UMA FERRAMENTA INTEGRADA PARA DESENVOLVIMENTO DE PROTOCOLOS (ProtCAD)

Gardel Moreira Delfino[1,2]
E-mail : gardel@gta.ufrj.br

Rodrigo C. Roman [1,3]
E-mail :roman@cos.ufrj.br

Marcial Porto Fernandez [1]
E-mail : marcial@gta.ufrj.br

Ricardo Neimo B. Lima [1]
E-mail : belem@gta.ufrj.br

Antônio C. Mesquita [1,2]
E-mail : mesquita@lpc.ufrj.br

Aloysio de Castro P. Pedroza [1,2]
E-mail : aloysio@gta.ufrj.br

[1]Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro (UFRJ)
COPPE/PEE - Programa de Engenharia Elétrica
C.P. 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brasil
Tel: (021) 260-5010 Fax: (021) 290-6626
http://www.gta.ufrj.br
[2] Departamento de Eletrônica / EE-UFRJ
[3]COPPE/PESC- Programa de Engenharia de Sistemas e Computação

RESUMO

O uso de técnicas de descrição formal para desenvolvimento de protocolos já é bastante difundido e utilizado no âmbito acadêmico e também no âmbito industrial. Em função do aumento da velocidade das redes de comunicação, cada vez mais desenvolve-se protocolos em circuito integrado, embora seu alto custo justifique a opção por uma solução mista (Co-design), otimizando o custo e desempenho. A ferramenta integrada proposta neste trabalho busca facilitar o desenvolvimento e depuração de protocolos a partir da definição funcional em modelo de Máquina de Estados Finita. Esta proposta tem como base um conjunto de ferramentas desenvolvida anteriormente e a experiência de uso destas ferramentas. Com o objetivo de tornar o sistema independente de plataforma toda a interação com o usuário será feita via WEB.

ABSTRACT

The usage of formal description techniques for developing protocols is already spread in academia and industry. Due to the increase in communication networks speed, frequently protocol development is made in integrated circuits, although its high cost may justify the option for a mixed solution (Co-design), optimizing cost and performance. The integrated tool proposed in this work tries to make easier the development and debugging of protocols from its functional definition in a Finite State Machine model. This proposal is based on a toolset developed earlier and its usage experience. In order to make the system platform-independent all user interaction will be made by the WEB.

Palavras Chaves: Protocolo, Especificação Formal, Ferramenta de Desenvolvimento

1. Introdução

As redes de computadores vêm tendo muitos avanços tecnológicos nos últimos tempos. Como as mudanças se processam de forma acelerada, vê-se a necessidade de tornar o projeto de aplicações de rede cada vez mais automatizado. Neste contexto, o papel das técnicas de descrição formal (TDF) tem se tornando cada vez mais importante.

As técnicas citadas apresentam a comodidade de se projetar uma determinada aplicação distribuída, em nível de abstração maior do que a implementação direta em linguagens de programação. Além disto, diversas ferramentas de projeto utilizando TDF permitem que se valide e teste uma determinada arquitetura antes de se partir para sua implementação em código executável.

Neste artigo, pretendemos dar uma amostra do que existe em termos de ferramentas disponíveis para esta utilização, além de apresentarmos uma proposta para construir um Sistema Integrado de Projeto de Protocolos (ProtCAD), que utiliza basicamente a TDF Estelle [Diaz89]. Este sistema propõe-se a modificar a arquitetura de um Sistema de Auxílio ao Projeto de Protocolos em Estelle (CAD-Estelle), que se encontra em utilização no Laboratório do Grupo de Teleinformática e Automação (GTA), acrescentando recursos importantes. O objetivo principal é tornar o projeto de protocolos de comunicação mais intuitivo, facilitando o trabalho de projeto e implementação de protocolos.

Na seção 2, mostramos um estudo comparando algumas ferramentas integradas disponíveis; na seção 3, apresentamos as características gerais da ferramenta ProtCAD; na seção 4, detalhamos as características dos seus diversos módulos e na seção 5 mostramos algumas das decisões feitas para a implementação do ProtCAD.

2. Avaliação de Algumas Ferramentas Disponíveis

Esta seção apresenta uma pequena introdução à TDF Estelle e um estudo comparativo de várias ferramentas integradas para desenvolvimento de protocolos disponíveis atualmente. Selecionamos apenas ferramentas que utilizam a linguagem Estelle e desprezamos ferramentas que executam unicamente uma função isolada, pois nosso foco são sistemas integrados, apesar de nenhum deles ser completo. Esta comparação não se propõe a ser exaustiva para os sistemas de desenvolvimento, mas serviu para situar nossa proposta no contexto atual.

2.1 Linguagem de Descrição Formal Estelle

O objetivo das linguagens de descrição formal é permitir ao projetista a especificação de um sistema em nível mais alto de abstração do que linguagem de programação. Estas técnicas geralmente são baseadas no mapeamento direto de um modelo que pode ser gráfico para uma forma textual ou para outra forma gráfica. São exemplos de tais técnicas a Linguagem Estelle, SDL, Lotos e VHDL .

Estas linguagens, quando utilizadas em conjunto com ferramentas que as interpretam, podem servir como forma de validação de uma dada arquitetura de sistema interativo. Tais ferramentas, partindo da especificação em TDF, permitem o teste do sistema. Alguns aplicativos permitem inclusive a geração de códigos implementáveis, diretamente da TDF. Em geral, tais implementações podem servir apenas como um bom ponto de partida para uma implementação mais eficiente. Entretanto, outros programas já possuem tal grau de sofisticação, que permitem uma implementação com critérios de desempenho bem definidos e com boa eficiência.

A Linguagem Estelle é uma técnica padronizada pela ISO para a descrição de sistemas distribuídos. Ela permite que se especifique um dado sistema seqüencial, a partir de seu modelo em Máquinas de Estado Finitas Estendidas (MEFE) ou Redes Predicado-Ação (RPA). Ela se baseia em instâncias de módulo comunicando-se através de filas FIFO. Cada instância de módulo possui uma ou mais MEFEs que modelam seu comportamento dinâmico, representado pelas transições do sistema. Estas transições baseiam-se em seu estado de origem, seu estado de destino, a condição de disparo e a ação efetuada pelo sistema. Este modelo entretanto não cobre a manipulação de dados, que é descrita em um trecho da especificação que utiliza diretamente a linguagem Pascal, inclusive no uso de funções para comunicação com o usuário (leitura de teclado e exibição na tela).

2.2 Etapas de Desenvolvimento de Protocolos

Em geral, o projetista segue uma seqüência lógica de procedimentos para o desenvolvimento de um protocolo de comunicação. Estes passos podem ser resumidos em um fluxograma como mostrado na Figura 1. Este fluxograma tem em vista a especificação de um protocolo baseado em MEFE (Estelle, por exemplo).

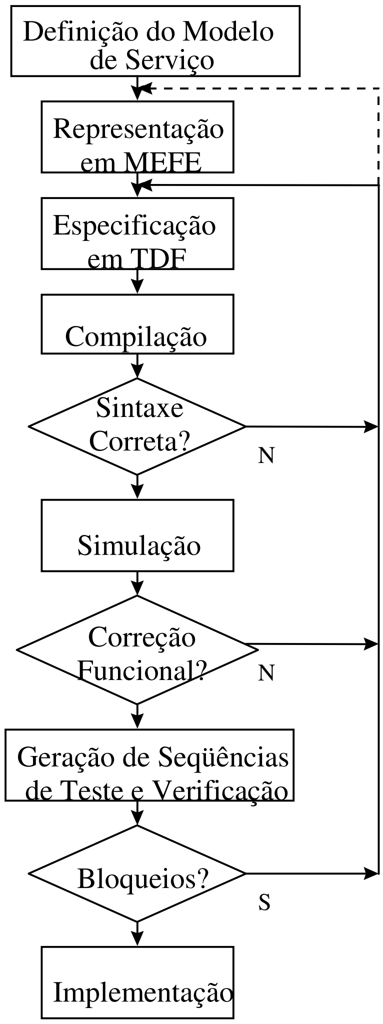

Figura 1 - Fluxograma de etapas de desenvolvimento de um protocolo com TDF

Descrevemos, a seguir, os procedimentos apontador no fluxograma:

a) Definição do modelo de serviço do protocolo.

b) Representação do sistema na forma de MEFEs.

c) Especificação em TDF do modelo descrito pelos itens a e b.

d) Compilação do sistema para simulação, o que consiste numa tradução da especificação para uma forma que lhe permita a depuração automatizada. Verificam-se erros de sintaxe na TDF escolhida. Em caso positivo volta-se para o passo c.

e) Simulação funcional do protocolo. Nesta etapa, busca-se verificar a correção funcional do protocolo, ou seja se ele está coerente com o modelo proposto. Caso haja erros volta-se ao passo c.

f) A partir desta etapa, podem ser geradas automaticamente e simuladas seqüências de teste para o protocolo, procurando cobrir todas as combinações de disparo possíveis. Se não forem detectados bloqueios no disparo das seqüências, segue-se adiante. Caso haja erros volta-se ao passo c.

g) Esta implementação pode ser feita através de uma ferramenta que automatize o processo e que pode não gerar uma implementação ótima, ou, manualmente, através de técnicas de mapeamento para linguagens de programação.

Podemos observar que seria interessante evitar os erros de codificação realizando a depuração apenas ao nível de MEFE. Este caminho é mostrado no fluxograma da Figura 1 pela linha tracejada. Assim, diminui-se os erros de digitação e a correção das falhas de lógica são facilitadas.

Pode-se incluir também uma etapa de análise de desempenho, que permitiria a obtenção de implementações mais eficientes, além do uso da abordagem de implementação mista hardware-software. Isto exigiria uma etapa de particionamento onde se decidiria o que seria implementado em silício e o que seria implementado em software.

Uma ferramenta integrada para utilização no projeto de protocolos necessita de alguns módulos básicos imprescindíveis. O primeiro módulo básico é um compilador, que realiza a tradução do sistema especificado em TDF para uma forma passível de depuração e, eventualmente, de implementação. Esta forma é a entrada do próximo módulo, o simulador.

O simulador faz a interpretação da forma intermediária e permite o disparo de transições no protocolo, verificando-se o caminho de todos os sinais de comunicação entre os módulos. Com isto, busca-se a detecção de bloqueios e verificação da correção no tratamento de dados pelo protocolo descrito.

O verificador integra as funções de geração de seqüências de teste para o protocolo e o disparo automático de tais seqüências. Ele interage com o simulador, enviando-lhe comandos que permita a simulação dos disparos de transições desejadas, detectando assim os possíveis bloqueios que possam ser gerados no uso do sistema.

2.3 EDT

A ferramenta EDT [EDT] é uma suíte comercial de desenvolvimento de protocolos em Estelle. Seu código é de propriedade do grupo BULL francês e do INT (Institut National des Tèlecommunications). Os seus módulos permitem realizar as diversas etapas do projeto de um protocolo, atingindo até a implementação distribuída. Ela se apresenta disponível para uso em diversas plataformas de desenvolvimento, entre elas Solaris, AIX e Linux. Os módulos deste sistema foram escritos em Linguagem C e a linguagem de implementação do sistema alvo é C ou C++.

O elemento base do sistema é o Compilador Estelle (EC-Estelle Compiler). Este módulo possui dois componentes: o Tradutor Estelle e Implementador em C. A tradução faz o mapeamento de Estelle em uma forma intermediária (FI) que permite sua implementação e simulação. O Implementador transforma esta FI em código executável em C, ou opcionalmente, C++.

Após as etapas de compilação, o código gerado pode ser depurado no simulador/debugger. Este ajuda o usuário na detecção e correção de erros de execução do protocolo. Esta simulação é feita com base no código executável e na FI. Na simulação, fica transparente para o usuário como ela foi feita.

Há um outro módulo adicional, que é o Gerador Universal. Este tem por objetivo gerar uma especificação Estelle modificada, substituindo alguns corpos de módulo por corpos universais (UTDG Universal Test Driver Generator). Além disto, pode quebrar uma especificação Estelle em especificações menores de subsistemas, gerenciados por um programa supervisor em C. Este enfoque permite a geração de especificações em Estelle abertas.

Também existe um Gerador de Tabelas de Estados/Eventos. Este permite a criação de Tabelas Estados/Eventos correspondentes ao comportamento de um corpo de módulo definido pelo usuário. Esta tabela é lida da FI que é gerada pelo Tradutor.

Integrando as diversas ferramentas, existe uma interface gráfica, que permite a execução dos módulos usando as facilidades de uma GUI.

2.4 PET-DINGO

O PET-DINGO não apresenta a complexidade do sistema EDT, pois na verdade são dois programas diferentes, o PET [Pet] e o DINGO [Dingo], que servem de base para integração a outras ferramentas.

O PET é na verdade um *parser* para a linguagem Estelle. Ele verifica a sintaxe e a semântica da especificação, montando uma representação estática do protocolo. Ele é o programa de interface básico para ferramentas que utilize Estelle para desenvolvimento. Uma destas ferramentas é o próprio XEC, que mencionaremos posteriormente.

O DINGO gera um código em C++ para implementações distribuídas de Estelle. O código gerado possui funções e estruturas que permitem a inclusão de uma interface gráfica do tipo X11 para os módulos em execução.

2.5 XEC

Este sistema é voltado basicamente para a geração de implementações eficientes a partir de uma Especificação Formal em Estelle [XEC]. Os códigos por ele gerados estão escritos em linguagem C++. Um dos módulos do sistema é o compilador PET, detalhado anteriormente, que a partir de Estelle gera um código binário portável. Gerada esta forma binária, ela é usada como entrada para o XEC, que faz a conversão de binário para código em C++.

O XEC possui uma ferramenta de análise de desempenho (PATO Performance Analysing Tool) e um Debugger Gráfico (EGD Estelle Graphical Debugger). Existe também um módulo que se encarrega da tradução inversa de binário para Estelle, e a ferramenta PetFuse, que permite a extensão Open Estelle para se realizar a especificação do sistema.

Diferentemente do EDT, o XEC se propõe a distribuir livremente seus códigos-fonte nos moldes das licenças GNU. Estas permitem que o código seja livremente alterado pelo usuário desde que seja mantida a estrutura original dos programas, além de se identificar por comentários as modificações introduzidas. Até o presente momento, porém, os códigos e os executáveis ainda não estão disponíveis na Internet.

2.6 CAD Estelle

O CAD-Estelle é, na atualidade, a ferramenta de projeto de protocolos em Estelle utilizada pelo Grupo de Teleinformática e Automação (GTA). Alguns exemplos de protocolos desenvolvidos com esta ferramenta são: Protocolo para gerenciamento hierárquico de redes de telecomunicações [FerPed97], Arquitetura para suporte a mobilidade e segurança no IPv6 [Sauer98], um Protocolo Multiponto-a-Multiponto [Vidal95], IPv6 na abordagem IP sobre ATM e AAL5 , ambos descritos em [CarLim97], e CMIP [LehPed94].

Esta ferramenta apresenta a possibilidade de simulação e testes do protocolo a ser projetado. Além disto, consegue-se gerar uma implementação do protocolo em linguagem Modula-2, que foi escolhida em virtude de facilitar a passagem da parte de manipulação de dados do protocolo para o código executável.

Este sistema também possui um compilador Estelle (CE), que tem funcionamento semelhante aos já descritos anteriormente. A partir da especificação textual em Estelle, após compilada, são geradas duas formas intermediárias: uma estática e uma dinâmica. A estática lista em um arquivo texto os módulos, canais, IPs, primitivas de serviço, estados e transições extraídas da especificação. A forma dinâmica é um arquivo em Modula-2 que traduz a especificação Estelle e que pode ser utilizada para simulação ou implementação dependendo do enfoque escolhido.

O simulador serve como depurador do programa, permitindo a detecção de erros de lógica e também os possíveis bloqueios que podem ser gerados na especificação. Ele se utiliza das formas intermediárias e pode ser usado para a validação do protocolo projetado.

O gerador de seqüências de teste serve para gerar um arquivo texto com a seqüência de possíveis disparos de transições, que permite garantir o funcionamento do sistema sem bloqueios. Estas seqüências são geradas a partir dos dados do modelo em MEF do protocolo, que é extraído manualmente pelo usuário. Utiliza-se o Simulador para testar as seqüências obtidas e este teste também é manual.

O Implementador gera um código executável transparente ao usuário em linguagem Modula-2. Ele também permite a implementação distribuída de protocolos utilizando APIs em nível de transporte na rede (sockets).

2.7 Tabela Comparativa

Com base nos dados levantados, chegamos às conclusões a seguir sobre as ferramentas estudadas, apresentadas na tabela 1.

Tabela 1 - Comparação entre algumas Ferramentas Integradas em TDF Estelle

	Versões Portáveis	Código Gerado	Interface para Ambiente Gráfico X11	Análise de Desempenho Integrada	Implementação Mista Hardware e Software	Estelle Aberta	Editor Gráfico para Especificação
XEC	Sim	C++	Sim	Sim	Não	Sim	Não
PET-DINGO	Sim	C/C++	Não	Não	Não	Não	Não
CAD-Estelle	Não	Modula-2	Não	Não	Não	Não	Não
EDT	Sim	C/C++	Sim	Não	Não	Sim	Não

A partir desta comparação podemos delinear as linhas gerais de como seria este ambiente novo, o ProtCAD. Em primeiro lugar, em nenhuma delas existe editor gráfico que gere uma especificação texto Estelle (ou qualquer outra TDF que siga este paradigma) a partir do modelo em MEFE ou RPA. Tampouco existem ferramentas de implementação Hardware/Software, com particionamento. Estes, portanto, são dois requisitos importantes que serão atendidos pelo sistema ProtCAD proposto.

É verificada a portabilidade ou existência de outras versões em todos os sistemas apresentados, excetuando-se o CAD-Estelle, que roda atualmente apenas em estações Sun-SPARC utilizando SunOS. Isto se deve em muito à utilização da linguagem Modula-2 como linguagem de implementação e da pequena popularidade desta linguagem para desenvolvimento na atualidade. É interessante que o sistema seja escrito em e gere código C ou C++.

A ferramenta de análise de desempenho encontra-se disponível apenas no XEC. Isto pode ser normalmente verificado, visto que as implementações geradas por ele procuram atingir um nível de eficiência bom. Unir uma ferramenta de análise de desempenho para avaliar o particionamento sugerido pelo projetista é muito útil.

Outro ponto a ser ressaltado é a existência de interfaces gráficas para projeto em ambiente X-Windows. Tais interfaces são em geral mais amigáveis e intuitivas do que interfaces em linha de comando, por isso facilitam a atividade do projetista e podem acelerar o processo de projeto do protocolo .

Uma idéia que também deve ser assimilada pelo ProtCAD é a possibilidade de implementação de especificações abertas em Estelle, presente no XEC e no EDT. Tais especificações podem ser compiladas e gerar uma biblioteca de módulos que podem ser simplesmente incluídos numa outra especificação que os queira utilizar para simulação, não se necessitando especificá-los completamente mais uma vez.

3. Características Gerais do ProtCAD

Nesta seção, descreveremos as características gerais do sistema ProtCAD, que serviram como guia para a especificação do sistema.

3.1 Interface com o usuário

O grande objetivo deste sistema é facilitar sua utilização por qualquer usuário, oferecendo-lhe uma interface com um maior nível de abstração que diminui o tempo de desenvolvimento do protocolo. Ao mesmo tempo, este nível mais alto de abstração não deve comprometer o desempenho do sistema, bem como a eficiência de sua implementação.

Uma interface gráfica para programa em Estelle foi apresentada por Goulart [EdGraf], que utilizou uma representação gráfica baseada no modelo estático do protocolo ao invés da representação em MEFE, que era descrita diretamente em Estelle, devido a pouca capacidade gráfica do ambiente de desenvolvimento utilizado na época (MS-DOS). Hoje esta limitação não existe mais e podemos definir uma ferramenta com um nível mais alto de abstração. A representação em máquinas de estado finitas é mais intuitiva e propicia um tempo menor para o desenvolvimento.

Uma forma de especificar um protocolo de forma mais abstrata é o uso de Sintetizador de Protocolos [Sintet]. O sintetizador de protocolos é um módulo que pode indicar um caminho para implementar em MEFE ou RPA o modelo do sistema desejado. Os pontos de partida para a síntese são as interações presentes no modelo de serviço do sistema considerado e as mensagens trocadas entre entidades deste sistema.

3.2 Linguagem de Descrição Formal

A escolha da linguagem Estelle deveu-se à experiência histórica do grupo de trabalho e ao conjunto de ferramentas que já foram desenvolvidas. Outra razão desta escolha é possibilidade de conversão para VHDL para representar o hardware, pois estudos apontam para o fato de que linguagens de descrição com sintaxe semelhante ao Pascal facilitam o mapeamento para VHDL[Pirmez].

Não é necessário que o sistema trate a linguagem Estelle completa, por isso utilizaremos a linguagem Estelle*, que é um subgrupo da Estelle, definida por Courtiat [Court88]. Como o sistema se apresenta como instância de módulos e MEFE, qualquer linguagem textual que siga este paradigma poderá ser utilizada, por exemplo SDL.

3.3 Ferramenta de Co-design

O desenvolvimento de protocolos de alta velocidade exige implementação em hardware. Entretanto, o custo desta implementação pode ser muito alto, exigindo uma avaliação de otimização de particionamento software-hardware. Essa técnica de particionamento, o Co-design, surge como uma solução para problemas de velocidade e custo. Não existe atualmente uma ferramenta automática para particionamento mas podemos oferecer recursos de avaliação de desempenho que indiquem ao projetista uma avaliação de determinado particionamento escolhido. Esta ferramenta semi-automática parece-nos ser o enfoque ideal para nosso sistema.

3.4 Licença de Distribuição

Esta ferramenta deve alcançar o maior número de usuários possível, e também deve receber melhorias ao longo do tempo de forma a manter a atualização e deve respeitar a autoria dos diversos módulos integrados. Uma forma possível de atingir esse objetivo é enquadrar todo o sistema conforme a licença GNU, que resumidamente diz [GNU]:

- O software será disponibilizado no formato programa fonte mas a autoria é mantida. Isto é, o código fonte é público mas os direitos autorais são respeitado.
- Qualquer pessoa tem o direito de alterar o código para acrescentar ou modificar sua funcionalidade, passando a ter o direito de autoria sobre o trecho de código modificado. O código original continua sob a autoria dos autores iniciais.
- A alteração ou inclusão de software com licença GNU **obriga** que o software alterado tenha também a licença GNU.

4. Especificação dos Módulos do ProtCAD

Nesta seção, apresentaremos o diagrama geral do sistema e a especificação dos diversos módulos do ProtCAD assim como suas principais características.

A proposta da ferramenta de desenvolvimento ProtCAD baseia-se na divisão em módulos bem definidos e relacionados a tarefas particulares. Esta divisão pode facilitar a compreensão do processo e, conseqüentemente, facilitar a manutenção e os eventuais acréscimos futuros. Isto é tanto mais importante porque a necessidade de se dividir o trabalho aparece naturalmente se tivermos em mente que a implementação da ferramenta completa será realizada por diversas pessoas (podendo até isto acontecer em épocas ou lugares diferentes).

Para que se alcance o objetivo final, devemos lembrar que a documentação dos módulos em desenvolvimento deve ser a mais perfeita possível, de modo que outras pessoas possam dar continuidade ao trabalho sem maiores dificuldades. Numa visão ideal de projeto, seriam definidas primeiro as ligações dos módulos com os seus subseqüentes e adjacentes, ou seja, quais os parâmetros de entrada e de saída de cada um deles. Feito isso, diversas pessoas implementariam os módulos de maneira paralela, comunicando-se o máximo possível e realizando testes, quando necessário. Deste modo, a integração dos módulos seria trabalho relativamente simples e teria mais chances de obter sucesso do ponto de vista de eficiência.

Nas próximas seções, vamos analisar, do ponto de vista funcional, cada módulo do sistema. Os parâmetros de entrada e saída serão analisados da maneira mais conveniente possível.

4.1 Diagrama Geral

Dentro desta filosofia, apresentamos na Figura 2 o diagrama geral proposto para o ProtCAD, enfatizando as ligações entre os módulos e os parâmetros de entrada e de saída de cada um deles. Nesta figura, os retângulos representam módulos e as elipses representa arquivos texto de interface intermediária, onde LDFP significa Linguagem de Descrição Formal de Protocolos, e LGDP, Linguagem Gráfica de Descrição de Protocolos, que pode ser do tipo RPA ou MEFE, por exemplo.

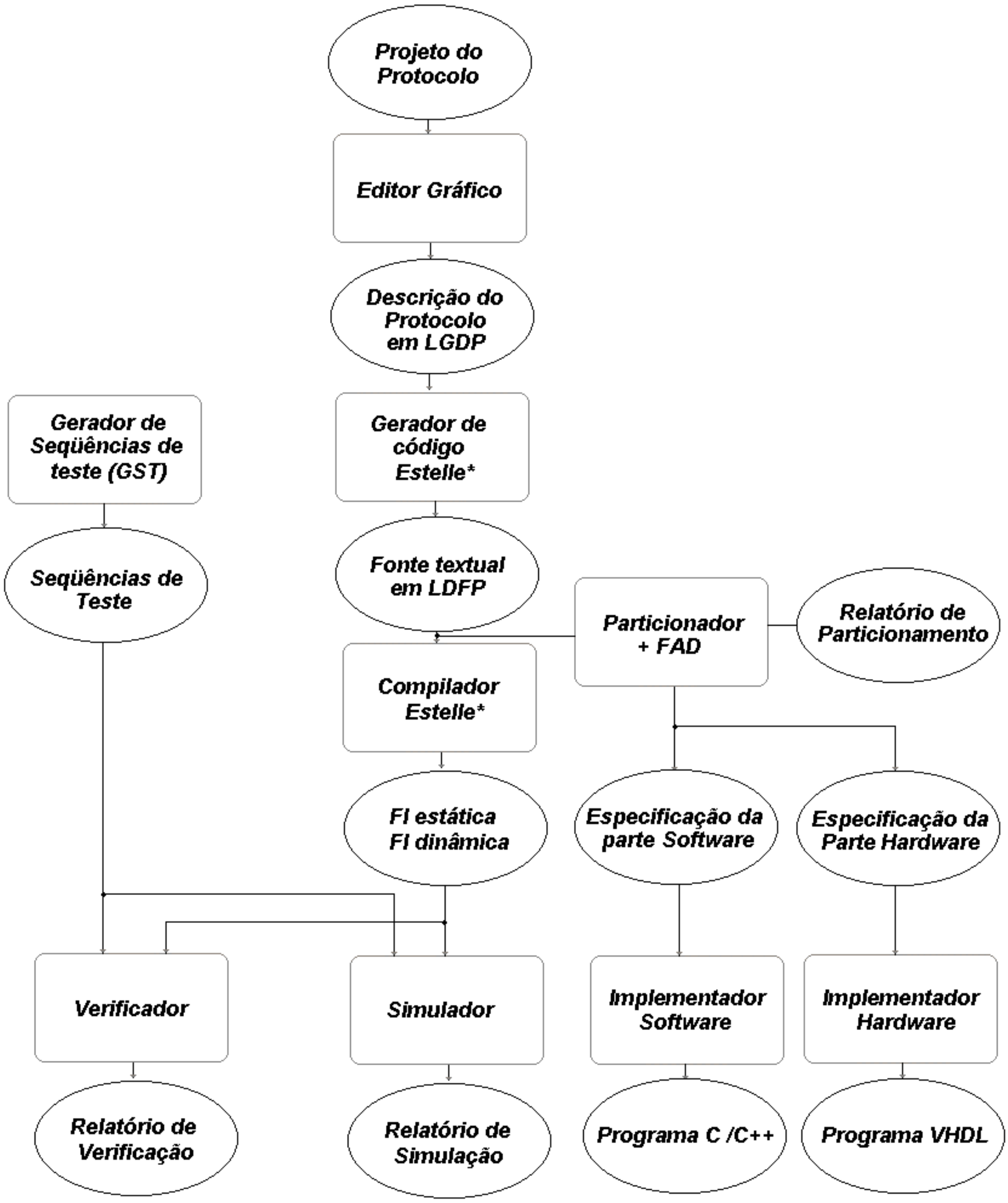

Figura 2 - Diagrama geral dos diversos módulos do ProtCAD

4.2 Editor Gráfico

O módulo de edição gráfica tem a função principal de tornar a especificação de um protocolo algo de mais alto nível do que sua especificação em linguagens textuais. Um exemplo deste módulo foi desenvolvido por Goulart [Edgraf]. De fato, isto permitiria que um projetista de protocolos ficasse livre da incumbência de se familiarizar com uma linguagem textual. Bastaria apenas que ele estivesse familiarizado com o conceito de máquina de estado finito e de módulos de serviços, que são definições bastante intuitivas. É claro que além disso há um aspecto sutil de projeto envolvido nesta proposta: uma descrição de um protocolo em linguagem gráfica permite o mapeamento em uma ou várias linguagens textuais com maior facilidade e eficiência, permitindo que a ferramenta originalmente concebida para Estelle possa ser adaptada para outras linguagens de especificação formal.

Dentro deste panorama, o parâmetro de entrada deste módulo seria a própria idéia do protocolo que se quer implementar, utilizando módulos de serviços compostos de MEFE ou RPA. Ou seja, dentro de cada módulo de serviço, o projetista descreveria uma ou mais máquinas de estados, em função das tarefas realizadas por cada um.

A primeira especificação de um protocolo passa pela especificação dos módulos e seus respectivos canais de uma forma bastante abstrata. Um exemplo de visualização desta primeira definição é mostrado na Figura 3. Neste exemplo, criamos os nomes dos módulos, seus pontos de interação e definimos o formato das PDUs e SDUs. Ao selecionar o canal, é mostrada uma tela onde se editam o nome do ponto de interação e o formato da mensagem, conforme exemplo da figura.

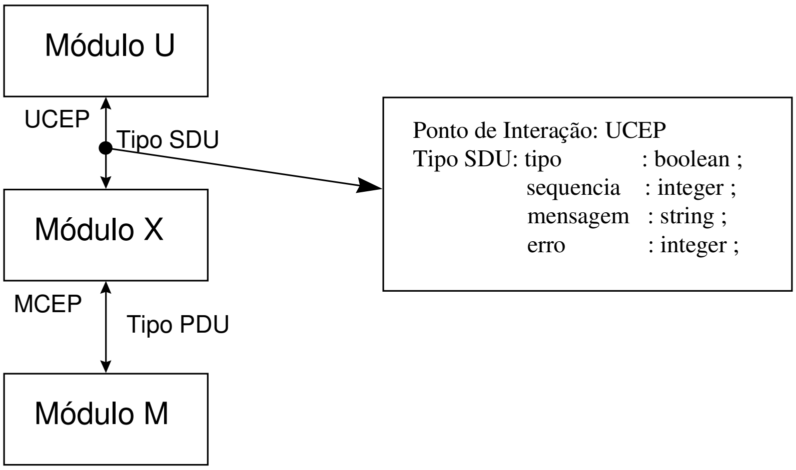

Figura 3 - Exemplo de edição de um canal

A segunda etapa de especificação é o comportamento do módulo. O controle de qualquer protocolo pode ser modelado através de uma RPA ou uma MEFE, assim uma ferramenta gráfica com os objetos estados e transições parece ser bastante útil. A edição de um módulo é mostrada na Figura 4, onde os objetos estado e transição são definidos.

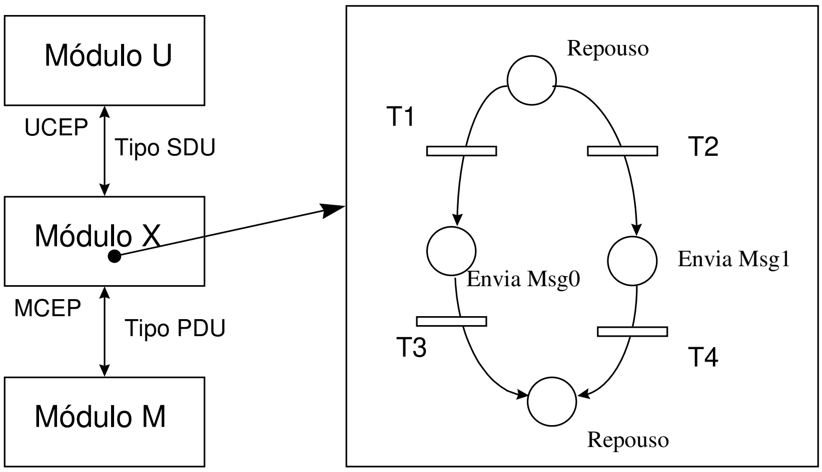

Figura 4 - Exemplo de edição de um módulo

O modelo RPA ou MEFE, entretanto, não consegue representar eficientemente a manipulação de dados, por isso cada transição incluirá as instruções desta manipulação, que em nossa ferramenta será no formato Pascal, seguindo a especificação Estelle. A edição de uma transição é mostrada na Figura 5.

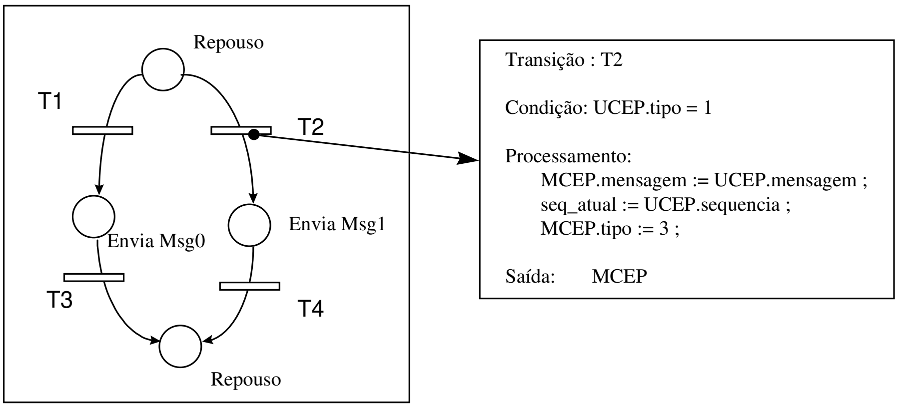

Figura 5 - Exemplo de edição de uma transição

O gerador de código Estelle realiza o mapeamento da especificação gráfica do protocolo em uma linguagem textual de descrição, neste caso Estelle. O objetivo é utilizar uma interface WEB, que além de dispor destes recursos também possibilita uma independência de plataforma de exibição. Como será necessário interatividade intensa com o usuário, provavelmente ela deverá ser escrita em JAVA.

4.3 Compilador

Neste módulo, conforme pode ser visto no diagrama geral apresentado na Figura 2, existe uma bifurcação. Ela nos leva a dois caminhos distintos: validação e implementação do protocolo. É evidente que a ordem mais razoável é primeiro validar o protocolo para depois implementá-lo.

O módulo compilador constitui o início da parte de validação do protocolo projetado e foi desenvolvido por Oliveira [Comp]. O parâmetro de entrada é a especificação textual do protocolo. Com este dado o compilador Estelle gera uma forma intermediária de especificação, que é dividida em duas partes, forma intermediária estática (FI Estática) e forma intermediária dinâmica (FI dinâmica).

A FI estática descreve os pontos de interação entre módulos onde transitam sinais, primitivas de serviço etc. Além disso, descreve o comportamento das transições em função dos estados e dos sinais recebidos e enviados, sem definir, entretanto, o formato detalhado dos sinais (formato das PDUs etc). A FI dinâmica define a interface com o usuário (requisição de dados, mensagens mostradas), o formato das mensagens e o tratamento de dados.

4.4 Simulador (Depurador)

Este módulo tem a função de gerar um objeto passível de execução que permita a correção e a detecção de bloqueios ou situações indesejadas, que foi desenvolvido por Loureiro [Simul]. Existe aí uma enorme interação com o usuário, que pode, eventualmente, disparar manualmente ou automaticamente seqüências de transições desejadas. Um bom exemplo seria a utilização das seqüências de testes geradas pelo Gerador de Seqüências de Teste. Além disso, poderiam ser executadas seqüências aleatórias, que talvez fossem úteis.

Este módulo tem uma interface muito grande com o projetista, por isso deverá ser gráfica e possibilitar a interação para executar os testes desejados e indicar a correção dos erros encontrados. O que seria ideal no nosso ponto de vista seria uma simulação automática, onde o próprio módulo tivesse acesso aos arquivos de seqüências de teste gerados, simulasse todas elas e gerasse como saída um relatório indicando os possíveis bloqueios e erros existentes.

4.5 Gerador de Seqüências de Teste

Este módulo tem a função de gerar um subconjunto ou o conjunto completo de seqüências possíveis e passíveis de ocorrerem, baseado na descrição do protocolo projetado. Os arquivos contendo seqüências de teste são de grande utilidade para validação do protocolo. Geralmente, são gerados dos arquivos básicos denominados seqüências de teste fortes e fracas. As seqüências de teste fortes prevêem todas as possibilidades existentes (mesmo aquelas que o projetista considera improváveis de acontecerem). As fracas prevêem apenas seqüências passíveis de ocorrer, ou seja, as que ocorrem de maneira natural no protocolo. A ferramenta desenvolvida por Soares [GST] não está integrada ao sistema atual e exigirá um trabalho de adaptação.

4.6 Verificador

O módulo verificador não é muito comum nos sistemas levantados, mas é uma ferramenta valiosa para garantir a confiabilidade do protocolo desenvolvido. A idéia parte de uma possível pergunta que poderia ser feita pelo projetista: dado que o sistema está em um estado, é possível (através de quantas transições forem necessárias) chegar a um outro estado qualquer?

Alguns trabalhos já foram realizados nesta área [Bernar], com a utilização de lógica difusa (Fuzzy) como algoritmo de determinação de respostas possíveis, podemos obter um resultado com razoável rapidez. A saída do módulo seria um relatório de verificação, contendo as informações de maneira organizada sobre o comportamento do protocolo em qualquer seqüência aleatória de transições.

4.7 Ferramenta de Auxílio ao Co-Design

A proposta deste módulo do sistema nasceu da idéia de implementação mista de protocolos (no nosso caso parte em ‘software’ e parte em ‘hardware’). Os ganhos que se pode obter em termos de eficiência são significativos e podem ser percebidos intuitivamente. Para que se possa determinar uma configuração ótima de divisão para implementação mista, a ferramenta de projeto integrado deve ser capaz de fornecer medidas de eficiência ou, em outras palavras, determinar parâmetros de desempenho de uma ou diversas configurações possíveis. Desta maneira estaremos trabalhando com parâmetros tais como vazão (throughput), tempo de espera total de um pacote no sistema (system time), tempo de espera na fila de um pacote (queue time), entre outros. Assim, seria gerado um relatório do particionamento, e o projetista deveria ter a percepção para escolher a configuração mais eficiente. Esta seria a chamada implementação semi-automática mista de protocolos de comunicação, e que está sendo proposta para o nosso sistema.

Dentro deste contexto, a entrada deste módulo seria a descrição do protocolo em linguagem textual, e a saída seria o relatório de particionamento que mostraria algumas possibilidades para implementação mista com suas respectivas medidas de desempenho. Com este relatório, o projetista poderia decidir a partição mais conveniente e o particionador separaria a especificação original em duas partes, uma parte a ser implementada em ‘software’ e outra em ‘hardware’.

Dependendo das características do sistema modelado, existe a possibilidade de termos problemas na implementação do protocolo puramente em programa ou somente em circuito integrado. Algumas implementações em software poderiam se tornar lentas e ineficientes, a partir do mapeamento direto da TDF. Nestes casos, uma parte do sistema apresentaria características que justificariam a possibilidade de implementação em hardware. Por outro lado, poderíamos optar pela implementação puramente em hardware do protocolo. Em um caso genérico, esta abordagem poderia, à primeira vista, gerar uma implementação mais rápida. Contudo, talvez tivéssemos um custo de hardware sintetizado inviável.

Com esta idéia em mente, buscaríamos uma solução de compromisso entre o hardware e o software, que geraria um desempenho mais eficiente para o sistema. Nesta etapa, seriam utilizados critérios de particionamento, que poderiam dar uma indicação ao projetista de quais partes do seu protocolo seriam melhor implementadas em circuito ou código de programa.

4.8 Implementador em Software

A implementação em software é obtida com o uso de ferramentas que mapeiem uma especificação em TDF sobre um dado ambiente de projeto. Uma ferramenta para isso foi desenvolvida por Nunes [Impsoft]. A especificação da parte a ser implementada em 'software' é o parâmetro de entrada deste módulo, que gera, como saída, um programa em linguagem já existente e que seja compilável (a nossa proposta seria em C /C++), para que se possa ter eficiência. Este módulo gera um dos produtos finais do sistema. Com este programa pode ser realizada a geração do objeto executável, através do compilador disponível.

4.9 Implementador em Hardware

A implementação em hardware segue procedimento semelhante à de software, sendo que ela mapeia uma forma gráfica de descrição de sistemas para uma linguagem de descrição de hardware. Uma proposta interessante seria a geração de uma descrição VHDL (VHSIC Hardware Description Language), que foi desenvolvida especialmente para especificação e implementação de sistemas em 'hardware' (independente de serem ou não protocolos). Desta maneira, a parte que faltava do produto final é obtida e temos, então, a implementação mista completa. Com o arquivo de descrição VHDL, pode-se utilizar ferramentas já existentes para implementarem esta parte em circuito integrado (ALTERA, por exemplo). Um implementador em hardware pode ser desenvolvido a partir das técnicas de mapeamento propostas por [Pirmez].

5. Detalhes de Implementação do ProtCAD

Os módulos do sitema proposto serão integrados de modo a se ter uma ferramenta completa passível de ser utilizada de maneira simples e agradável. Dentro deste contexto, enquadra-se uma interface de integração gráfica. É claro que os parâmetros de entrada e de saída de cada módulo devem ser gerenciados apropriadamente por esta interface. Para que se possa divulgar futuramente o trabalho a ser desenvolvido de maneira mais eficiente e abrangente (necessidade gerada, principalmente, devido ao seu teor acadêmico), será necessária a utilização da tecnologia INTERNET.

Os recursos que seriam necessários para tanto tornariam os 'sites' tão extensos que o tempo de espera para sua utilização chegaria ao limite do insustentável. O que se propõe, então, é a elaboração de uma interface completa para WEB que poderia ser utilizada de maneira local em uma rede (conceito de INTRANET), e de uma outra interface WEB que possuiria pelo menos um subconjunto das funções da versão completa, para divulgação do trabalho pela INTERNET.

Um trabalho desenvolvido por [Martins] propôs e implementou um sistema de desenvolvimento CAD-Estelle distribuído através de uma rede TCP/IP. Este sistema apresenta uma interface X-Windows e também somente está disponível para o ambiente SunOS. A arquitetura e o módulo de acesso ao sistema serão aproveitados. O diagrama do sistema distribuído do ProtCAD, baseado nesse trabalho, é mostrado na Figura 6.

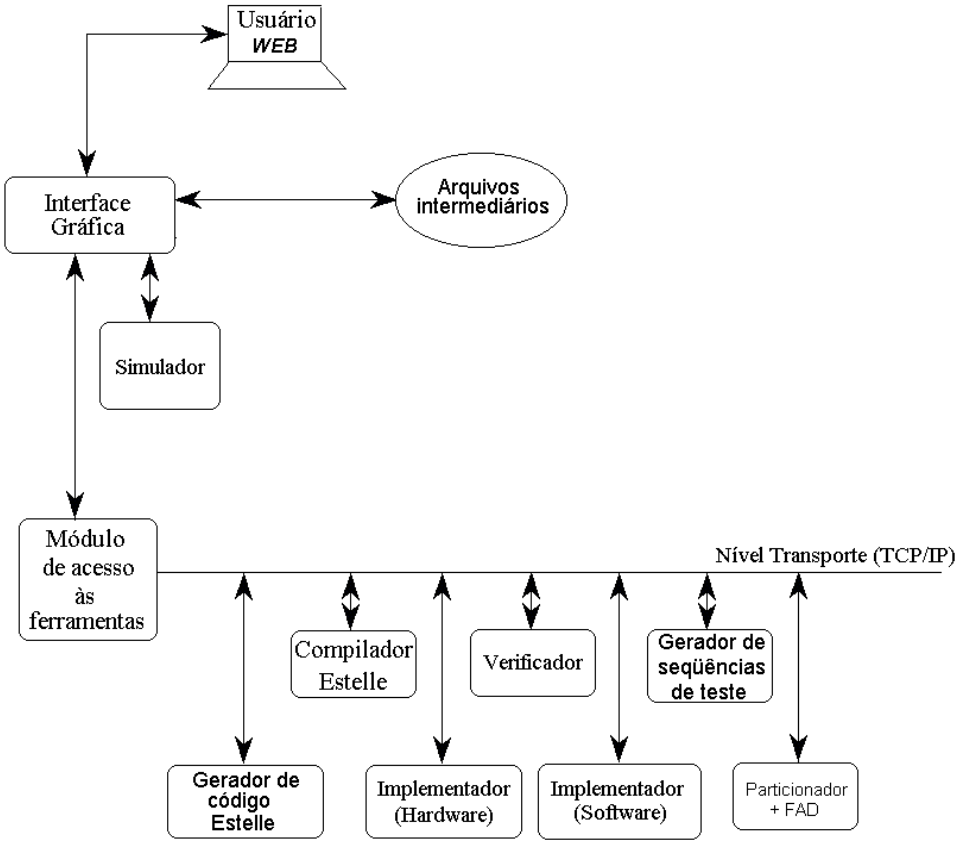

Figura 6 - Diagrama do sistema distribuído

Desta forma, apresentamos o diagrama representativo da integração dos módulos utilizando as duas interfaces propostas. Como pode se notar, o editor gráfico e o simulador são executados localmente na máquina do usuário. Isto se deve ao fato de que muitas interações são feitas entre o usuário e este módulo, de modo que sua execução remota traria perda de eficiência para o sistema.

O ambiente de desenvolvimento da nova ferramenta tentará usar os recursos dentro da comunidade GNU, o sistema operacional Linux ou FreeBSD, compilador GCC, a linguagem JAVA e as bibliotecas necessárias.

6. Conclusão

Este trabalho apresentou uma proposta para um sistema de desenvolvimento de protocolos utilizando a TDF Estelle. A interface do sistema com o usuário, entretanto, se dará de uma forma gráfica e com modelo de MEFE ou RPA, possibilitando seu uso para outras linguagens baseadas neste modelo.

Esta interface gráfica irá gerar uma especificação Estelle que será compilada e simulada com as ferramentas adequadas. Tendo um protocolo correto, podemos aplica-lo em uma ferramenta de análise de desempenho, que poderá indicar ao projetista um particionamento entre software e hardware para implementar o protocolo. Tomada esta decisão, o sistema produzirá uma descrição em VHDL (para hardware) e C/C++ (para software). A utilização de Estelle Aberto (Open Estelle) permitirá a utilização de módulos já prontos, ganhando-se em eficiência e rapidez de implementação.

O ProtCAD será implementado de forma distribuída (Cliente/Servidor) com utilização de interface WEB para interagir com o usuário. Esta arquitetura permite a execução local ou remota, sendo adequada para o ambiente acadêmico.

Com o objetivo de manter a ferramenta atualizada, o código fonte deverá ser disponibilizado sob as regras GNU/GPL. Esta forma de distribuição possibilitará que qualquer pessoa melhore ou aumente os recursos deste sistema.

O sistema ProtCAD deverá ser concluído nos próximos dois anos e deverá possibilitar uma melhoria significativa em relação à ferramenta atual, o CAD-Estelle. Além disso, a concepção dentro de uma filosofia mais aberta deverá garantir uma sobrevida significativamente maior do que a do sistema atual.

7. Referências

[GPL] ITU - INTERNATIONAL TELECOMMUNICATION UNION. *GNU General Public Licence*. http://www.gnu.org/copyleft/gpl.html, outubro 1992.

[FerPed97] FERNANDEZ, M. P., PEDROZA, A. C. "Especificação de um Protocolo para Gerenciamento Hierárquico em Ambiente TMN" In: *15° Simpósio Brasileiro de Redes de Computadores SBRC 97*. São Carlos, pp. 3-18, maio 1997.

[Diaz89] DIAZ, M., at al. *The Formal Description Technique ESTELLE*. 1 ed.North-Holland.USA, 1989.

[Pet] SIJELMASSI, R., STRAUSER, B. *The Portable Estelle Translator: an Overview and User Guide*. Technical Report of National Institute of Standards and Technology, System and Network Architechture Division, Gaithersburg, USA, janeiro 1991.

[Dingo] SIJELMASSI, R., STRAUSER, B. *The Distributed Implementation Generator: an Overview and User Guide*. Technical Report of National Institute of Standards and Technology, System and Network Architechture Division, Gaithersburg, USA, janeiro 1991.

[XEC] UNIVERSITÄT KAISERSLAUTERN. *XEC-The eXperimental Estelle Compiler.* http://rn.informatik.uni-kl.de/activities/Estelle/XEC/index.html, novembro 1998.

[EDT] INSTITUT NATIONAL DES TELECOMMUNICATIONS. *Estelle Development Toolset.* http://alix.int-evry.fr/~stan/edt.html, outubro 1998.

[Edgraf] GOULART, C. C., PEDROZA, A. C. P. "Proposta de uma Forma Gráfica para a Linguagem Estelle" In: *7º Simpósio Brasileiro de Telecomunicações 7º SBT.* Fortaleza, setembro 1989.

[Sintet] MENDES, M. C., PEDROZA, A. C. P. "Uma Ferramenta para a Síntese de Protocolos em Estelle" In: *XXIII SEMISH/SBC*. Recife, agosto 1996.

[Pirmez] PIRMEZ, L., PEDROZA, A. C. P., MESQUITA, A. C. "A Methodology to the Implementation of Distributed Systems in Hardware from a Formal Description" In: *IFIP/ PSTV 95.* Varsóvia, Polônia, junho 1995.

[Comp] OLIVEIRA, R. C. *Um Compilador para a Linguagem Estelle.* Tese MSc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1991.

[Simul] LOUREIRO, F. R. *Um Simulador para a Linguagem Estelle.* Tese MSc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1995.

[GST] SILVA, G. S., PEDROZA, A. C. P. "Um Sistema para Geração Automática de Seqüência de Testes para Protocolos de Comunicação" In: *8º Simpósio Brasileiro de Redes de Computadores 8º SBRC.* Campinas, 1990.

[Impsoft] NUNES, I. P. *Implementação Semi-Automática de Protocolos em Plataformas UNIX.* Tese de MSc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1997.

[Bernar] BERNARDO, O., PEDROZA, A. C. P., LEÃO, J. L. S. "Uma Técnica com Lógica Nebulosa para Verificação de Sistemas Distribuídos" In: XII CBA, Uberlândia, setembro 1998.

[Martins] MARTINS, M. A. *Um Ambiente Distribuído para o Desenvolvimento de Protocolos.* Tese de MSc., COPPE/UFRJ, 1997.

[Sauer98] OLIVEIRA, F. S. G., PEDROZA, A. C. P. "Uma Arquitetura para Suporte a Segurança e Mobilidade sobre IPv.6" In: 16^o Simpósio Brasileiro de Redes de Computador 16^o SBRC, Rio de Janeiro, maio 1998.

[Vidal95] VIDAL, M . T. L., DUARTE, O. C. M. B., PEROZA, A. C. P. "Serviço de Transferência de Dados Multiponto-a-Multiponto" In: 13^o Simpósio Brasileiro de Redes de Computadores 13^o SBRC, Belo Horizonte, maio 1995.

[CarLim] CARLI, E., LIMA R. N. B., PEDROZA, A. C. P., PIRMEZ, L., MESQUITA, A. C. "Implementação em Hardware de Protocolos para Redes de Alta Velocidade" In: 2^o SFBSID, SFBSID'97, Fortaleza, novembro 1997

[LehPed94] LEHMANN, E. O., PEDROZA, A. C. P. "Especificação e Verificação do Protocolo CMIP para Gerenciamento de Redes de Computadores" In: 12^o Simpósio Brasileiro de Redes de Computadores 12^o SBRC, Curitiba, maio 1994.

[Court88] COURTIAT, J. P. "Estelle* (1988) A Powerful Dialect of Estelle for OSI Protocol Description" In: International Workshop on Protocol Specification, Testing and Verification, França, 1988.