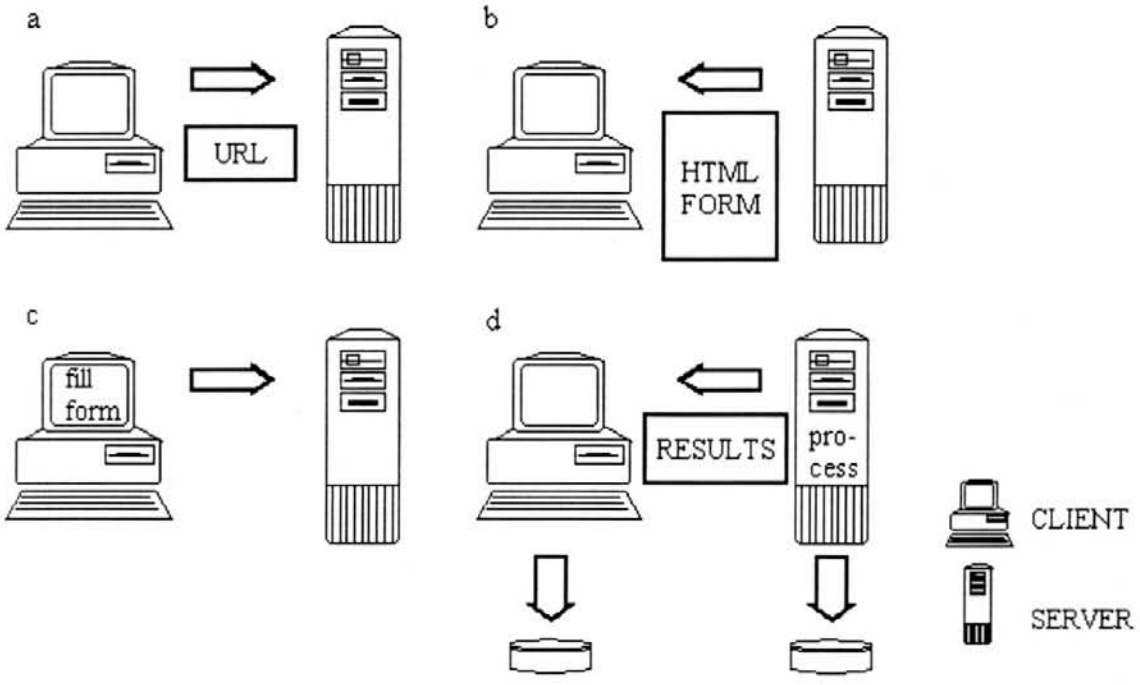

FIGURA 5.4 - Distribuição de recursos entre cliente e servidor no Grupo 2

No caso das tarefas executadas no servidor durarem um tempo excessivamente longo, uma boa prática pode ser o uso de técnicas de *push*. Utilizando essa técnica, a conexão de rede entre cliente e servidor pode ser encerrada assim que os dados são recebidos pelo servidor, desde que o cliente possa ser localizado por um endereço padrão Internet. Isso evita que a rede fique congestionada com os pacotes de controle de conexão enquanto o servidor processa os dados.

6 Implementação de um Protótipo

6.1 Introdução

Para sedimentar os conceitos estudados e validar a arquitetura proposta, foi implementado um protótipo de ambiente de apoio ao projeto de circuitos integrados baseado no WWW. O ambiente - chamado Cave - foi montado a partir de ferramentas comercialmente disponíveis, ferramentas implementadas nos projetos do Grupo de Microeletrônica da UFRGS (GME) e por ferramentas implementadas especificamente para esse fim. Nas seções seguintes, serão relatados os estudos, implementação e resultados obtidos para cada bloco do ambiente, desde o servidor de ambiente até o cliente, passando pela estrutura de hiperdocumentos, pela interface gráfica em 3 níveis e pelas ferramentas - estas usando as diferentes arquiteturas de integração.

6.2 Servidor de Ambiente

Ao iniciar a especificação do servidor de ambiente, foi cumprida uma etapa de estudo dos softwares atualmente disponíveis para esse fim. Após o estudo de servidores HTTP já consagrados, como o NCSA [NCS9?c] e o Apache [ROB95, FIE97], passamos a estudar o Java Web Server, bem mais recente, e que se mostrou o melhor software para a função. Os motivos da escolha são detalhados a seguir.

6.2.1 Java Web Server

Lançado recentemente pela Sun Microsystems, o Java Web Server mostrou-se capaz de preencher o espaço até então vago para um web server de custo acessível e de fácil administração. Além disso, o JWS quebra a corrente da dependência de plataforma do software servidor. Por ser totalmente escrito em Java, o servidor está disponível em versões em código nativo para Microsoft Windows 95, Windows NT e Sun Solaris, bem como em bytecodes, para qualquer plataforma que suporte o Java Runtime Environment [SHI97].

Além da portabilidade, outras características técnicas do JWS o diferenciam dos demais produtos de sua categoria:

6.2.1.1 Interface de Administração

A interface de administração do servidor é baseada em applets Java. Assim, o servidor pode ser administrado remotamente, através de um web browser. A interface é totalmente gráfica, onde o administrador controla através de ícones os diferentes níveis de administração: configuração, segurança, servlets e estatísticas de acesso (o JWS não necessita nenhum programa externo para análise de logs).

A administração é feita em real-time, ou seja, não é necessário reinicializar o servidor a cada mudança de configuração. As mudanças são executadas com o servidor em operação [CHP97].

A administração está dividida em 4 contextos: Setup, Monitor, Security e Servlets.

Dentro do contexto Setup (Figura 6.1) o administrador ajusta parâmetros gerais do funcionamento do servidor, como numero máximo de conexões, valor de timeout, cache de memória, aliases de arquivos, nomes de hospedeiros virtuais, arquivos de log e tipos de arquivos cadastrados (MIME). Nesse contexto também se define a porta na qual o servidor está operando.

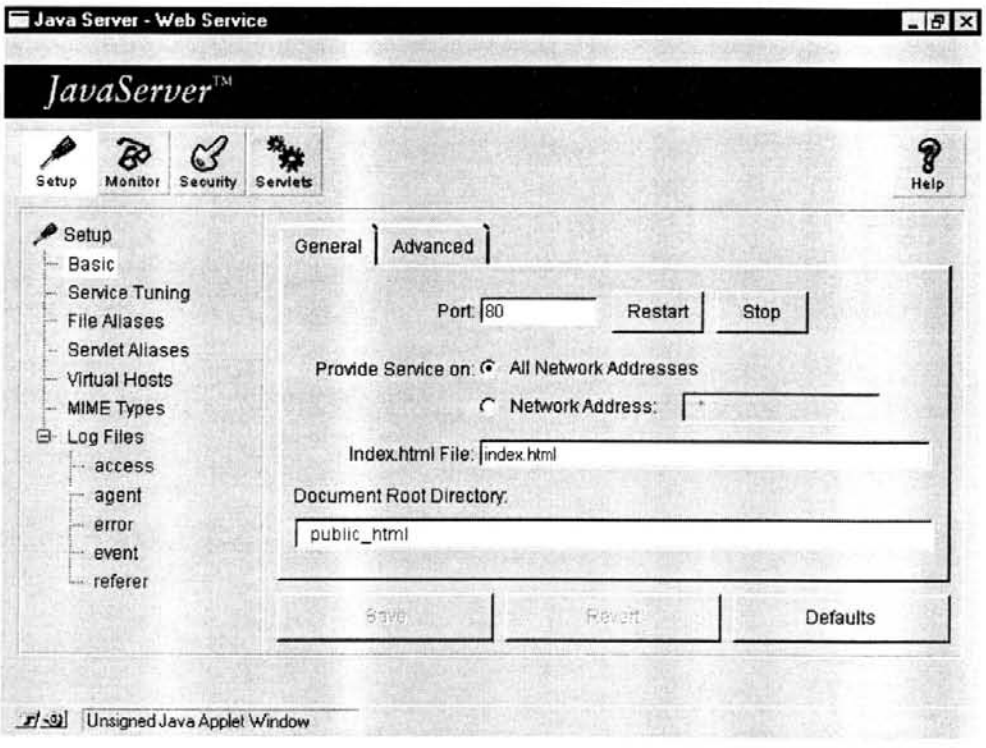

FIGURA 6.1 - Contexto de Setup

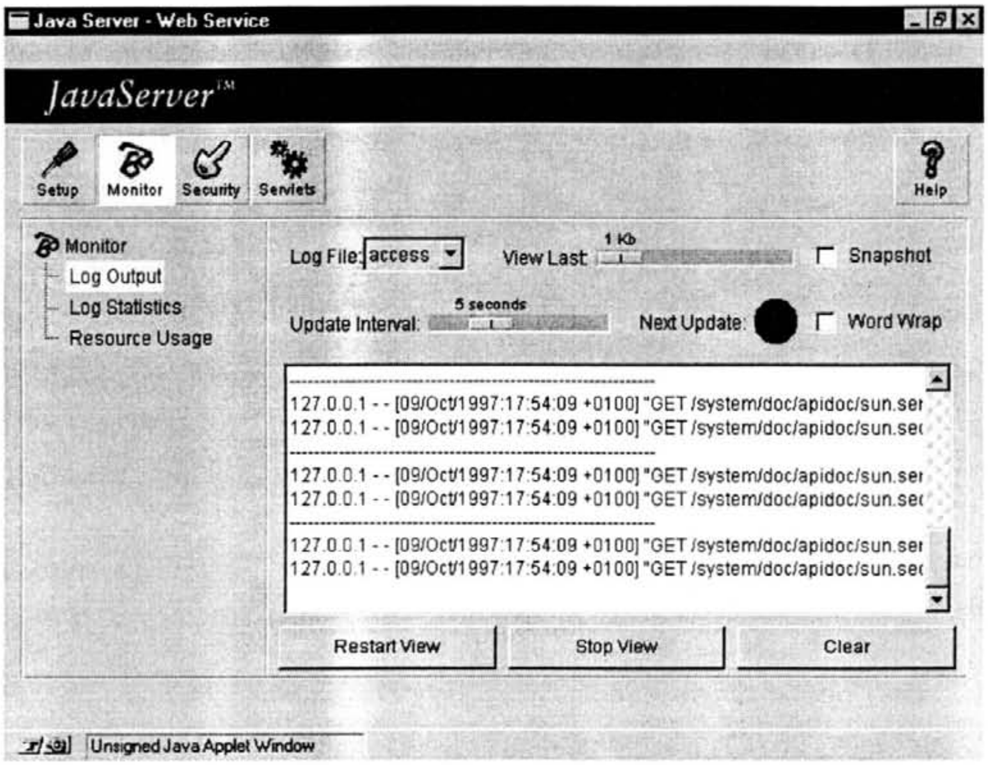

FIGURA 6.2 - Contexto Monitor - Logs

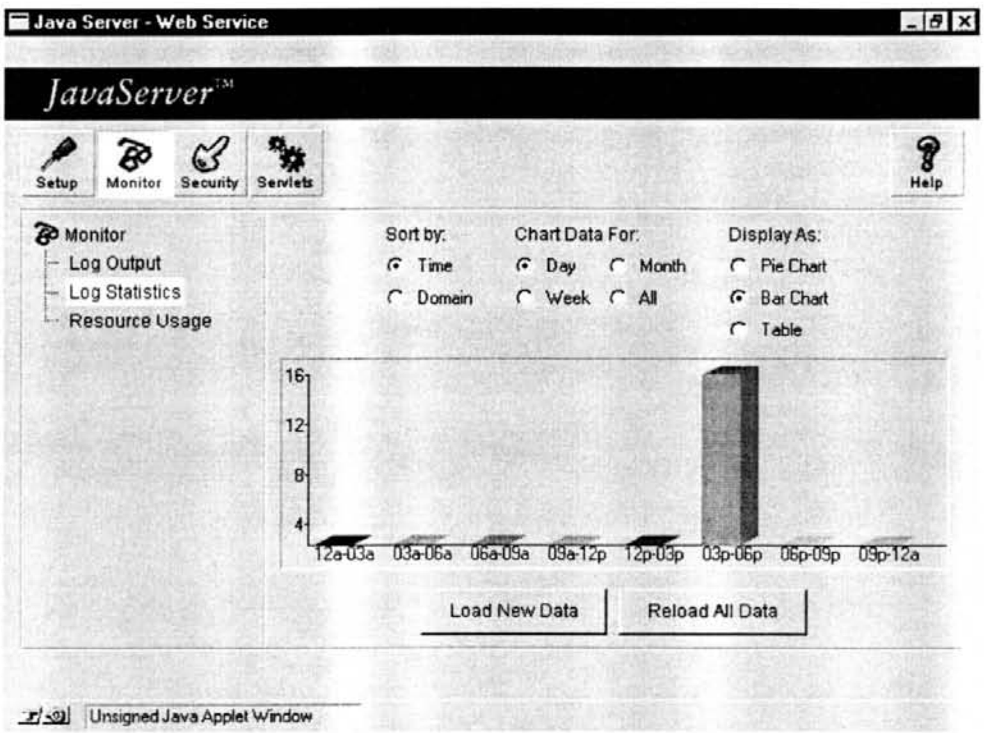

FIGURA 6.3 - Contexto Monitor - Estatísticas

No contexto Monitor, podem ser acessados os logs do servidor, que registram todo e qualquer acesso ao servidor, anotando o endereço da máquina que acessou, o horário e o arquivo acessado. Os logs podem ser vistos em estado bruto

(Figura 6.2) ou analisados por um módulo de estatística, que apresenta os resultados em forma de tabela, gráfico de barras ou diagrama de torta (Figura 6.3). Além dos logs, pode-se também monitorar a alocação de recursos de memória exigida pelo servidor.

O contexto Security (Figura 6.4) permite definir diferentes permissões de acesso a usuários e grupos de usuários. Com isso, pode-se ter arquivos e diretórios de acesso restrito armazenados no servidor, que podem ser acessados apenas mediante autenticação com senha. O administrador deve então cadastrar no web server as senhas e nomes de usuários associados a cada um dos arquivos e diretórios restritos. Esse cadastro alimenta uma pequena base de dados, na qual o web server efetua uma procura a cada requisição de acesso a arquivo que recebe.

Finalmente, o contexto Servlets permite ativar, desativar e configurar os servlets já integrados ao web server, bem como cadastrar novos servlets.

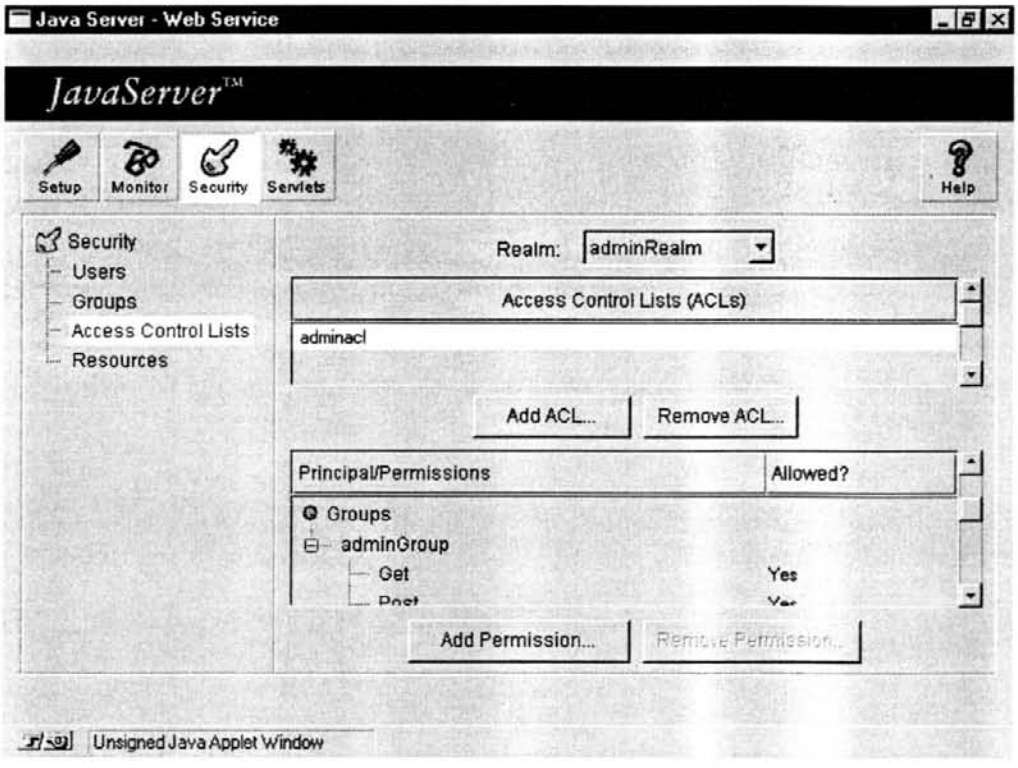

FIGURA 6.4 - Contexto Security

6.2.1.2 Extensibilidade

O conceito Java de extensibilidade, principal motivo da popularidade da linguagem na Internet, também é usado no Java Web Server. Assim como os web browsers recebem programas Java - os applets - o JWS também permite que sejam

UFRGS
INSTITUTO DE INFORMÁTICA
BIBLIOTECA

anexados a ele os chamados servlets [VOS97], para ampliar sua funcionalidade e interatividade com o usuário. Assim como os applets, os servlets são programas Java, implementados utilizando API específica. Facilitando o trabalho dos desenvolvedores, uma série de classes foi reunida e disponibilizada pela Sun no Servlet Development Kit.

Com o uso de servlets, todos os benefícios da linguagem Java são trazidos para o lado do servidor: independência de plataforma, aplicações mais robustas, acesso simples à rede, uso de threads, entre outras. Acesso a bancos de dados, webcasting, geração dinâmica de páginas, troca de dados entre usuários, transferência e upload de arquivos são algumas das funções que podem ser implementadas através de servlets. Isso quer dizer que as tarefas atualmente executadas usando CGI podem agora ser executadas através de servlets, além de várias que não podiam ser executadas usando CGI.

Sabendo que a solução utilizando servlets aumenta o poder do web server, imagina-se numa primeira análise que deva precisar máquinas de maior desempenho e mais caras, se comparada com soluções CGI. Isto não ocorre, pois as aplicações CGI são executadas como um processo separado no sistema operacional, enquanto os servlets são tratados como um ou mais threads (sub-processos) dentro do processo no qual roda o web server. Na realidade, é possível que a performance de um web server que utiliza servlets possa até ser superior [SHI97]. Além disso, eventuais erros causados pelos servlets não são propagados até o sistema operacional, como ocorre em aplicações CGI.

Outra vantagem dos servlets é a independência de plataforma. Uma vez compilados, os servlets podem ser integrados ao JWS, independentemente da plataforma de hardware e software que o servidor esteja utilizando.

6.2.1.3 Suporte

O Java Web Server dá suporte em três formas, todas baseadas em documentos HTML: a documentação para webmasters e desenvolvedores de servlets, o help on-line do applet de administração e o web site java.sun.com, que traz informações adicionais atualizadas. O help on-line do applet de administração do servidor integra a interface gráfica do applet com as páginas HTML da documentação para administradores, simplificando a tarefa de administração mesmo para administradores sem familiaridade com servidores Web.

6.2.1.4 Funcionalidade Adicionada

Juntamente com o servidor web, fazem parte do pacote JWS um proxy server para o protocolo HTTP, o Servlet Development Kit e uma série de servlets. Esses servlets já implementados e integrados ao servidor adicionam funcionalidade para tarefas como páginas geradas dinamicamente (Server Side Includes), interface para programas CGI, uso de cookies, chat, teste de hyperlinks e resposta a HTML forms. Juntamente com a instalação do servidor, há a instalação do Java Runtime Environment, permitindo o uso do software servidor mesmo que a máquina servidora ainda não tivesse suporte à linguagem Java.

6.3 Interface Gráfica

O ambiente de projeto Cave segue a arquitetura de 3 níveis para a interface gráfica descrita anteriormente. O web browser - o primeiro nível da interface - será descrito a seguir. O segundo nível - elementos de interface anexo aos hiperdocumentos - exigiu o uso de barras de navegação previsto no modelo anteriormente descrito.

Na Figura 6.5 temos a barra de navegação principal do ambiente de projeto, que vai anexada a cada hiperdocumento, na parte superior.

A área contendo o nome do ambiente e o seu logotipo - no caso o projeto Cave - leva o usuário à página inicial (Project Home), de onde podem ser acessados todos os blocos do ambiente de projeto. No canto superior direito, o logotipo da instituição que mantém o ambiente de projeto - no caso o Grupo de Microeletrônica da UFRGS - levando o usuário à homepage dessa instituição.

Já na parte inferior, botões de navegação permitem acesso rápido às entidades contextuais - partes importantes do ambiente de projeto, detalhadas na Seção 6.4: Project Manager, Tool Manager e On-Line Documentation.

FIGURA 6.5 - Barra de navegação principal do projeto Cave

O nível 3 da interface gráfica do ambiente depende da funcionalidade de cada ferramenta. A interface das ferramentas do ambiente será detalhada na Seção 6.5 a seguir.

6.4 Estrutura de Hipermídia

A estrutura de navegação do ambiente de projeto faz uma divisão de recursos entre entidades contextuais e permite que essas entidades - que são Project Home, Project Manager, Tool Manager e On-line Documentation - sejam acessíveis diretamente a partir de qualquer parte do ambiente.

O Project Home traz a apresentação do ambiente, introduzindo os conceitos utilizados e mostrando como esses conceitos são divididos entre as demais entidades da estrutura de hipermídia. O Project Manager permite o acesso do projetista às ferramentas de projeto e de administração de fluxo de projeto. Já o Tool Manager dá acesso aos recursos direcionados aos outros dois tipos de usuários do ambiente: integradores e desenvolvedores de ferramentas. A documentação on-line integra hipertextos ao ambiente de projeto, permitindo referência rápida para projetistas experientes e tutoriais para projetistas iniciantes e estudantes.

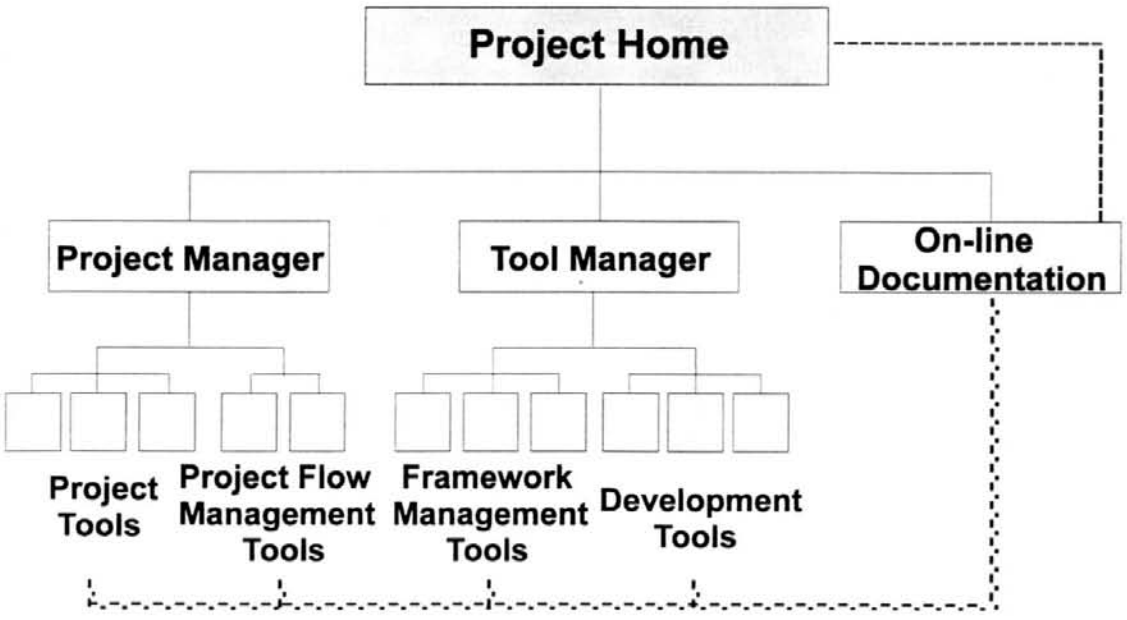

FIGURA 6.6 - Estrutura de navegação do protótipo implementado

A Figura 6.6 mostra uma representação das ligações entre as entidades da estrutura de hipermídia.

6.5 Ferramentas

Para o protótipo implementado, procurou-se utilizar as arquiteturas propostas para a integração de ferramentas ao ambiente, com o intuito de validá-las. Assim, implementou-se ferramentas de alto grau de interação com o usuário utilizando a linguagem Java, mas também criou-se interfaces para ferramentas de interatividade mais baixa. Nas seções a seguir, a forma de interação com o projetista, o desenvolvimento e a integração de cada ferramenta serão analisados.

Não foram incluídas neste protótipo ferramentas de automatização de tarefas para desenvolvedores e administradores. Para minimizar essa falha - devida ao grande número de tarefas para um limitado cronograma - disponibilizou-se para os desenvolvedores blocos de software reutilizáveis (Seção 6.5.4) e para os administradores os recursos do Java Web Server. Pretende-se, entretanto, em versões futuras do ambiente implementar ferramentas que supram essas lacunas.

6.5.1 CIF2VRML

6.5.1.1 Introdução

Uma ferramenta que permita a visualização de um modelo em 3 dimensões de um circuito integrado pode ser bastante válida [SIC95]. A partir dessa visualização pode-se ter acesso a vários pontos de vista além da vista superior (planta baixa), à qual já se está habituado pelo estudo dos modelos em 2 dimensões usualmente disponíveis. Assim, esta ferramenta pode trazer recursos tanto para o projetista de circuitos integrados quanto para o estudante desta disciplina.

A facilidade de estudar o circuito em múltiplos pontos de vista possibilita o estudo de novas construções e variações de layout. Além disso, sendo capaz de movimentar-se por entre cada camada do circuito, o estudante terá maior facilidade em entender o funcionamento do circuito e seu processo de fabricação.

A visualização mais detalhada do layout do circuito também pode ser de grande valia ao projetista de circuitos integrados, pois com a ajuda de uma ferramenta que proporcione maior visibilidade ele teria mais subsídios para buscar soluções em posicionamento, roteamento e simulações elétricas com saídas gráficas, principalmente em pontos críticos ou em circuitos de maior complexidade (maior número de camadas).

Uma maneira de implementar tal ferramenta seria a partir de um sistema gráfico de três dimensões, modelando o layout do circuito em uma estrutura

de dados contendo vértices e arestas. O acabamento final poderia ser dado usando técnicas de realismo, como sombreamento, transparência e cor. A implementação de tal sistema com certeza seria bastante custosa em termos de tempo e recursos humanos.

Uma alternativa que traz uma série de vantagens é a aplicação de conceitos de realidade virtual. Utilizando a linguagem para modelagem de realidade virtual VRML pode-se descrever a estrutura de um circuito integrado, e permitir que o aluno ou projetista se desloque por através dessa representação usando um dos vários visualizadores VRML já implementados e disponíveis comercialmente [IND96a].

Outra vantagem está no fato de que esse tipo de descrição de realidade virtual pode ser carregado através de uma conexão Internet, facilitando a criação de bibliotecas de circuitos a serem facilmente consultadas. As descrições podem inclusive estar anexas a um hiperdocumento, que provê a elas conceituação teórica e as liga a possíveis referências bibliográficas na rede. Além disso, o VRML incorpora as características multiusuário e independente de plataforma da Internet, aumentando consideravelmente a abrangência do processo de visualização 3D de circuitos integrados.

Uma forma de conversão automática para VRML de formatos de descrição de layout em duas dimensões torna-se então um fator importante para a afirmação do processo de visualização 3D como recurso educacional e de apoio ao projeto de microcircuitos. A implementação de uma ferramenta capaz de realizar essa conversão, entretanto, deve preocupar-se em manter a conectividade com a rede Internet, que dá o suporte à parte multiusuário e de acesso remoto do processo, bem como manter a capacidade de execução independente de plataforma de hardware.

6.5.1.2 Simulação usando VRML

O processo de obtenção de conhecimento é baseado na representação mental de características do universo real a partir de associações de eventos similares armazenados na memória. Isso mostra que a simulação de eventos - se mantido um nível aceitável de fidelidade em relação ao evento original - pode ser de grande valia no fornecimento de recursos para a formação das representações mentais desses eventos [BEL96a]. Atualmente esses conceitos têm sido usados com eficiência em atividades de aprendizado visando diminuição de custos e aumento da segurança do aluno, como em simuladores de vôo e de equipamentos militares.

Como outra grande utilidade da simulação, temos a possibilidade da vivência de situações fisicamente impossíveis a seres humanos, como andar na superfície do sol ou por entre as ligações de um circuito integrado.

Segundo sua especificação [BEL96], a Linguagem para Modelagem de Realidade Virtual (VRML) é uma linguagem usada para descrever simulações interativas com múltiplos participantes, conectados através da rede mundial de computadores Internet e hiperligados ao World Wide Web.

Diversas interpretações dessa breve definição estão mobilizando grupos de trabalho em todo o planeta para a exploração das potencialidades que essa nova mídia apresenta.

A idéia principal do VRML - herdada dos conceitos básicos de Realidade Virtual - é modelar aspectos do mundo real através de gráficos em três dimensões: figuras geométricas com altura, largura e profundidade [AME96]. A esses objetos prevê-se acrescentar comportamento, o que permite a simulação de eventos ao provocar a interação dos objetos entre si ou com o usuário.

Aproveitando-se da estrutura da World Wide Web, o acesso aos ambientes 3D se dá baseado no modelo cliente-servidor. Os ambientes são modelados em arquivos texto (ascii) residentes em um servidor, identificados por URL (Universal Resource Locator) do tipo:

protocolo://servidor.dominio/arvore_de_diretorios_no_servidor/arquivo.wrl

No lado do cliente, um browser recebe do usuário a URL do ambiente desejado - a partir de entrada manual ou pela ativação de um link em um hiperdocumento ou outro ambiente VRML (Figura 6.7). A partir desses parâmetros o browser abre uma conexão de rede com o servidor, requisitando o arquivo descritor do ambiente. Recebido o arquivo .wrl, ele é interpretado e materializado no vídeo, aguardando a interação com o usuário (Figura 6.8).

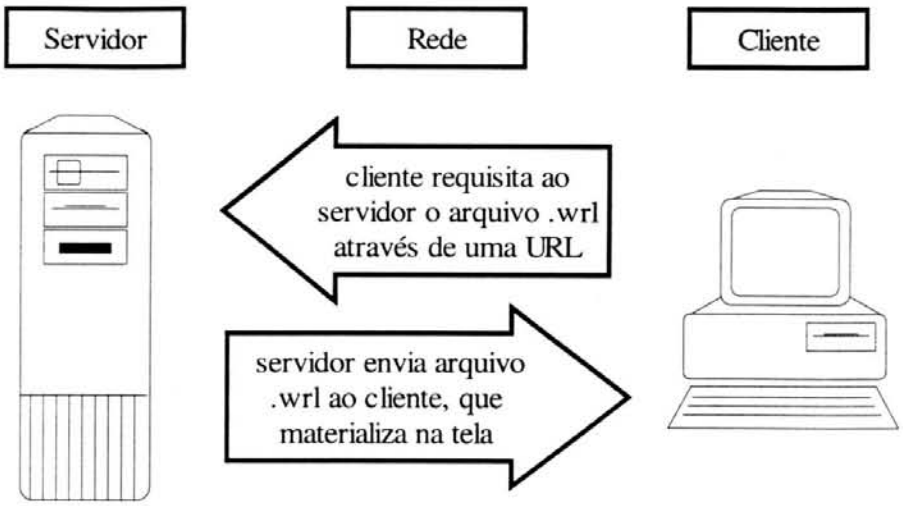

FIGURA 6.7 - Trajeto de um arquivo VRML entre cliente e servidor

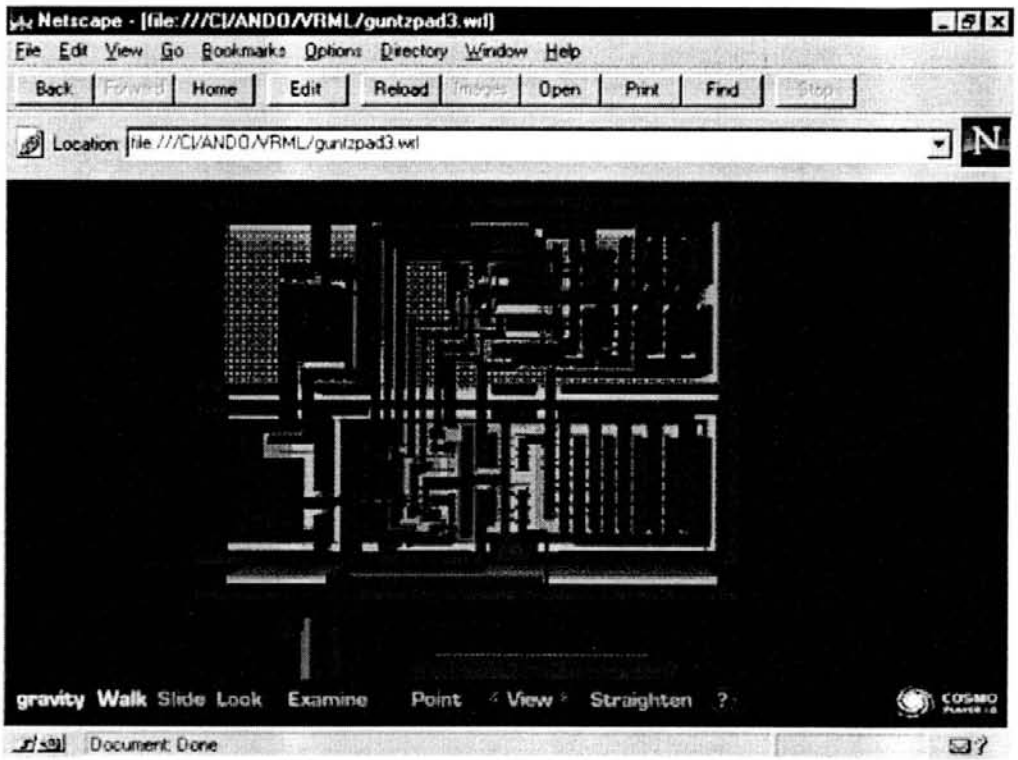

FIGURA 6.8 - Modelo VRML visualizado no browser

6.5.1.3 Representando Layout de Circuitos Integrados - Linguagem CIF

A representação de circuitos integrados usualmente se dá por meio de linguagens de descrição de layout. O layout de um circuito integrado é, de modo simplificado, um grande conjunto de polígonos divididos em diferentes camadas. Assim, as linguagens de descrição de layout devem ter recursos para abranger esse

tipo de estruturas - a dimensão e o posicionamento de cada um dos polígonos em suas respectivas camadas - com o máximo de performance.

A linguagem CIF - Caltech Intermediate Format - é uma das formas mais utilizadas para se representar layout de circuitos integrados. Sua sintaxe apresenta, através de comandos simples, as formas básicas de elementos de layout, conforme Tabela 6.1.

O número de elementos de layout em um circuito regular é muito alto, o que faz com que o tamanho do arquivo CIF seja grande também. A definição simbólica permitida pela linguagem é uma maneira de se minimizar esse problema (Tabela 6.2). Pode-se associar uma série freqüentemente usada de comandos da descrição a um símbolo e instanciar esse símbolo a cada vez que a série de comandos é necessária. Para circuitos de layout regular e repetitivo o ganho é significativo.

Tabela 6.1 - Descrição dos comandos de modelagem de elementos de layout na linguagem CIF

Sintaxe	Representação
B comprimento largura centro direção	Box: retângulo com comprimento e largura especificados por inteiros e centro e direção especificados por pares ordenados
P lista	Polygon: polígono especificado por lista ordenada de vértices
R diâmetro centro	Circle: círculo com diâmetro especificado por inteiro e centro especificado por par ordenado
W largura lista	Wire: conexão com largura especificada por inteiro e caminho especificado por lista ordenada de vértices

Tabela 6.2 - Descrição dos comandos de definição simbólica na linguagem CIF

Sintaxe	Representação
C símbolo transformação	Call: chamada de definição simbólica representada por inteiro
DD símbolo	Delete definition: Apaga definição simbólica representada por inteiro

DF	Finish definition: Encerra definição simbólica
DS símbolo indice1 indice2	Start definition: inicia definição simbólica representada por símbolo definido por inteiro

Outro aspecto importante na descrição é a diferenciação de camadas (Tabela 6.3). Ao invés de especificar para cada elemento de layout em sua descrição a camada a que pertence, a linguagem adota um indicador de camada, que ao ser interpretado passa a considerar todos os elementos subseqüentes como pertencentes àquela camada, até que encontre um indicador apontando para uma camada de nome diferente. A combinação de todas essas camadas, que devem ter sua integridade mantida a todo custo, forma o projeto global.

Tabela 6.3 - Descrição dos comandos de descrição de camadas de layout na linguagem CIF

Sintaxe	Representação
L nome	Layer: camada do circuito especificado por um nome

Por definição, o principal propósito da linguagem CIF é servir como padrão em se tratando de linguagens para troca de dados de layout entre máquinas. A partir dela devem ser criadas as formas de passar esses dados para impressoras, monitores, plotters, etc. Com isso, o acesso a todos esses recursos é facilitado, o que é ponto importante para o bom andamento do fluxo de projeto em equipe.

Assim sendo, usamos descrições CIF como base para a construção dos modelos 3D de circuitos integrados por conveniência, pois é uma linguagem amplamente difundida e adequada aos propósitos de aplicação multiusuário e multiplataforma especificados para o presente trabalho. Isso não exclui nenhuma outra forma de descrição de layout do conjunto de bases possíveis para a construção de modelos 3D de circuitos integrados, apenas padroniza uma forma de entrada.

6.5.1.4 Relacionamento entre CIF e VRML

A partir dos comandos da linguagem CIF para descrição de layout define-se um conjunto de relações entre eles e os comandos VRML de modelagem de sólidos - os chamados nodos. É necessária também uma série de critérios a serem usados na transformação de uma descrição 2D em uma equivalente 3D, que por afetar

alguns elementos do conjunto de relações mencionado acima será abordada primeiramente.

Podemos considerar dois ítens básicos na conversão de uma descrição 2D para uma equivalente 3D: a definição das propriedades do material e a especificação de valores de profundidade para as diferentes camadas.

A definição das propriedades do material preocupa-se com a aparência de cada uma das camadas do circuito. No presente caso, permitimos a parametrização de cor e transparência. Assim como nas descrições 2D de circuitos integrados, as cores têm uma padronização que serve para diferenciar as camadas. A transparência tem como objetivo permitir a visualização de camadas ocultas por outras camadas mais próximas do ponto de vista do usuário. Por isso, a parametrização da transparência deve ser deixada ao usuário no momento da transformação da descrição 2D em 3D, pois a posição do ponto de vista desejado para estudo vai determinar a porcentagem de transparência em cada camada.

Já a especificação de valores de profundidade das camadas deve manter fidelidade à tecnologia do processo de fabricação do circuito integrado, mas também permitir a parametrização pelo usuário para o caso da criação de modelos didáticos com o exagero das distâncias entre camadas, com o objetivo de facilitar a visualização de determinadas estruturas.

Dentre todos os nodos especificados na linguagem VRML podemos destacar alguns grupos de nodos que consigam modelar em 3D o layout de um circuito integrado descrito em linguagem CIF. Um desses grupos, que será usado no decorrer do trabalho, é detalhado na Tabela 6.4. As diferenças de sintaxe dos nodos nas diferentes versões da linguagem também são mostradas.

Além dos nodos detalhados na Tabela 6.4, os recursos de instanciação do VRML também se fazem necessários para a completa modelagem de um circuito integrado descrito na linguagem CIF. Utilizando o comando DEF, podemos associar determinado trecho da descrição VRML a um nome, que será usado no ato de instanciação do trecho selecionado. A instanciação se dá através do comando USE.

De posse dos dois grupos de comandos de modelagem de estruturas de layout, estabelecemos relações de equivalência entre eles. Os dados associados a cada comando CIF devem ser passados para um ou mais nodos VRML, que devem ser capazes de modelar em 3D a estrutura 2D descrita em CIF. Os parâmetros discutidos anteriormente são de grande importância nessas relações, pois fornecem o subsídio para a modelagem da terceira dimensão.

Importante notar que todos os nodos VRML 1.0 equivalentes a cada comando CIF devem estar dentro de um nodo Separator, pois o sistema de

coordenadas CIF refere-se sempre à origem do sistema cartesiano, enquanto o VRML refere-se à posição atual do cursor. No caso do VRML 2.0, cada elemento CIF tem a delimitação de um nodo Transform.

Tabela 6.4 - Nodos VRML utilizados na descrição de layout de circuitos integrados

Nodos
VRML 1.0: Translation{translation X Y Z}VRML 2.0: Transform{translation X Y Z} Faz a translação do cursor VRML nas direções X, Y e Z especificadas entre chaves na declaração do nodo. A posição do cursor VRML antes de ser mudada por qualquer nodo é a origem do sistema de referência (0,0,0).
VRML 1.0: Cube { L A P } VRML 2.0: Shape { geometry Box { size L A P}} Desenha um poliedro de seis faces retangulares com as dimensões de largura, altura e profundidade especificadas entre chaves na declaração do nodo. Se não for declarada no nodo uma ou mais dimensões, é tomado o valor default 2. O centro do poliedro será dado pela posição atual do cursor VRML.
VRML 1.0: Material {transparency *t* diffuseColor *R G B*} VRML 2.0: Shape{ appearance Appearance{ material Material {transparency *t* diffuseColor *R G B*}}} Define o material da superfície dos poliedros gerados subseqüentemente. A definição do material permite parametrização de cores, texturas, emissividade, transparência, reflexão, entre outras características, mas no caso de descrições de layout de circuitos trabalhamos apenas com cor e transparência. A cor, que permite a visualização da camada do elemento de layout, é dada por porcentagens de vermelho, verde e azul. A transparência também é dada por porcentagem, e permite que mesmo em uma vista superior seja possível visualizar as estruturas de layout dos níveis mais baixos.
VRML 1.0: Cylinder { R A } VRML 2.0: Shape { geometry Cylinder { size R A}} Desenha um cilindro de altura e raio da base definidos entre chaves na declaração do nodo.

VRML 1.0: Coordinate3 {point [*p1*, *p2* ...] }
VRML 2.0: Coordinate { point [*p1*, *p2* ...] }

Lista ordenada de pontos 3D. Cada ponto deve ter coordenadas nos 3 eixos.

VRML 1.0 e 2.0: IndexedFaceSet{ coordIndex[n, n, ..., -1, n, n, ..., -1, n, n, ...] }

Cria faces delimitadas por arestas formadas por pontos de um nodo Coordinate, referenciados através de seu número de ordem. O número de pontos de cada face é delimitado pelo número de ordem -1.

VRML 1.0: Separator{ }

Impede que as modificações feitas dentro do nodo afetem o restante da descrição.

6.5.1.4.1 Relacionamentos para comandos de modelagem de elementos de layout

O comando Box pode ser facilmente relacionado ao nodo Cube/Box, passando-lhe dados de altura e largura. Os dados relativos ao centro do Box devem ser passados a um nodo Translation, que deve ser colocado anteriormente ao nodo Cube. O valor relativo à profundidade é lido do conjunto de parâmetros da tecnologia do circuito ou fornecido pelo usuário.

Analogamente ao comando Box, o comando Circle relaciona-se ao nodo Cylinder, passando-lhe valores relativos ao diâmetro da base, que reduzido à metade fornece o valor do raio, necessário ao nodo. Como no caso anterior, um nodo Translation também se faz necessário antes do nodo Cylinder para locar o centro do elemento de layout. A altura do cilindro é lida do conjunto de parâmetros da tecnologia do circuito ou fornecida pelo usuário.

O comando Polygon já apresenta uma complexidade um pouco maior na conversão. Requer um nodo Coordinate3, que é alimentado por um algoritmo que recebe as coordenadas dos vértices passadas pelo comando Polygon e pelos parâmetros de profundidade da tecnologia do circuito ou fornecidas pelo usuário. Requer também um nodo IndexedFaceSet, que delimita o poliedro unindo os vértices e formando as faces.

O comando Wire pode ser relacionado a um conjunto de nodos análogo ao relacionado ao comando Polygon. É necessário, entretanto, um algoritmo mais complexo que o anterior para modelar o conjunto de faces que delimita o poliedro resultante.

6.5.1.4.2 Relacionamentos para comandos de definição simbólica

Os comandos de definição simbólica do formato CIF podem ser associados aos comandos de instanciação VRML. Os comandos de início e fim de definição podem ser associados a um nodo Separator precedido por um comando DEF

do VRML, enquanto o comando de chamada relaciona-se diretamente ao comando USE do VRML.

6.5.1.4.3 Relacionamentos para comandos de descrição de camadas de layout

Associamos o comando Layer ao nodo Material, diferenciando pela cor as diferentes camadas do circuito. Ao definir as camadas, precisamos fazer uma referência à profundidade e à cor de cada uma delas no modelo, buscando dados nas propriedades da tecnologia ou dados fornecidos pelo usuário.

6.5.1.5 Automatização da Conversão

Visando facilitar o processo de visualização 3D dos circuitos integrados, ferramentas de conversão automática entre o formato CIF e o VRML se fazem necessárias. Aproveitando as associações feitas anteriormente, podemos implementar uma aplicação - essencialmente um parser - que reconheça a sintaxe e a semântica do formato CIF.

Para manter as características multiplataforma do processo de visualização 3D de microcircuitos e permitir a integração da ferramenta de conversão ao ambiente Cave, a implementação foi feita usando a linguagem de programação Java.

A ferramenta é composta por 3 blocos básicos: o bloco de leitura, o bloco de conversão e o bloco de controle de entrada e saída.

Os blocos de leitura e conversão podem ser implementados manualmente, mas para acelerar o processo de implementação foram gerados com o auxílio do gerador de parsers JavaCC [SAN97]. Um gerador de parsers toma como entrada a descrição da sintaxe e da semântica da linguagem a ser lida - no caso o formato CIF - bem como as ações a serem tomadas para armazenar cada um dos elementos dessa linguagem em estruturas de dados a serem lidas pelo bloco de conversão. Essas ações são basicamente os relacionamentos vistos na seções 6.5.1.4.1, 6.5.1.4.2 e 6.5.1.4.3. O parser CIF foi disponibilizado para ser usado por outros desenvolvedores de ferramentas e é detalhado na Seção 6.5.4.1.

Assim, podemos descrever o processo de conversão em etapas:

- o bloco de leitura recebe do bloco de controle de entrada e saída o arquivo CIF que o usuário deseja converter. Ele analisa esse arquivo e armazena ordenadamente os dados encontrados

- o bloco de conversão recebe as estruturas de dados geradas pelo bloco de leitura e faz os relacionamentos dos dados de layout lidos de cada comando CIF ao conjunto de nodos VRML equivalente. O arquivo VRML resultante é passado ao bloco de controle de entrada e saída, que direciona para o armazenamento ou para a visualização.

As etapas são ilustradas simplificadamente na Figura 6.9:

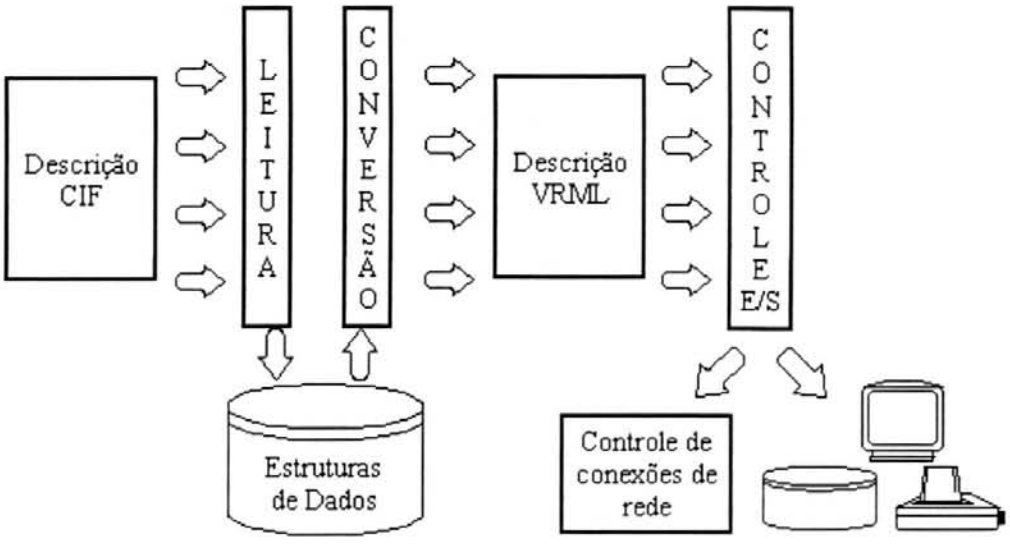

FIGURA 6.9 - Diagrama de funcionamento da conversão

O bloco de entrada e saída controla conexões de rede ao manipular o arquivo de descrição VRML, bem como a descrição CIF de entrada, nos casos de utilização de arquivos remotos no processo de conversão.

A conversão CIF para VRML é efetuada utilizando um browser WWW, que carrega pela rede o applet Java e o executa (Figura 6.10a). Esse applet, por sua vez, carrega a descrição CIF através da rede ou mesmo de uma unidade de disco local, passando-a para o bloco de conversão (Figura 6.10b).

A conversão será feita conforme visto anteriormente e a descrição VRML será passada ao bloco de controle de entrada e saída, que direcionará o arquivo conforme instruções do usuário. Caso volte para o bloco de controle de conexão de rede, este abrirá nova conexão para armazenar o arquivo em máquina remota (Figura 6.10c).

6.5.1.6 Implementação da Ferramenta

Para a primeira versão da ferramenta, chamada CIF2VRML [IND97a], implementou-se um applet Java e integrou-se a mesma ao ambiente de projeto conforme arquitetura vista na Seção 5.3.1. Essa decisão se deu para que os recursos de utilização de arquivos remotos possam ser utilizados, bem como para permitir o uso simplificado da ferramenta por estudantes e projetistas, pois a ferramenta é carregada e hospedada por um web browser na máquina cliente (Figura 6.11).

Os parâmetros de conversão 2D/3D são carregados de arquivos de configuração (locais ou remotos) com informações pertinentes à tecnologia [PAL89] usada na concepção do circuito a ser convertido, ou ainda completamente parametrizados pelo usuário através da interface gráfica da ferramenta.

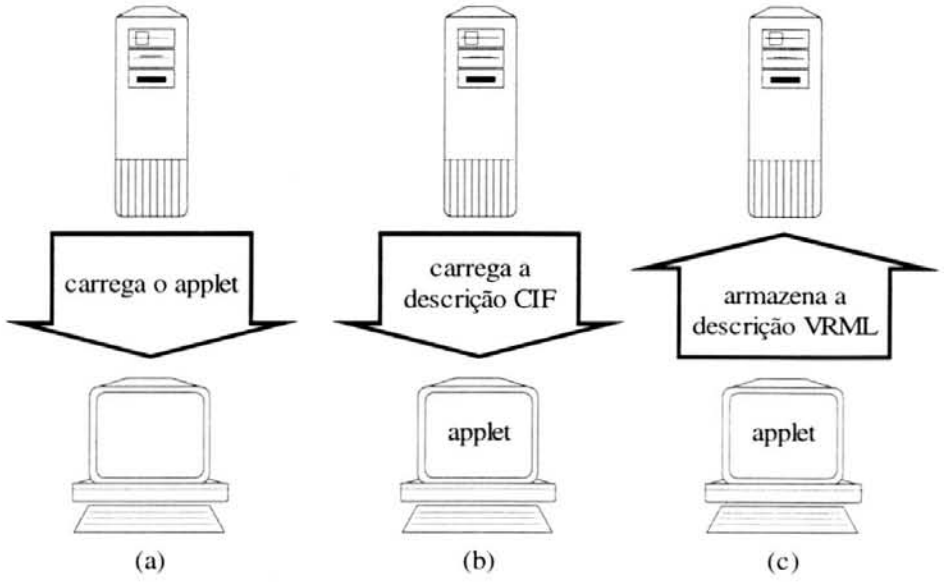

FIGURA 6.10 - Procedimentos do bloco de controle de conexão de rede

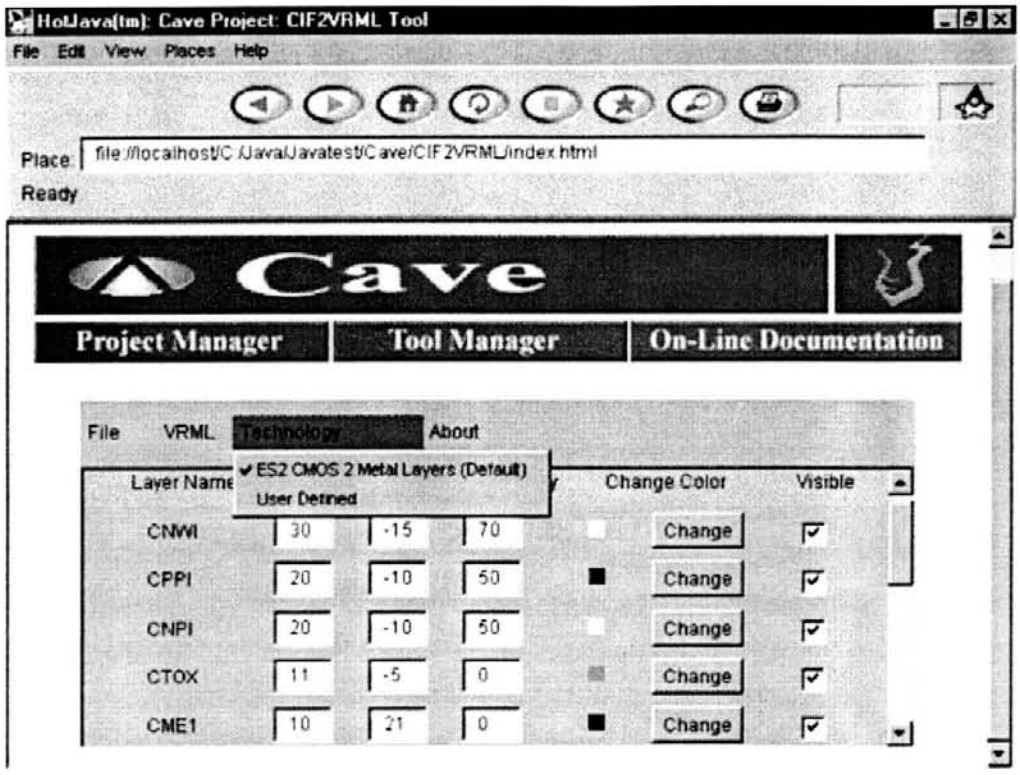

FIGURA 6.11 - Interface da ferramenta CIF2VRML

6.5.1.7 Avaliação do Processo de Visualização 3D de Circuitos Integrados usando VRML

Apesar de estar cercada das indefinições comuns em se tratando de um padrão recém definido, a linguagem VRML mostrou que à medida em que suas características forem sendo sedimentadas, várias possibilidades de aplicação se abrirão.

A utilização da linguagem VRML como descrição de layout de circuitos integrados mostrou-se bastante adequada, apresentando melhoria em recursos educacionais e de apoio ao projeto em relação às descrições 2D usuais como o formato CIF. Possibilidades de transparência entre camadas, mudança de pontos de vista, movimento do modelo 3D (Figura 6.12) são recursos que entre vários outros permitidos pela descrição VRML asseguram essas melhorias.

Entretanto, o uso do VRML como padrão estrutura de dados usada por programas de apoio a projeto seria completamente equivocado, pois a linguagem é bastante legível e compreensível a nível de usuário, se opondo aos formatos otimizados para a manipulação pelos programas. Isso mostra que não pode existir a intenção de substituição de um formato por outro, e sim a definição clara do uso de cada um.

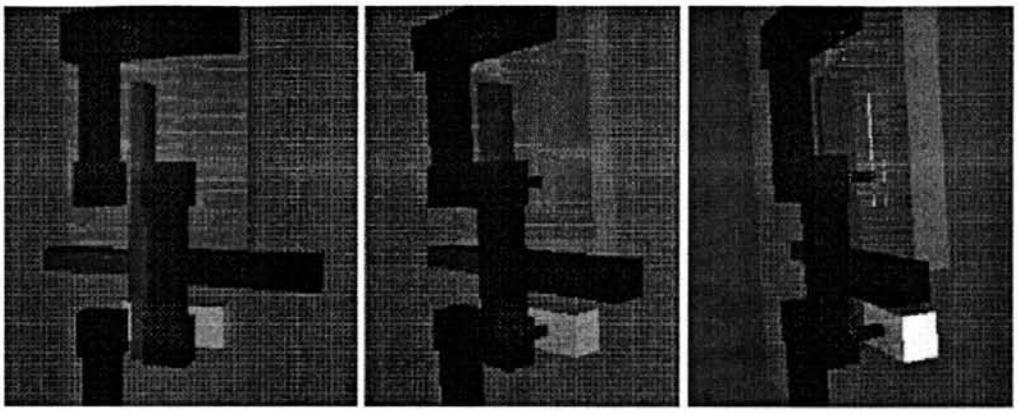

FIGURA 6.12 - Giro do modelo 3D de um bloco de circuito integrado

6.5.2 Ágata

O sistema Ágata [CAR96] é um ambiente para suporte ao projeto com circuitos pré-difundidos totalmente realizado no Brasil. Desenvolvido por pesquisadores do Grupo de Microeletrônica da UFRGS, o sistema de CAD viabiliza a realização de ASICs baseados na matriz de transistores pré-difundidos denominada GA2500, projetada no CTI, responsável pelo processo de personalização do circuito.

O fluxo de projeto do sistema Ágata parte de uma descrição estrutural com portas lógicas, disponíveis na biblioteca do sistema, com compatibilidade TTL. Esta descrição é utilizada tanto na simulação como na etapa de síntese física.

A integração do sistema Ágata ao protótipo do ambiente Cave se deu de forma parcial. Por se adequar ao Grupo 2 da arquitetura de integração de ferramentas, apenas a etapa de síntese física do circuito foi integrada ao ambiente, deixando a etapa de simulação como trabalho futuro.

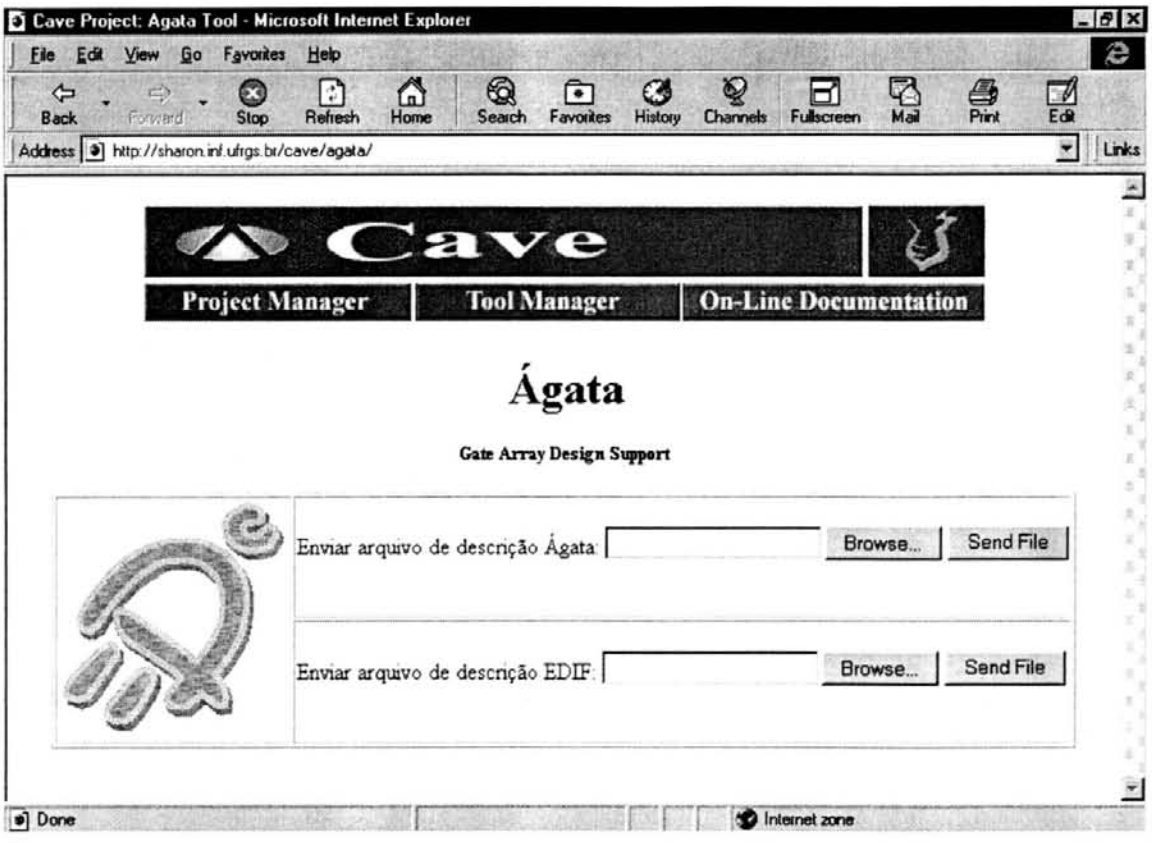

FIGURA 6.13 - Interface da ferramenta Ágata

A interface, conforme definido na arquitetura de integração de ferramentas do Grupo 2, é baseada em HTML Forms, que enviam o arquivo de descrição do circuito na linguagem do sistema ao servidor. No servidor, um servlet recebe o arquivo, dispara um arquivo batch com a seqüência de ferramentas que vão gerar o layout físico das conexões na matriz de transistores do GA2500: planificação de netlist, posicionamento relativo, posicionamento absoluto, roteamento simbólico e geração do layout. O uso de descrições hierárquicas com o uso de subcircuitos não foi suportado pela primeira versão da ferramenta no ambiente Cave.

Além da linguagem de entrada do sistema, a descrição do circuito também pode ser enviada na linguagem EDIF. Para isso, é feita a conversão para a linguagem do sistema com a ferramenta EDIF2H [KIN96].

Após todo o processo de geração de layout, o projetista recebe o arquivo CIF, que é necessário para a fabricação do circuito.

6.5.3 LayEd

A ferramenta LayEd foi a primeira ferramenta a ser desenvolvida dentro do programa de apoio a desenvolvedores de ferramentas para o ambiente Cave.

Consiste de um editor de layout de circuitos integrados completamente escrito em Java - conforme arquitetura do Grupo 1. As funções do LayEd são a edição e visualização dos arquivos de layout dentro do ambiente Cave. O editor pode acessar tanto arquivos locais quanto bibliotecas de arquivos em web servers, referenciando-os por URLs. Os primeiros testes de criação e edição de layout em uma interface WWW podem ser vistos na Figura 6.14. Nessa figura, pode-se notar também o uso das classes padronizadas para construção de menus, detalhadas na Seção 6.5.4.2.1.

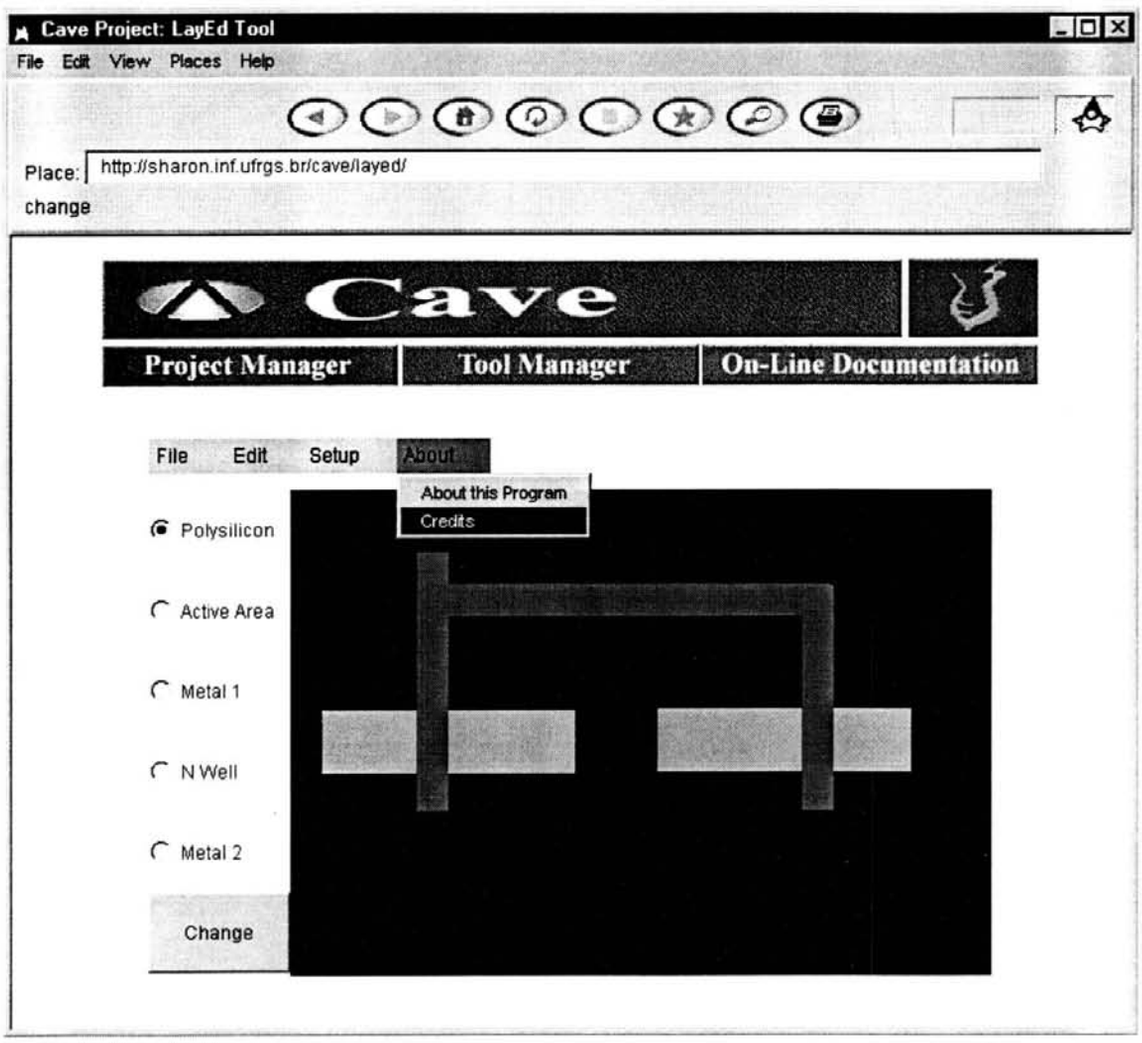

FIGURA 6.14 - Interface da ferramenta LayEd

6.5.4 Apoio a Desenvolvedores

A primeira versão do ambiente Cave não traz nenhuma ferramenta que automatize etapas do processo de implementação de ferramentas para o ambiente. Entretanto, alguns blocos de código utilizado na implementação das outras ferramentas, vistas nas seções anteriores, foram implementados genericamente, de forma a poderem ser reutilizados em outras ferramentas.

Esses blocos de software - que na verdade são classes Java - são repositórios de métodos estáticos (funções) ou geradores de objetos a serem instanciados e utilizados pelos desenvolvedores. O código fonte e a documentação dessas classes estão disponíveis no contexto Tool Manager do ambiente Cave. Nas seções seguintes, essas classes são detalhadas.

6.5.4.1 CIF Parser

CIF Parser é um conjunto de classes Java que fazem a leitura da descrição CIF [SHE93] textual de layout de um circuito integrado e armazena essa informação em estruturas legíveis por outras classes Java.

Gerado a partir de uma descrição da linguagem CIF pelo gerador de parsers JavaCC [SAN97], o CIF Parser analisa em cada arquivo lido a sintaxe e a semântica da descrição nele contido, comparando-as com aquela que o gerou. Com isso evita-se que descrições que não estejam íntegras sejam passadas a outros blocos de um programa que utilize o parser.

A descrição de entrada do JavaCC difere da descrição BNF (Backus-Naur Form) freqüentemente utilizada (Figura 6.15). Então criou-se a partir desta descrição uma versão de acordo com os requisitos do JavaCC (Anexo A-2), que tem sintaxe baseada na própria linguagem Java.

O CIF Parser é formado pelas classes CifParser.class, ASCII_CharStream.class, CifParserConstants.class, CifParserTokenManager.class, Token.class e a classe que representa erro na leitura ParseError.class. Ao ler uma descrição CIF, o CIF Parser armazena seqüencialmente todos os comandos lidos na forma de instâncias de classes Java. Foram criadas classes para os principais comandos CIF: CIFBox.class, CIFCall.class, CIFDefEnd.class, CIFDefStart.class e CIFTransformation.class.

Todas essas classes estão reunidas em um package Java denominado cave.cif. A documentação para o uso dessas classes foi gerada pela ferramenta javadoc e encontra-se disponível no servidor de ambiente.

Alphabet	Rules
cifFile	= {{blank}[command]semi} endCommand{blank}
command	= primCommmand | defDeleteCommand | defStart Command semi{{blank}[primCommand]semi}defFinishCommand.
primCommaad	= polygonCommand | boxCommand | roundFlashCommand | wireCommand | layerCommand | callCommand | userExtensionCommand | commentCommand.
polygonCommand	= "P" path
boxCommand	= "B" integer sep integer sep point [sep point]
roundFlashCommand	= "R" integer sep point
wireCommand	= "W" integer sep path
layerCommand	= "L" {blank} shortname
defStartCommand	= "D" {blank} "S" integer [sep integer sep integer]
defFinishCommand	= "D" {blank} "F"
defDeleteCommand	= "D" {blank} "D" integer
callCommand	= "C" integer transformation
userExtensionCommand	= digit userText
commentCommand	= "("comment Text")"
endCommand	= "E"
transformation	= {{blank} ("T" point | "M" {blank}"X"|"M" {blank} "Y" | "R" point)}
path	= point {sep point}
point	= sInteger sep sInteger
sInteger	= {sep}["-"] integerD
integer	= {sep} integerD
integerD	= digit {digit}
shortname	= c[c][c][c]
c	= digit | upperChar
userText	= {userChar}
commentText	= {commentChar} | commentText "("commentText")"commentText
semi	= {blank} ";"{blank}
sep	= upperChar | blank
digit	= "0"| "1"| "2"| "3"| "4"| "5"| "6"| "7"| "8"| "9"
upperChar	= "A"| "B"| "C"| "D"| "E"| "F"| "G"| "H"| "I"| "J"| "K"| "L"| "M"| "N"| "O"| "P"| "Q"| "R"| "S"| "T"| "U"| "V"| "W"| "X"| "Y"| "Z"
blank	= any ASCII character except digit, upperChar, "-", "(", ")", or";"
userChar	= any ASCII character except ";"
commentChar	= any ASCII character except "(" or ")"

FIGURA 6.15 - Representação BNF da linguagem CIF

6.5.4.2 Elementos de Interface Gráfica

Uma série de classes foi construída sobre as bases do Abstract Window Toolkit para suportar o desenvolvimento da interface gráfica das primeiras ferramentas do ambiente de projeto. Essas classes foram disponibilizadas para outros desenvolvedores. Agrupadas no package cave.awt, essas classes proporcionam uma padronização de interface das ferramentas do ambiente, mantendo a mesma identidade gráfica.

A documentação dessas classes também foi gerada pela ferramenta javadoc, estando disponível no servidor de ambiente.

6.5.4.2.1 AppletMenuBar e AppletTopMenuItem

Essas classes permitem a inclusão de menus na interface de applets - funcionalidade não permitida pelo AWT básico. A classe AppletTopMenuItem representa os contextos de menu (File, Edit, Help), e o conjunto desses contextos é agrupado pela classe AppletMenuBar. O menu propriamente dito é materializado pela classe PopupMenu, disponível no AWT básico.

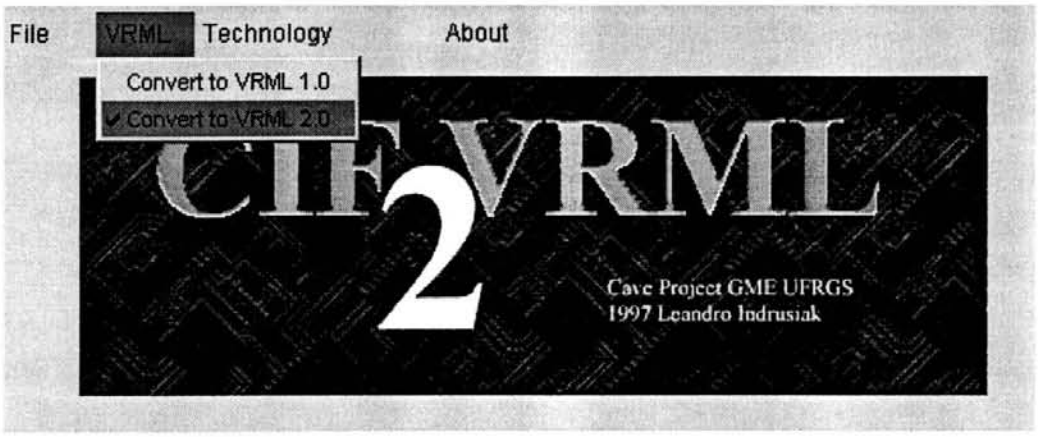

FIGURA 6.16 - Uso de AppletMenuBar e AppletTopMenuItem

6.5.4.2.2 ErrorWindow e EnterDialog

A classe ErrorWindow pode ser instanciada para se obter janelas para informar erros ao usuário. A classe EnterDialog pode ser instanciada para obter uma janela que bloqueie a interface da aplicação que a chamou, permitindo que o usuário volte a interagir com a aplicação apenas quando entrar com os dados exigidos por ela.

6.5.4.2.2 ColorSelector

A classe ColorSelector pode ser utilizada por aplicações que permitam que o usuário escolha cores. É materializada na forma de uma janela com várias opções de cores e com a possibilidade de entrada de parâmetros RGB para obter cores além das mostradas.

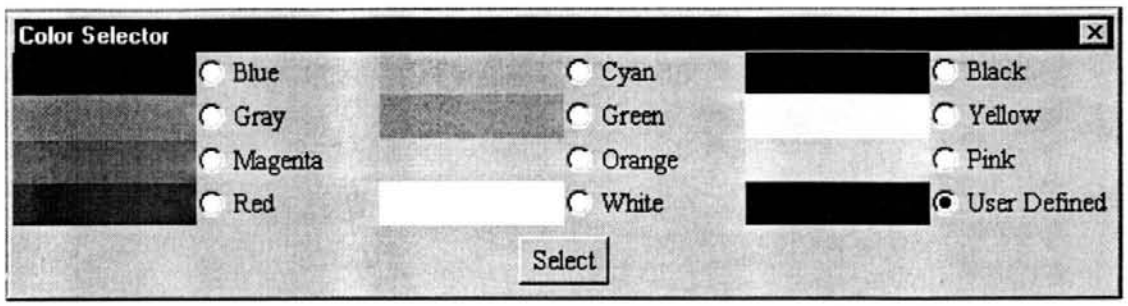

FIGURA 6.17 - Uso de ColorSelector

6.6 Cliente

Conforme definido no capítulo 5, o único software necessário nas máquinas clientes é o web browser. Para a implementação do protótipo do ambiente, uma escolha entre os web browsers disponíveis teve de ser feita. Os principais candidatos, detalhados a seguir, passaram por uma pré-seleção onde foram avaliadas as mais recentes versões, levando em conta disponibilidade, suporte a Java e às mais recentes extensões HTML e versões para as principais plataformas de software e hardware.

Os browsers Netscape Navigator 3.0, Netscape Communicator e Microsoft Internet Explorer 4.0 são considerados adequados nos quesitos de disponibilidade, suporte às novas extensões do HTML e à linguagem Java. O ponto negativo é o suporte incompleto à linguagem Java 1.1, bem como a impossibilidade da parametrização do modelo de segurança para applets.

O único browser disponível no momento da implementação do protótipo que permite ao usuário controle completo das restrições de segurança a applets é o HotJava, da Sun Microsystems (Figura 6.18). Apesar de ter uma performance mais limitada e não suportar todos os recursos da linguagem HTML, a parametrização pelo usuário do modelo de segurança fez com que ele tenha sido a aplicação escolhida para as máquinas cliente do ambiente de projeto. É importante ressaltar que qualquer um dos outros web browsers poderia ser usado, usando a parametrização de segurança através de applets assinados [WUT97]. Como trabalho futuro, pretende-se aplicar os métodos de assinatura nas ferramentas do ambiente, permitindo que tenham acesso pleno aos recursos da máquina cliente qualquer que seja o web browser.

Escrito totalmente em Java, o browser HotJava tem a interface de configuração toda baseada em applets anexos a documentos HTML, o que mantém a convenção definida para todo o ambiente de projeto. Para versões futuras do protótipo, pode-se parametrizar os documentos HTML de configuração do web browser conforme os padrões do nível 2 da interface do ambiente, substituindo a identidade gráfica do fabricante, como mostrado na Figura 6.18.

Para os aspectos de segurança no cliente também foi analisado o Personal Web Server, que faz parte do pacote Microsoft Windows. O PWS permite que uma máquina cliente também seja servidora, rodando um software servidor bastante simples. Isso garante a privacidade dos arquivos armazenados no cliente, pois pode-se limitar o acesso de applets e aplicações externas apenas dentro do domínio do servidor ao permitir o acesso de disco local apenas por intermédio desse software.

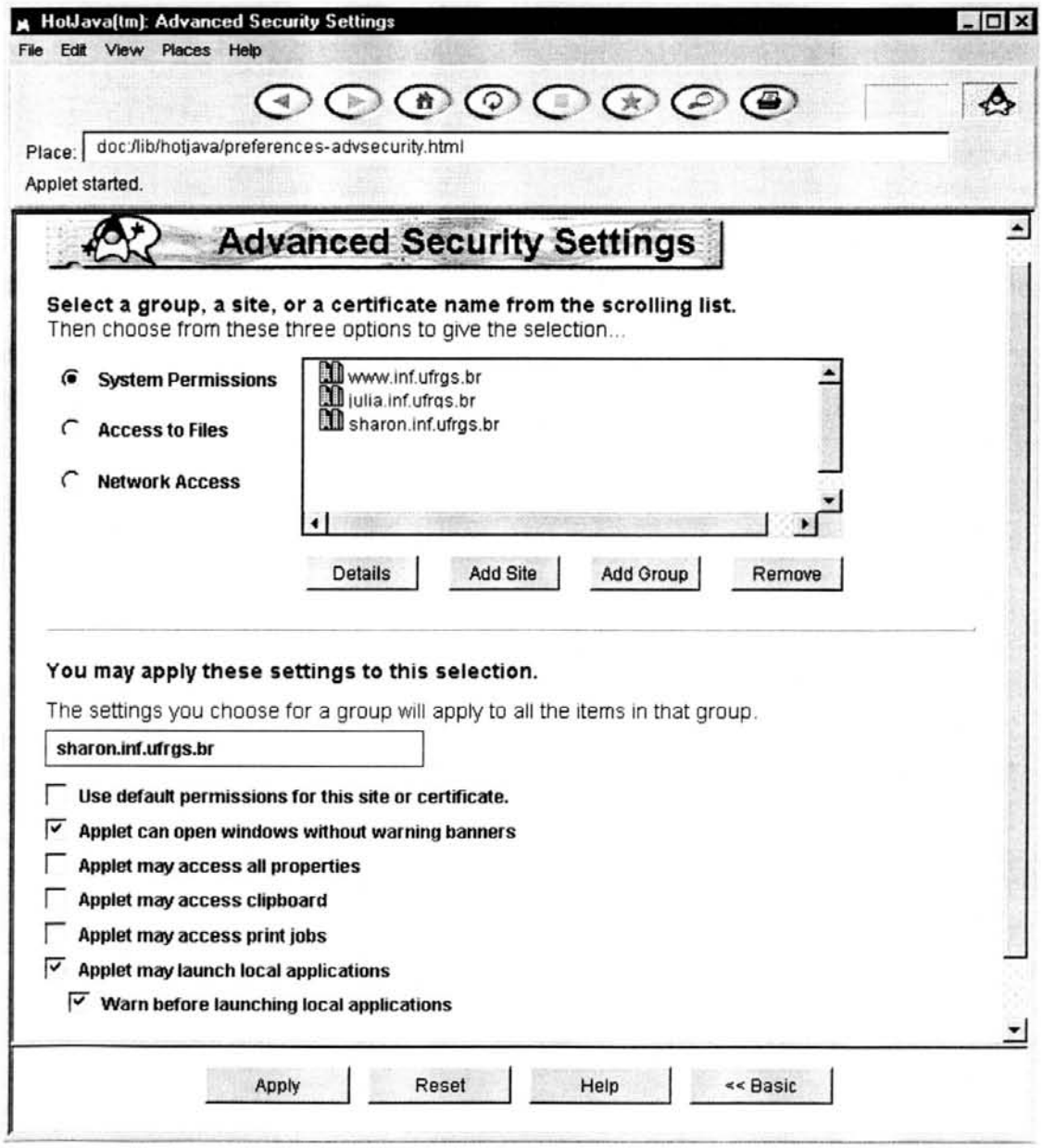

FIGURA 6.18 - Parametrização do modelo de segurança no HotJava Browser

7 Conclusão

7.1 Contribuição do Trabalho

Após a revisão bibliográfica, definição de uma arquitetura e implementação de um protótipo, chegou-se a um ambiente com as seguintes características:

Interface padronizada: as interfaces para navegação através do *framework* e a interface das ferramentas são baseadas em hiperdocumentos padronizados escritos em HTML. Com isso, o tempo necessário para a adaptação do projetista ao ambiente é reduzido.

Multiplataforma: sendo baseado no World Wide Web, todo o *framework* pode ser acessado pelos projetistas através de web browsers. Esses browsers estão disponíveis comercialmente e fazem a interface entre o ambiente e as principais plataformas de hardware e software. Assim, praticamente todo projetista que tenha uma conexão à rede Internet pode acessar o ambiente de projeto usando o web browser de sua preferência. A arquitetura proposta para a integração de ferramentas usando Java e CGI - padrões multiplataforma - também ajuda a manter a independência de plataforma do ambiente.

Distribuição de recursos transparente ao usuário: a interface criada pelos hiperdocumentos permite ocultar a localização geográfica dos recursos do ambiente distribuídos entre as máquinas da rede. Isso permite ao administrador distribuir a carga do processamento entre as máquinas da rede. O projetista, entretanto, não precisa administrar essa distribuição e possivelmente não a notará, uma vez que terá uma resposta adequada e interagirá em interfaces padronizadas qualquer que seja a ferramenta.

Atualização simples: a simplicidade da linguagem HTML permite uma atualização fácil do ambiente de projeto. Para anexar uma nova ferramenta ao ambiente, basta criar um novo hiperdocumento que dê suporte a ela e a ligue à estrutura de hipermídia previamente existente.

Trabalho remoto: um *framework* baseado no World Wide Web pode permitir acesso aos recursos de projetistas conectados à Internet, independentemente de suas localizações. Isso permite o trabalho colaborativo de equipes de projeto dispersas em uma grande área, como por exemplo o projeto conjunto entre universidades e/ou centros de pesquisa de empresas.

Requisitos simples de hardware e software: o único requisito do sistema é uma máquina conectada à rede Internet/Intranet rodando um

web browser que implemente o Java Runtime Environment. Recentes pesquisas prevêem que uma nova geração de computadores mais simples podem vir a ocupar parte significativa do mercado de computadores pessoais - os network computers [WAY96]. Esse conceito encaixa-se nos requisitos mínimos do proposto *framework*. Isso quer dizer que no caso de se concretizarem as previsões, o *framework* proposto poderá ser uma aplicação típica para os network computers.

Integração com hiperdocumentos educacionais: O *framework* pode ser apoiado por hiperdocumentos educacionais - help on-line, tutoriais, cursos - porque o usuário pode alternar de forma simples entre o ambiente de projeto e o ambiente de instrução, uma vez que ambos compartilham a mesma mídia: são baseados em hiperdocumentos. Além de toda a documentação armazenada no servidor de ambiente, pode-se ligar informação adicional sobre cada tópico abordado através de links de hipertexto.

Muitas dessas características ainda podem ser exploradas para que aumentem ainda mais a eficiência do apoio ao projetista de circuitos integrados. Muito trabalho ainda precisa ser pesquisado e executado, por isso deve-se considerar o presente texto como um ponto de partida.

A Figura 7.1 mostra a abrangência do presente trabalho dentre os elementos que formam um framework segundo [BAR92].

7.2 Trabalho Futuro

A implementação de ferramentas a serem integradas no ambiente é ponto essencial e já está sendo executada dentro do GME UFRGS, como descrito na Seção 6.5. Além das ferramentas de apoio ao projeto, o *framework* é carente de recursos que apoiem o trabalho do administrador do ambiente e do desenvolvedor de ferramentas.

Ferramentas de automatização da integração de novas ferramentas ao *framework*, controle de acesso de projetistas, comunicação entre usuários são alguns exemplos de facilidades a serem implementadas para o administrador do *framework*.

A implementação de um maior número de classes a serem reutilizadas pelos desenvolvedores será conseqüência da implementação de um maior número de ferramentas, mas o apoio a desenvolvedores pode ser ainda maior com a criação de recursos no *framework* que auxiliem na criação e padronização dessas ferramentas.

Ferramentas de modelagem de fluxo de projeto também devem ser estudadas e implementadas de acordo com o novo paradigma da integração de ferramentas baseada em hiperdocumentos. Nesse caso, o fluxo de projeto pode ser modelado através de uma rede de links de hipertexto, ligando uma etapa (ferramenta) às etapas subseqüente e precedente.

A integração do ambiente a um sistema de banco de dados também deve ser estudada. Uma vez que o sistema é essencialmente distribuído, com ferramentas e dados trafegando entre o servidor do ambiente e as máquinas cliente, o gerenciamento de versões é ponto crítico. O armazenamento, para o caso de aplicar os conceitos pesquisados a um *framework* comercialmente viável, também deve ser mais cuidadoso, principalmente no caso de arquivos de grande porte, freqüentemente encontrados em projetos de circuitos integrados. No sistema atual, os arquivos são armazenados no sistema de arquivos do servidor de ambiente, não havendo nenhum controle de persistência dos dados.

Algumas tarefas ainda devem ser executadas para tornar o uso da rede Internet mais racional por parte das ferramentas do *framework*. Um sistema automático de compactação e criptografia a ser usado sempre que dados ou ferramentas forem trafegar pela rede aumentariam sensivelmente a performance e a confiabilidade do sistema. No que tange a segurança, é necessário o estudo de métodos de assinatura das ferramentas do Grupo 1 - applets executados nas máquinas cliente - para garantir o acesso dos applets aos recursos do sistema operacional da máquina na qual são executados, passando sobre as restrições impostas aos applets de origem desconhecida.

Para que toda essa funcionalidade seja adicionada de forma organizada, mudanças na arquitetura do ambiente devem ser estudadas. A inclusão de um nível de serviços de rede [CHA97] entre cliente e servidor é uma opção que deve ser abordada, evoluindo a arquitetura proposta para um modelo em três níveis explicitamente divididos (*three-tier*). Esse nível deverá acompanhar e auxiliar as tarefas em execução nos clientes e servidores do ambiente e ainda prover facilidades de comunicação entre eles.

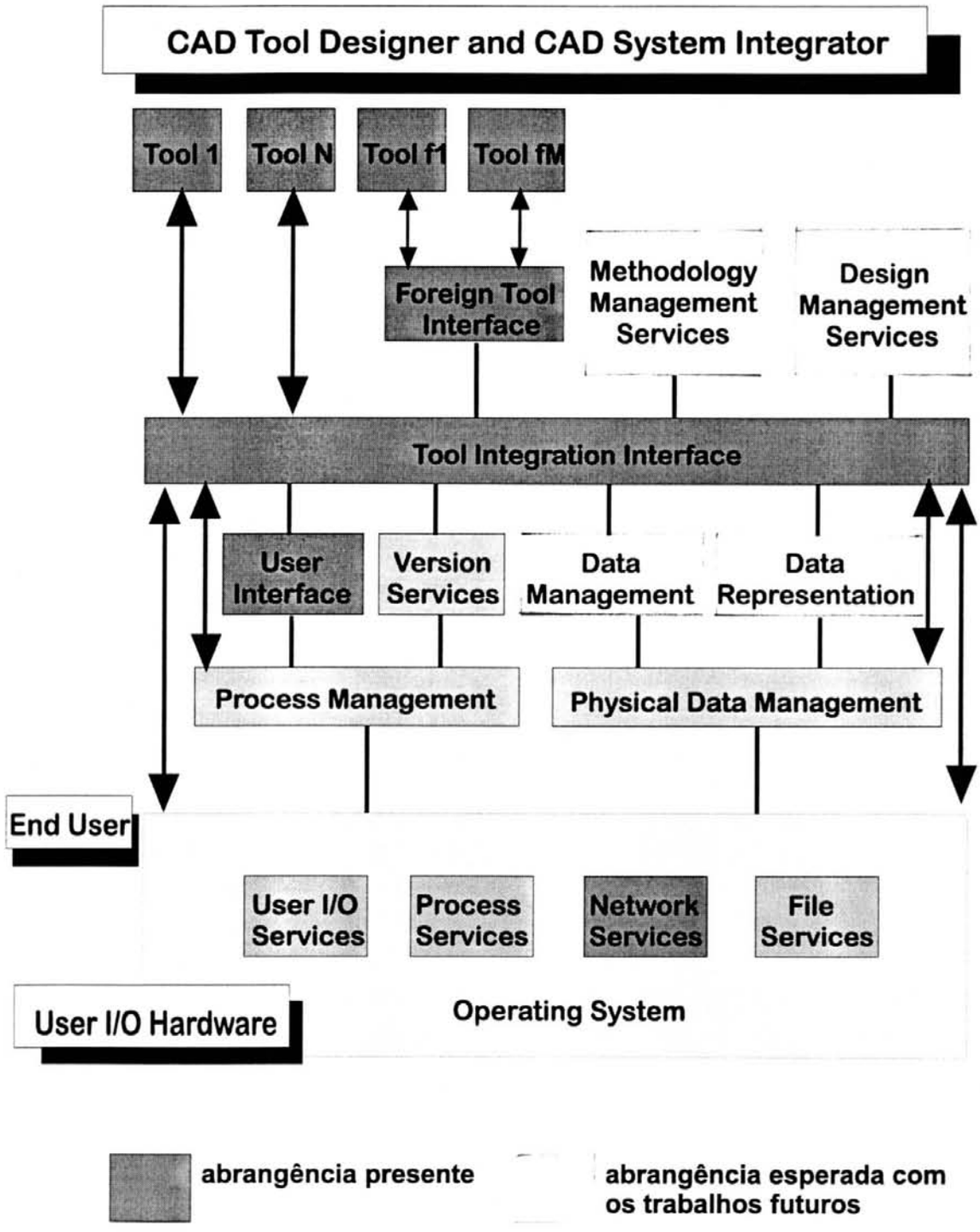

FIGURA 7.1 - Abrangência do trabalho

Anexo 1 Linguagem Java

A-1.1 Conceito

Java é uma linguagem computacional completa, adequada para o desenvolvimento de aplicações baseadas na rede Internet, redes fechadas ou ainda programas stand-alone [CAM96].

Foi desenvolvida na 1a metade da década de 90 nos laboratórios da Sun Microsystems com o objetivo de ser mais simples e eficiente do que suas predecessoras. O alvo inicial era a produção de software para produtos eletrônicos de consumo (fornos de microondas, agendas eletrônicas, etc.). Um dos requisitos para esse tipo de software é ter código compacto e de arquitetura neutra.

A linguagem obteve sucesso em cumprir os requisitos de sua especificação, mas apesar de sua eficiência não conseguiu sucesso comercial. Com a popularização da rede Internet, os pesquisadores da Sun Microsystems perceberam que aquele seria um nicho ideal para aplicar a recém criada linguagem de programação. A partir disso, adaptaram o código Java para que pudesse ser utilizado em microcomputadores conectados a rede Internet, mais especificamente no ambiente da World Wide Web. Java permitiu a criação de programas batizados applets, que trafegam e trocam dados através da Internet e se utilizam da interface gráfica de um web browser. Implementaram também o primeiro browser compatível com a linguagem, o HotJava, que fazia a interface entre as aplicações Java e o sistema operacional dos computadores.

Com isso, a linguagem conseguiu uma popularização fora de série, passando a ser usada amplamente na construção de documentos web que permitam maior interatividade. Os principais web browsers disponíveis comercialmente passaram a dar suporte aos programas Java, e outras tecnologias em áreas como computação gráfica e banco de dados também buscaram integrar-se com o novo paradigma proposto pela linguagem: aplicações voltadas para o uso de redes de computadores.

Atualmente, a linguagem Java é a força propulsora por trás de alguns dos maiores avanços da computação mundial, como:

- Acesso remoto a bancos de dados
- Bancos de dados distribuídos
- Comércio eletrônico no WWW
- Network CAD

- Interatividade em páginas WWW
- Interatividade em ambientes de Realidade Virtual distribuídos
- Gerência de Documentos
- Integração entre dados e forma de visualização
- Network Computer
- Ensino à distância
-Jogos e entretenimento

A-1.2 Características

Java é uma linguagem poderosa em ambientes distribuídos complexos como a rede Internet. Mas sua versatilidade permite ao programador ir além, oferecendo uma poderosa linguagem de programação de uso geral, com recursos suficientes para a construção de uma variedade de aplicativos que podem ou não depender do uso de recursos de conectividade [WUT97] .

As principais características da linguagem foram divulgadas pela primeira vez em The Java Language: A White Paper em 1995 [GOS95]. A seguir, fazemos um detalhamento geral da linguagem a partir das características presentes nesse documento.

A-1.2.1 Simplicidade e eficiência de código orientado a objetos

Java é uma linguagem simples, de fácil aprendizado ou migração, pois possui um reduzido número de construções. A diminuição das construções mais suscetíveis a erros de programação, tais como ponteiros e gerenciamento de memória via código de programação também faz com que a programação em Java seja mais eficiente. Contém um conjunto de bibliotecas que fornecem grande parte da funcionalidade básica da linguagem, incluindo rotinas de acesso à rede e criação de interface gráfica.

Baseada no paradigma da Orientação a Objetos - encapsulamento em um bloco de software dos dados (variáveis) e métodos de manipulação desses dados - a linguagem permite a modularização das aplicações, reuso e manutenção simples do código já implementado.

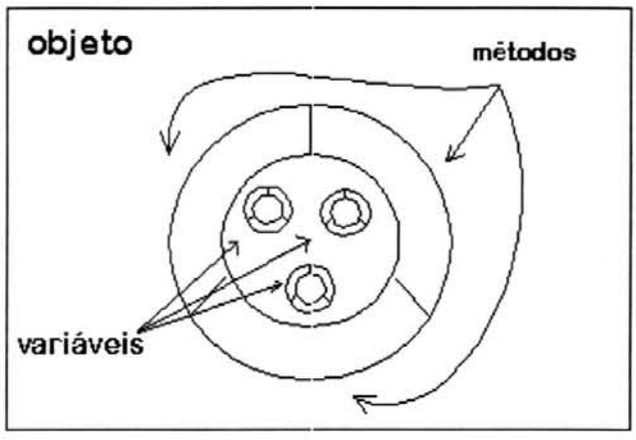

FIGURA A-1.1 - Encapsulamento de variáveis e métodos em um objeto

A-1.2.2 Código Interpretado e Portável

As diversas linguagens de programação podem ser tanto compiladas como interpretadas. Quando se utiliza uma linguagem compilada, é necessário executar um programa para traduzir os arquivos fonte, legíveis em linguagem de alto nível, em código executável. As linguagens compiladas têm a vantagem de produzir código de alta performance, o qual está ajustado para o funcionamento em um tipo específico de processador ou arquitetura de processador. Aplicativos compilados, chamados de código binário, só podem rodar no tipo de computador para o qual foram compilados, uma vez que esses aplicativos consistem, na realidade, em instruções em linguagem de máquina, entendidas e executadas pelo microprocessador.

As linguagens interpretadas só existem em código fonte. Quando em execução, um programa chamado interpretador toma o código fonte e executa as ações indicadas pelos comandos no arquivo. O interpretador é, na realidade, o único aplicativo que está sendo executado. Entre os benefícios das linguagens interpretadas está o fato dos programas interpretados poderem rodar em uma variedade de plataformas diferentes, pois estes só existem em código fonte. Além disso são mais fáceis de depurar.

A linguagem Java é tanto compilada como interpretada. Após escrever um programa em Java, utilizando um editor de textos qualquer, salva-se o programa como código fonte. A seguir, pode-se compilar esse fonte, a fim de produzir um tipo de arquivo binário chamado de arquivo de classe. Esses arquivos não são executados diretamente pois eles não contêm instruções que são entendidas diretamente pelos processadores atualmente disponíveis no mercado. Os programas Java são compilados em um formato intermediário chamado bytecodes. Assim, esses programas podem ser executados em qualquer sistema através de um interpretador Java (runtime environment). Com isso, o código precisa ser escrito e compilado apenas uma vez, pois os bytecodes gerados serão executados da mesma forma em qualquer plataforma de hardware e software.

A Sun Microsystems atualmente está trabalhando em compiladores específicos para uma plataforma de hardware. Se o hardware onde o código Java será executado é conhecido, pode-se melhorar a performance do aplicativo. Em contrapartida, não será mais possível rodar esse código em outras plataformas.

Além disso, processadores que executem os bytecodes Java sem a necessidade de um interpretador também estão sendo colocados no mercado pela Sun.

A-1.2.3 Segurança

Por ter seu projeto voltado para a simplicidade de código, as possibilidades de erro de programação em Java são reduzidas. Apesar disso, a linguagem traz outros recursos para tornar seu código ainda mais eficiente. O processo de compilação - geração de bytecodes - é projetado para a detecção prévia dos possíveis erros, evitando que os erros se manifestem em tempo de execução. O uso de código para tratamento de exceções · exception handling - permite manter a consistência da aplicação no caso de erros.

Além de diminuir as possibilidades de erro de programação, a linguagem tem um esquema de segurança para garantir a integridade de código - principalmente no caso do código originário de rede insegura.

Esses recursos de segurança são notáveis principalmente dentro do ambiente do interpretador [GON97]. Após baixar um applet da rede, o interpretador faz uma verificação do código, buscando alterações intencionais ou não. A seguir, o interpretador determina o layout de memória para execução. Em outras palavras, não é possível acessar informações diretamente da memória ou inserir código estranho ao código original. Além disso, um programa em Java não pode acessar o sistema de arquivos, salvo nos casos previstos pelo cliente: diferente conjunto de permissões de acordo com a origem das aplicações [MCG97].

A-1.2.4 Aplicações distribuídas e processamento paralelo

Com a larga utilização dos recursos da rede Internet, a linguagem teve seu projeto voltado para as aplicações em rede. Assim, a linguagem traz classes para o suporte a vários níveis de conectividade: acesso a URLs (padrão Internet), uso de conexões em sockets, criação de protocolos, criação de clientes e servidores. A introdução desses conceitos se deu de forma simples para o programador, pois permite a ele o acesso às informações da rede com a mesma facilidade do acesso aos arquivos locais.

Para permitir uma melhor performance de execução, mesmo em tarefas de maior complexidade, a linguagem permite a programação de threads - processos de execução de tarefas que ocorrem simultaneamente. A linguagem traz também mecanismos para sincronização, ativação e desativação parametrizada desses processos.

A-1.3 Recursos para desenvolvimento em Java

Ao lançar a primeira versão pública da linguagem, a Sun Microsystems preocupou-se em fornecer aos futuros desenvolvedores de aplicações Java um pacote de ferramentas e bibliotecas básicas. Esse pacote, chamado Java Development Kit [KRA97], é indispensável para o desenvolvedor iniciante. Posteriormente, quando estiver mais familiarizado com a linguagem, o desenvolvedor pode migrar para algum outro ambiente de desenvolvimento mais sofisticado (existem vários disponíveis comercialmente), já que o JDK não tem uma interface muito amigável.

A-1.3.1 JDK Tools

O JDK oferece ao desenvolvedor uma série de ferramentas, como o interpretador, o compilador, o debugger, o gerador de documentação, o visualizador de applets, o visualizador de bytecodes, o compactador de classes, o gerador de assinaturas digitais, entre outros. As ferramentas mais freqüentemente utilizadas são:

A-1.3.1.1 javac

javac é o compilador da linguagem Java. Ele lê arquivos fonte .java e gera arquivos de classe .class no formato de bytecodes. Para cada classe especificada (pode-se especificar mais de uma classe em cada arquivo fonte) é gerado um arquivo de classe chamado NomedaClasse.class. Importante: o arquivo fonte deve ter o nome da classe que ele define seguido da extensão .java, caso contrário o compilador acusa erro.

A-1.3.1.2 java

O interpretador java é utilizado para executar aplicações em Java. Ele interpreta os bytecodes gerados pelo javac. O interpretador é executado a partir de linha de comando da forma:

java NomedaClasse arg1 arg2 arg3...

Ao ser executado, o interpretador busca o arquivo NomedaClasse.class, nele procura um método public static void main() e a ele passa os argumentos que recebeu (ou não, pois os argumentos são opcionais e sua existência depende da funcionalidade da classe que está sendo interpretada).

A-1.3.1.3 appletviewer

O appletviewer é uma ferramenta para visualização de applets fora do ambiente web browser. Executado a partir de linha de comando, recebe como argumento o arquivo HTML ao qual está anexo o applet Java. Esta ferramenta é bastante útil no processo de desenvolvimento e teste de applets.

A-1.3.1.4 javadoc

O javadoc é um gerador automático de documentação. Ele lê declarações de classes - código fonte - e busca por comentários especiais (iniciam com /** e terminam com */), gerando páginas HTML com as informações contidas nesses comentários, bem como informações sobre os métodos, variáveis e herança da classe declarada.

A-1.3.2 Java API

O conjunto de bibliotecas inclusas no JDK é conhecido como Java API (Interface de Programação de Aplicações). Nessas bibliotecas estão um conjunto grande de classes, organizadas em pacotes. Cada um desses pacotes traz classes com funcionalidade básica e vital para um determinado ramo de programação Java. A seguir, uma lista dos pacotes disponíveis na versão 1.1 do JDK. Os pacotes não marcados com + também estão disponíveis na versão 1.0.

Tabela A-1.1 - Lista de pacotes da linguagem Java

java.applet	+ java.lang.reflect	+ java.security
+ java.awt.datatransfer	+ java.math	+ java.security.interfaces
+ java.awt.event	java.net	+ java.sql
java.awt	+ java.rmi.dgc	+ java.text
java.awt.image	+ java.rmi	java.util
+ java.beans	+ java.rmi.registry	+ java.util.zip
java.io	+ java.rmi.server	
java.lang	+ java.security.acl	

Os pacotes de uso mais comum em aplicações básicas são:

A-1.3.2.1 java.applet

Contém as classes necessárias para o desenvolvimento de applets. Basicamente, todos os métodos para a criação de um aplicativo a ser rodado dentro de um web browser são encontrados dentro de uma única classe do package: a classe Applet.

A-1.3.2.2 java.awt

Contém classes relacionadas à interface gráfica [ZUC97], como botões, caixas de texto, menus, etc. Contém ainda classes para processamento de imagens e as classes que são empregadas no tratamento dos eventos gerados pela interface gráfica (clique de mouse sobre um botão, seleção de um item de um menu, etc.), no pacote java.awt.event.

A-1.3.2.3 java.io

Classes para entrada e saída de dados das mais variadas formas. Torna transparente ao usuário as características do sistema de arquivos da plataforma na qual o programa está rodando.

A-1.3.2.4 java.lang

São as classes que dão suporte ao modelo computacional da linguagem. São indispensáveis para o funcionamento de qualquer programa Java.

A-1.3.2.4 java.net

Contém classes aptas a estabelecer conexões de rede. Possuem métodos específicos para a rede Internet, usando referências a URLs e conexões usando TCP/IP.

A-1.3.2.5 java.util

Contém uma série diversa de classes de apoio ao programador, como estruturas de dados básicas (hash tables, pilhas, etc.), referência à data do sistema, gerador de números randômicos.

A-1.3 Programas Java - Applets e Applications

Os programas escritos em Java podem assumir duas formas básicas: applets e applications. A programação desses dois tipos é baseada nos mesmos conceitos da linguagem, mas têm características bem distintas. Segurança, interface gráfica, acesso à unidades de disco e acesso à rede são pontos de divergência entre applets e applications.

Entretanto, a diferença básica está no fato de que applets precisam de um web browser para existir. Isso quer dizer que applets são aplicações voltadas para o ambiente Internet/Intranet e que são transportadas pela rede junto com hiperdocumentos HTML. Já as applications são programas escritos para operar sem a necessidade de um web browser - aplicações convencionais para computadores, como editores de texto, gerenciadores de arquivos, etc.

A seguir, temos exemplos de applets e applications, bem como as características de cada um.

A-1.3.1 Applications

Applications são aplicativos stand-alone escritos em Java. São, na realidade, classes independentes que o interpretador reconhece e executa. O método principal é nomeado como main(). Ao reconhecer uma application, o interpretador chama o método main(), que deve iniciar o funcionamento de toda a aplicação.

Principais características das applications:

- não necessitam de um web browser para montar sua interface gráfica, ou seja, precisam criar janelas para montar a interface gráfica, ou ainda operar em linha de comando
- não têm restrições de acesso a unidades de disco
- não têm restrição de acesso à rede

A seguir na Figura A-1.2, temos um exemplo simples do código de um application. Esse código deve ser armazenado em um arquivo de mesmo nome da classe que ele define, seguido da extensão **.java**.

arquivo MeuApplication.java

```
public class MeuApplication {
        public static void main (String[] args) {
                System.out.println( "Este é meu application!" );
        }
}
```

FIGURA A-1.2 - classe MeuApplication

A-1.3.2 Applets

Um applet é um tipo específico de aplicativo que é dependente de um navegador web. Em vez de ter um método main(), um applet implementa um conjunto de métodos que lidam com situações tais como inicialização, quando e como desenhar a tela, o que fazer quando ocorre um clique de mouse e assim por diante. Os navegadores habilitados para Java se beneficiam do fato dessa linguagem ser dinâmica, colocando applets ligados a páginas, carregando-os automaticamente quando essas páginas forem carregadas. O applet passa a fazer parte do navegador quando ocorre a sua execução.

Principais características dos applets:

- necessitam de um web browser para montar sua interface gráfica
- tem restrições de acesso a unidades de disco - só acessa com a permissão explícita do usuário (alguns web browsers não permitem o acesso a disco em nenhuma circunstância)
- tem restrição de acesso à rede - podem acessar apenas o site do qual vieram (alguns web browsers permitem ao usuário definir permissões para que o applet acesse outros sites)
- utilizam os métodos init( ), start( ), stop( ) e destroy( ) para definir seu ciclo de vida:

  init( ) - método chamado imediatamente após a criação do applet. É chamado pelo web browser na primeira vez que o applet é carregado.

  start( ) - método chamado pelo web browser toda a vez que o applet é materializado na tela.

  stop( ) - método chamado pelo web browser sempre que o applet deixa de ser visível

destroy( ) - método invocado quando todos os recursos computacionais alocados pelo applet precisam ser liberados.

A seguir, temos um exemplo simples do código de um applet. Esse código deve ser armazenado em um arquivo de mesmo nome da classe que ele define, seguido da extensão **.java**.

arquivo MeuApplet.java

```
import java.applet.*;
import java.awt.*;

public class MeuApplet extends Applet {
        public void paint (Graphics g) {
                g.drawString( "Este é meu applet!" );
        }
}
```

FIGURA A-1.3 - classe MeuApplet

Após o processo de compilação do arquivo MeuApplet.java, que pode ser efetuado pelo compilador **javac** encontrado no JDK, é necessário criar um arquivo HTML que contenha a chamada para o applet. Esse arquivo HTML é que será chamado pelo web browser. O exemplo a seguir mostra o arquivo HTML com a chamada ao applet:

arquivo MinhaApplet.html

```
<HTML>
<applet code="MeuApplet.class" width="200" height="100">
</applet>
</HTML>
```

FIGURA A-1.4 - Código HTML para inclusão de um applet em documento WWW

Os applets são carregados e formatados dentro de uma página web de forma semelhante a uma imagem. Na maioria dos browsers, quando o arquivo HTML indica que uma imagem deve ser colocada na página, a imagem é carregada a partir do servidor web e exibida no local apropriado: a imagem é desenhada dentro da janela do navegador, tendo o texto fluindo em torno desta, em vez de dispor de uma janela externa exclusiva.

Em um browser que suporte Java, o applet também é exibido dentro da página web. Consequentemente, nem sempre o usuário pode ter certeza se uma imagem materializada em seu browser é um arquivo de imagem ou um applet. Quando um applet é colocado em uma página web, é definida uma área específica para a sua exibição. Essa área de exibição pertence ao applet, para ser utilizada conforme a sua execução. Alguns applets utilizam essa área para apresentar

animações; outros a utilizam para exibir informações a partir de um banco de dados ou para permitir que o usuário selecione itens ou digite informações. A largura e altura dessa área são definidas no arquivo HTML, dentro da chamada ao applet, nos campos width e height.

Anexo 2 Representação compatível com JavaCC para linguagem CIF

```
PARSER_BEGIN(CifParser)

public class CifParser
 {
 public static void main(String args[]) throws ParseError
  {
  CifParser parser = new CifParser(System.in);
  parser.cifFile();
  }
 }
PARSER_END(CifParser)

<DEFAULT>
TOKEN:
{
<P: "P">
| <B: "B">
| <R: "R">
| <W: "W">
| <L: "L">
| <S: "S">
| <D: "D">
| <F: "F">
| <C: "C">
| <E: "E">
| <T: "T">
| <M: "M">
| <X: "X">
| <Y: "Y">
| <DIGIT: ["0"-"9"]>
| <UPPER: ["A"-"Z"]>
| <SEMI: ";">
| <MINUS: "-">
| <LBRACE: "(">
| <RBRACE: ")">
| <BLANK: ~["0"-"9", "A"-"Z", "-", "(", ")", ";"]>
}

void cifFile():{}
 {
 ((blank())* [command()] semi())* [endCommand()] (blank())*
 }
void command():{}
 {
 primCommand() | defCommand()
 }
void primCommand():{}
 {
  ( polygonCommand() | boxCommand() | roundFlashCommand() |
        wireCommand() | layerCommand() | userExtensionCommand() |
        commentCommand() | callCommand() ) (blank())*
 }
void defCommand(): {}
 {
 <D> (blank())*
 ( defDeleteCommand() |
 defStartCommand() (blank())* semi() ( [primCommand() (blank())*] semi())*
  <D> defFinishCommand() )
 }
void defFinishCommand():{}    { <F> (blank())* }
void defDeleteCommand():{}    { <D> integer() (blank())* }
void defStartCommand():{}     { <S> integer() [ sep() integer()  sep() integer()] }
```

```
void polygonCommand():{}        { <P> path() }
void boxCommand():{}            { <B> integer() sep() integer() sep() point() (sep())* [ point() ] }
void roundFlashCommand():{}     { <R> integer() sep() point() }
void wireCommand():{}                   { <W> integer() sep() path() }
void layerCommand():{}                  { <L> (blank())* shortname() }
void callCommand():{}                   { <C> integer() (blank())* transformation() }
void userExtensionCommand():{}          { <DIGIT> userText() }
void commentCommand():{}                { <LBRACE> commentText() <RBRACE> }
void endCommand():{}                    { <E> }
void transformation():{}        { ( (<T> point() | <M> mirrorDir() | <R> point()) (blank())* )* (blank())* }
void mirrorDir():{}             { (blank())* (<X> | <Y>) }
void path():{}                  { point() (blank())* ( sep() point() )* }
void point():{}                 { sInteger() sep() sInteger() }
void sInteger():{}              { (sep())* [<MINUS>] integerD() }
void integer():{}               { (sep())* integerD() }
void integerD():{}              { <DIGIT> (<DIGIT>)* }
void userText():{}              { ( userChar() )* }
void commentText():{}           { ( commentChar() )* }
void semi():{}                  { <SEMI> (blank())* }
void sep():{}                   { upperChar() | blank() }
void regularChar():{}           { <DIGIT> | upperChar() }
void blank():{}                 { <BLANK> }
void shortname():{}             { regularChar() [regularChar()][regularChar()][regularChar()] }
void upperChar():{}             { <UPPER> |<B>|<R>|<W>|<L>|<S>|<D>|<F> |<C>|<E>|<T>|<M>|<X>|<Y>|<P> }
void userChar():{}              { <BLANK> | <DIGIT> | <MINUS> | upperChar() | <LBRACE> | <RBRACE> }
void commentChar():{}           { <BLANK> | <DIGIT> | <MINUS> | upperChar() | <SEMI> }
```

Bibliografia

[ADI97] ADIDA, B. It all starts at the server. **IEEE Internet Computing**, Los Alamitos, Feb. 1997. p. 75.

[ADI97a] ADIDA, B. Taking Web Clients to the Next Level. **IEEE Internet Computing**, Los Alamitos , p. 65-67, Mar./Apr. 1997.

[ALB93] ALBERTI, B. et al. **The Internet Gopher Protocol**. University of Minnesota, IETF RFC 1436, 1993. Disponível por WWW em http://reference.nrcs.usda.gov/ietf/rfc/1500/rfc1436.txt.

[AME96] AMES, A.L.; NADEAU, D.R.; MORELAND, J.L. **The VRML Sourcebook**. [S.l.]: John Wiley & Sons, Inc., 1996. 650 p.

[BAL93] BALASUBRAMANIAN, V. **State of the Art Review on Hypermedia Issues And Applications**. Newark, NJ: Rutgers University, 1993.

[BAR92] BARNES, T.J. et al.. **Electronic CAD *Frameworks***. [S.l.]: Kluwer Academic Publishers, 1992. 196 p.

[BEL96] BELL, G.; PARISI, A.; PESCE, M. **The Virtual Reality Modeling Language**. Version 1.0C Specification, January 1996. Disponível por WWW em http://vag.vrml.org/vrml10c.html

[BEL96a] BELL, G. et al. **An Overview of the Virtual Reality Modeling Language**. Version 2.0, August 1996. Disponível por WWW em http://vrml.sgi.com/moving-worlds/Overview/overview.main.html.

[BEM97] BENDA, M. The Right User Interface. **IEEE Internet Computing**, Los Alamitos, p. 68-70, Mar./Apr. 1997.

[BEN96] BENINI, L.; BOGLIOLO, A.; DE MICHELI, G. Distributed EDA tool integration: the PPP paradigm. In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, ICCD, 1996. **Proceedings**... [S.l.:s.n.], 1996.

[BER93a] BERNERS-LEE, T.; CONNOLY, D. **Hypertext Markup Language (HTML)**. Internet Draft (work in progress), Jul.1993. Disponível

por WWW em http://info.cern.ch/hypertext/WWW/MarkUp/HTML.html.

[BER93b] BERNERS-LEE, T. **Uniform Resource Locators (URL)**. Internet Draft (work in progress), Oct.1993. Disponível por FTP anônimo em ftp.w3.org, no arquivo /pub/www/doc/url-spec.txt.

[BER94] BERNERS-LEE, T. et al. The World-Wide Web. **Communications of the ACM**, New York, vol. 37, no. 8, p. 76-82, 1994.

[BER95] BERNERS-LEE, T; FIELDING, R. T.; NIELSEN, H. F. **Hypertext Transfer Protocol - HTTP/1.0**. Internet Draft (work in progress) - HTTP Working Group, Mar.1995. Disponível por FTP anônimo em ftp.w3.org, no arquivo /pub/www/doc/http-spec.txt.

[BOR97] BORLAND INTERNATIONAL INC. **JBuilder Professional**. 1997.

[BOS95] BOSCH, O.; WOLF, P.; HOEVEN, A. Design Flow Management: more than convenient tool invocation. In: RAMMIG, F.J.; WAGNER, F. R. (Eds.). **Electronic Design Automation Frameworks.** London: Chapman & Hall, 1995. p. 149-158.

[BRA94] BRANDÃO, H.J.R., GONDIN, P.R.L. Terminal Virtual. In: CARVALHO, T.C.H. de B. (Org.). **Arquiteturas de Redes de Computadores OSI e TCP/IP.** São Paulo: Makron Books, 1994. p. 473-505.

[BRE95] BREDENFELD, A. Cooperative Concurrency Control for Design Environments. In: EUROPEAN DESIGN AUTOMATION CONFERENCE, 1995, Brighton. **Proceedings...** Los Alamitos: IEEE Computer Society, 1995. CD-ROM.

[CAM96] CAMPIONE, M; WALRATH, K. **The Java Tutorial**: Object-Oriented Programming for the Internet. [S.l.]: SunSoft Press, 1996.

[CÂM95] CÂMARA, J.; SARMENTO, H. **AutoCap: An Automatic Tool Encapsulator**. In: RAMMIG, F.J.; WAGNER, F. R. (Eds.). **Electronic Design Automation Frameworks.** London: Chapman & Hall, 1995. p. 35-44.

[CAR96] CARRO, L. et al. Ágata : Brazilian Environment for Designing ASIC's. In: UFRGS MICROELECTRONICS SEMINAR, 11., 1996, Porto Alegre. **Proceedings...** Porto Alegre: CPGCC da UFRGS, 1996. p. 1-4.

[CHA96] CHAN, F.; SPILLER, M.D. **The Design of an Internet-Based Collaborative System Architecture**. U.C. Berkeley, 1996.

[CHA96a] CHAN, F.; SPILLER, M.D. **Enabling Technologies for Collaborative CAD**. U.C. Berkeley, 1996.

[CHA97] CHAN, F.; SPILLER, M.D. **WELD Infrastructure for a Distributed Design Environment**. U.C. Berkeley, 1997. Disponível por WWW em http://www-cad.eecs.berkeley.edu/~fchan/docs/weldarch.htm.

[CHP97] CHANG, P.I. **Java Web Server Ships**. JavaWorld, IDG Communications, June 1997. Disponível por WWW em http://www.javaworld.com/javaworld/jw-06-1997/jw-06-webserver.html.

[CHS97] CHATTERJEE, S.; YAKOWENKO, W.J. Architecture for a Web-Acessible Simulation Environment. **IEEE Computer**, Los Alamitos, p. 88-91, Jun. 1997.

[COM9?] COMPASS DESIGN AUTOMATION. **VHDL Scout - A Practical Introduction to IEEE Standard 1076-1987, the VHSIC Hardware Description Language**.

[CON87] CONKLIN, Jeff. Hypertext: An Introduction and Survey. **IEEE Computer**, Los Alamitos, v.20, n.9. p.17-40, Sep.1987.

[DON95] DONNELLY, C.; STALLMAN, R. **The Bison Manual**. [S.l.]: Free Software Foundation, 1995, 104 p.

[FIE97] FIELDING, R.T.; KAISER, G. The Apache HTTP Server Project. **IEEE Internet Computing**, Los Alamitos, p. 88-90, Jul./Aug. 1997.

[FLA96] FLANAGAN, D. **Java in a Nutshell**. [S.l.]: O'Reilly & Associates, 1996. 460 p.

[GON97] GONG, L. Java Security: present and near future. **IEEE Micro**, Los Alamitos, p. 14-19, May/Jun. 1997.

[GOS96] GOSLING, J.; JOY, B.; STEELE,G. **The Java Language Specification**. Sun Microsystems, July 1996. Disponível por WWW em http://java.sun.com/doc/language_specification.html.

[GOS97] GOSLING, J. The Feel of Java. **IEEE Computer**, Los Alamitos, p. 53-57, June 1997.

[GRA96] GRALLA, P. **Como Funciona a Internet**. [S.l.]: Quark do Brasil, 1996.

[HOR96] HORSTMANN, C.; CORNELL, G. **Core Java**. SunSoft Press, Prentice-Hall, 1996.

[IND95] INDRUSIAK, L.S. **Relatório de Estágio**. Santa Maria: Centro de Processamento de Dados, Universidade Federal de Santa Maria, 1995.

[IND96] INDRUSIAK, L.S.; ZART, M.A. **Network CAD**. Porto Alegre: CPGCC da UFRGS, 1996.

[IND96a] INDRUSIAK, L.S.; REIS, R.A.L. **Visualização 3D de circuitos integrados utilizando VRML**. Trabalho Individual, Porto Alegre: CPGCC da UFRGS, 1996.

[IND97] INDRUSIAK, L.S.; REIS, R.A.L. A WWW approach for EDA tool integration. In: BRAZILIAN SYMPOSIUM OF INTEGRATED CIRCUITS DESIGN, SBCCI, 10., 1997, Gramado. **Proceedings...** Porto Alegre: CPGCC da UFRGS, 1997.

[IND97a] INDRUSIAK, L.S.; REIS, R.A.L. Visualização 3D do Layout de Circuitos Integrados utilizando VRML. In: WORKSHOP DE REALIDADE VIRTUAL, 1., 1997, São Carlos, SP. **Anais...** São Carlos:UFSCAR, 1997.

[JAC95] JACOME, M.F.; DIRECTOR, S.W. Planning and Managing Multi-disciplinary and Concurrent Design Processes. In: RAMMIG, F.J.; WAGNER, F. R. (Eds.). **Electronic Design Automation Frameworks.** London: Chapman & Hall, 1995. p. 159-168.

[JAV97] JAVA.SUN.COM. **The Source for Java**. Disponível por WWW em http://java.sun.com (1997).

[KAH94] KAHN, H.J. Design Representation in EDIF Version 3 0 0 and CFI Version 1.0. In: RAMMIG, F.J.; WAGNER, F. R. (Eds.). **Electronic Design Automation Frameworks.** London: Chapman & Hall, 1995. p. 211-220.

[KIN96] KINDEL, M.; CARRO, L.; REIS, R.A.L. EDIF 200 Parser Development. In: UFRGS MICROELECTRONICS SEMINAR, 11., 1996, Porto Alegre. **Proceedings...** Porto Alegre: CPGCC da UFRGS, 1996. p. 149-152.

[KRA96] KRAMER, D. **The Java Platform:** A White Paper. [S.l.]: Sun Microsystems, 1996.

[KRA97] KRAMER, D. **JDK Documentation**. [S.l.]: Sun Microsystems, 1997.

[KRN91] KRAFT, N. Embedded Tool Encapsulation. In: RAMMIG, F.J.; WAXMAN, R. (Eds.). **Electronic Design Automation Frameworks**. Amsterdan: North-Holland, 1991. p. 9-20.

[KWE95] KWEE-CHRISTOPH, E.; FELDBUSCH, F.; KUMAR, R.; KUNZMANN, A. Generic Design Flows for Project Management in a Framework Environment. In: EUROPEAN DESIGN AND TEST CONFERENCE, 1995, Paris. **Proceedings...** Los Alamitos: IEEE Computer Society, 1995. CD-ROM.

[LAU90] LAUREL, B. **The Art of Human-Computer Interface Design**. [S.l.]: Addison-Wesley, 1990.

[MCC93] MCCOOL, Rob. **Common Gateway Interface Overview**. Illinois University: National Center for Supercomputing Applications, 1993. Disponível por WWW em http://hoohoo.ncsa.uiuc.edu/cgi/overview.html.

[MCG97] MCGRAW, G.; FELTEN, E.W. **Java Security**. [S.l.]: Wiley Computer Publishing, 1997. 194 p.

[MIC96] MICROSOFT CORPORATION. **Microsoft Visual J++**. 1996.

[MUR96] MURRAY, J.D.; VANRYPER, W. **Encyclopedia of Graphics File Formats**. [S.l.]: O'Reilly & Associates, 1996.

[NCS9?a] NCSA - National Center for Supercomputing Applications. **A Beginner's Guide to HTML**. Illinois University. Disponível por WWW em http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html (1996).

[NCS9?b] NCSA - National Center for Supercomputing Applications. **NCSA Fill-out Forms**. Illinois University: NCSA HTTPD Development Team. Disponível por WWW em http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill_out_forms/overview.html (1996).

[NCS9?c] NCSA - National Center for Supercomputing Applications. **NCSA HTTP Server**. Illinois University: NCSA HTTPd Development Team Disponível por WWW em http://www.ncsa.uiuc.edu/General/Internet/WWW/ (1997).

[NCS95] NCSA - National Center for Supercomputing Applications. **The Common Gateway Interface**. Illinois University: NCSA HTTPd Development Team, 1995. Disponível por WWW em http://hoohoo.ncsa.uiuc.edu/cgi/.

[NEM92] NEWMAN, M.; RHYNE, T. **Proceedings of the 3rd IFIP WG 10.2/WG 10.5 Workshop on Electronic Design Automation Frameworks**. Amsterdan: North-Holland, 1992.

[NEW96] NEWTON, A.R. et al. **WELD Project - Web-Based Electronic Design**. U.C. Berkeley, 1996. Disponível por WWW em http://www-cad.eecs.berkeley.edu/Respep/Research/weld/.

[NIE90] NIELSEN, J. **Hypertext & Hypermedia**. Boston: Academic Press, Inc, 1990. p.268.

[PAL89] PALM, E. **Dual Layer Metal 1.2 um CMOS Design Rules**. [S.l.]: European Silicon Structures, 1989. 30 p.

[RAG9?] RAGGETT, D. **HTML Specification**, Disponível por WWW em http://www-uk.hpl.hp.co.uk/people/dsr/ (1997).

[RAM91] RAMMIG, F.J.; WAXMAN, R. **Proceedings of the 2nd IFIP 10.2 Workshop on Electronic Design Automation Frameworks**. Amsterdan: North-Holland, 1991.

[REA94] REALE, G.P.V. Transferência de Arquivos. In: CARVALHO, T.C.H. de B. (Org.). **Arquiteturas de Redes de Computadores OSI e TCP/IP**. São Paulo: Makron Books, 1994. p. 397-471.

[REI96] REIS, R. Concepção Automática de Microsistemas. In: **IV Escola Regional de Informática da SBC-RS.** [S.l.: s.e.], 1996. p. 151-170.

[ROB95] ROBINSON, D. et al. **Apache: an HTTP Server - Reference Manual**. Apache Group, 1995. Disponível por WWW em http://www.apache.org.

[ROC95] ROCKEMBERG, A.; HEYMANN, S.; WESTERHOLZ, U. A User Model Supporting Communication in High-Level Design Systems. In: RAMMIG, F.J.; WAGNER, F. R. (Eds.). **Electronic Design Automation Frameworks.** London: Chapman & Hall, 1995. p. 169-175.

[SAN97] SANKAR, S; VISWANADHA, S. **The Java Compiler Compiler Documentation**. Sun Microsystems, 1997. Disponível por WWW em http://www.suntest.com/JavaCC/.

[SCH95] SCHUBERT, J.; KUNZMANN, A.; ROSENTIEL, W. Reduced Design Time by Load Distribution with CAD *Framework* Methodology Information. In: EUROPEAN DESIGN AUTOMATION CONFERENCE, 1995, Brighton. **Proceedings...** Los Alamitos: IEEE Computer Society, 1995. CD-ROM.

[SHE93] SHERWANI, N.A. **Algorithms for VLSI physical design automation**. [S.l.]: Kluwer Academic Publishers, 1993. p. 100-104.

[SHI97] SHIMMIN, B.F. Java jolts server performance. **LAN Times**, Aug., 1997.

[SIC95] SICARD, E. **Introduction to Microelectronics** version 5.1. [S.l.]: Institut National des Sciences Appliquées de Toulouse, 1995. 130 p.

[SIL 95] SILVA, M.J.; KATZ, R.H. The Case for Design Using the World Wide Web. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, DAC, 32., 1995. **Proceedings...** Los Alamitos: IEEE Computer Society, 1995. CD-ROM.

[SYB97] SYBASE, INC. **PowerJ Enterprise**. Powersoft, 1997.

[TAN96] TANENBAUM, A.S. **Computer Networks**. 3. ed. [S.l.]: Prentice Hall, 1996.

[TAR95] TAROUCO, L.M.R.; OTSUKA, J.L. **Repositório de Educação à Distância**. Universidade Federal do Rio Grande do Sul, 1995. Disponível por WWW em http://www.penta.ufrgs.br/edu/home_edu.html (1996).

[THI90] THIMBLEBY, H. **User Interface Design**. [S.l.]: Addison-Wesley, 1990.

[TRI90] TRIMBERGER, S.M. **An Introduction to CAD for VLSI**. San José: Domencloud Publishers, 1990. 292 p.

[VOS97] VOSS, G. **JavaServer Technologies**. JavaSoft, Java Developer Connection, 1997. Disponível por WWW em http://jdc.javasoft.com/.

[WAG92] WAGNER, F.R. et al. Design version management in the STAR *framework*. In: IEEE/ACM DESIGN AUTOMATION CONFERENCE, 28., 1992. **Proceedings...** Los Alamitos: IEEE Computer Society, 1992. p. 704-710.

[WAG94] WAGNER, F.R. **Ambientes de Projeto de Sistemas Eletrônicos**. [S.l.: s.e.], 1994. 156 p.

[WAY96] WAYNER, P. Inside the Network Computer. **Byte**, New York, p. 105-110, Nov. 1996.

[WOD93] WODASKI, R. **Multimídia**. [S.l.]: Ciência Moderna, 1993.

[WOL97] WOLLRATH, A.; WALDO, J.; RIGGS, R. Java-Centric Distributed Computing. **IEEE Micro**, Los Alamitos, p. 44-53, May/Jun. 1997.

[WUT97] WUTKA, M. **Java:** Técnicas Profissionais. [S.l.]: Berkeley, 1997.

[ZUK97] ZUKOWSKI, J. **Java AWT Reference**. [S.l.]: O'Reilly & Associates, 1997.

Informática  UFRGS

CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

"Ambiente de Apoio ao Projeto de Circuitos Integrados Baseado no World Wide Web"

por

Leandro Soares Indrusiak

Dissertação apresentada aos Senhores:

Prof. Dr. Claudionor Coelho (UFMG)

Prof. Dr. Flávio Rech Wagner

Prof. Dr. Ricardo Pezzuol Jacobi (UnB)

Vista e permitida a impressão.
Porto Alegre, 15/09/98.

Prof. Dr. Ricardo Augusto da Luz Reis,
Orientador.