INTEGRAÇÃO DE FERRAMENTAS NO SISTEMA AMPLO: Crítica e Proposta de Extensões

Flávio Rech Wagner

RP-149 Março/1991

Trabalho realizado com o apoio do CNPq e da IBM Brasil.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
Av. Osvaldo Aranha, 99
90.210 - Porto Alegre - RS - BRASIL
Telefone: (0512) 271999
Telex: (051) 2680 - CCUF BR
FAX: (0512) 244164
E-MAIL: PGCC@sbu.ufrgs.anrs.br

Correspondência: UFRGS-CPGCC
Caixa Postal 1501
90001 - Porto Alegre - RS - BRASIL

Sumário

Este relatório discute o processo de integração de ferramentas no ambiente AMPLO. São analisadas as alterações necessárias para que, além de um ambiente onde ferramentas de projeto de sistemas digitais já estão integradas de acordo com um modelo de dados pré-estabelecido e uma interface uniforme com o usuário, AMPLO passe a ser um ambiente efetivamente aberto, que permita fácil e uniforme integração de quaisquer ferramentas, novas ou já existentes. São discutidas em particular as extensões a serem feitas no ambiente de modo que ele seja aberto à integração de novos níveis discretos de projeto, sem alteração do modelo de dados atualmente implementado.

Abstract

This report discusses the process of tool integration in the AMPLO design environment. It analyzes the modifications that should be needed in order to extend AMPLO so that, beyond an environment where tools are integrated according to a common data model and a uniform user interface, it turns to be an open framework, which allows an easy and uniform integration of new or already existing tools. The report details the extensions that should be implemented in the environment so that it allows the integration of new discrete design levels, without changing the current data model.

Editor: Ingrid E. S. Jansch Pôrto

Conteúdo

1 Introdução

Um **ambiente de projeto** de sistemas digitais corresponde a um conjunto de ferramentas que estão integradas a duas interfaces principais. A interface de dados permite que as ferramentas compartilhem dados de maneira consistente, isolando-as de detalhes de implementação de um sistema de banco de dados onde estão armazenados todos os dados de projeto. Um pacote de funções gráfico-interativas oferece um conjunto de recursos para a construção de interfaces homogêneas com o usuário para as diversas ferramentas. Além destas duas interfaces principais, o ambiente deve oferecer recursos de gerência de dados, tais como controle de versões e gerência de transações de projeto, necessários à manipulação de grandes volumes de dados por equipes de projetistas.

Um *framework*, por outro lado, é um conjunto de recursos que permite a integração de ferramentas diversas, de modo que se possa construir ambientes dedicados a diferentes aplicações e metodologias de projeto. Um *framework* pode oferecer as seguintes facilidades:

- uma interface de dados genérica, baseada em algum modelo para representação de dados de projeto, que é uniforme para todos os possiveis níveis de abstração de sistemas digitais [1];
- funções para a construção de interfaces homogêneas para comunicação entre usuários e ferramentas, isolando o desenvolvimento das ferramentas do projeto das interfaces;
- um mecanismo para comunicação de dados em tempo real entre ferramentas, de modo que o usuário possa trabalhar simultaneamente em várias tarefas que acessam dados comuns (por exemplo, um editor de esquemáticos e um simulador);
- um mecanismo genérico para gerência de dados (controle de versões e de configurações [1], gerência de transações de projeto, distribuição de dados);
- um mecanismo para a definição de diferentes metodologias de projeto, entendidas como seqüências de ativações de ferramentas a serem seguidas pelos projetistas.

Dependendo de suas características, um *framework* pode permitir a integração de ferramentas já existentes, exigindo mínimas alterações nestas, ou pode demandar alterações consideráveis para adequação das ferramentas às interfaces e utilização dos mecanismos de gerência de dados.

Um *framework* mais flexível pode oferecer um modelo de dados genérico, sobre o qual possam ser definidos diferentes esquemas conceituais, dedicados a determinados conjuntos de ferramentas. Uma característica adicional interessante é a possibilidade de extensão do esquema conceitual sem necessidade de reconstrução do sistema

de banco de dados, com o que se poderia acrescentar ferramentas mantendo-se um esquema bastante sintonizado com o conjunto corrente de ferramentas.

AMPLO [2] é um ambiente de projeto de sistemas digitais em desenvolvimento na UFRGS que reúne as principais características desejáveis em ambientes de projeto, mas apresenta várias deficiências quando analisado como um *framework*. O objetivo deste relatório é analisar o processo de integração de ferramentas disponível no ambiente, apresentando as extensões e modificações que seriam necessárias para que o ambiente passasse a suportar de maneira efetiva a integração de novas ferramentas de projeto. A análise envolve três tipos de extensões de acordo com a natureza das ferramentas a serem integradas:

- extensões para a integração de novas ferramentas baseadas no modelo de dados e linguagens de descrição de hardware atuais;
- extensões para a integração de ferramentas, em particular de descrição e simulação, para novos níveis discretos de projeto;
- extensões para a integração de ferramentas que exijam alterações no modelo de dados.

O modelo de dados suportado pelo ambiente AMPLO já foi exaustivamente discutido em trabalhos anteriores (ver por exemplo [3]). Ele é apenas brevemente reapresentado na seção 2. Na seção 3 é explicado o processo de integração para o conjunto de ferramentas originalmente previsto para o ambiente. Na seção 4 é feita uma crítica aos recursos existentes no ambiente quanto à sua adequação à integração de novas ferramentas, de acordo com as três classes de problemas antes mencionadas. Na seção 5 são apresentadas em detalhe as adaptações que devem ser efetuadas no ambiente para que ele suporte a integração de ferramentas dedicadas a novos níveis discretos de projeto. A migração do ambiente para uma estação de trabalho é discutida na seção 6. Outras extensões, não diretamente relacionadas à integração de ferramentas, mas importantes no contexto de ambientes de projeto, são apresentadas na seção 7. A seção 8 resume as restrições de ambiente quanto ao processo de integração de ferramentas e apresenta brevemente as características do novo *framework* a ser desenvolvido, que deverá resolver os problemas apontados e acrescentar diversos outros recursos.

2 Modelo de dados

Sistemas digitais são descritos em AMPLO de forma modular e hierárquica como **redes de agências**. Cada agência na rede pode ser descrita de uma de duas formas:

- como uma agência **composta**, constituída por uma rede de ocorrências de outras agências, nada sendo dito nesta descrição a respeito da função interna de cada agência; para esta descrição é utilizada a linguagem REDES [4];
- como uma agência **primitiva**, através de uma de três linguagens de descrição de hardware, dedicadas a três diferentes níveis de projeto
  - LAÇO [5], uma linguagem para descrições comportamentais no nível algorítmico,
  - KAPA [6], uma linguagem para descrições estruturais no nível RT,
  - NILO [7], uma linguagem para descrições estruturais no nível de portas lógicas elementares e chaves bidirecionais.

O projetista pode criar diferentes **alternativas de projeto** para cada agência. Alternativas implementam diferentes definições para a interface de uma mesma agência. A interface é composta por **sinais**, que possuem como atributos um nome, um tipo de dados, um sentido e uma largura em bits. As linguagens NILO, KAPA e LAÇO possuem repertórios fixos e pré-definidos de tipos de dados para os sinais.

A cada alternativa podem ser associadas diversas **versões de projeto**. Uma versão pode ser uma descrição composta, em REDES, ou uma descrição primitiva, em qualquer uma das outras linguagens (LAÇO, KAPA ou NILO).

Uma alternativa é originalmente definida num certo nível de projeto, de modo que apenas tipos de dados definidos neste nível podem ser associados aos seus sinais de interface. Embora uma alternativa possa estar definida no nível NILO, por exemplo, não há nenhuma restrição à criação de versões KAPA e LAÇO para a mesma, para o que exige-se apenas uma operação de redefinição para os outros níveis.

Uma versão composta pode conter ocorrências de alternativas de agências (**configurações dinâmicas**) ou de versões de agências (**configurações estáticas**). No primeiro caso, é necessário um processo de configuração (seleção de versões) antes da simulação da descrição.

Uma descrição composta pode ser configurada com ocorrências de versões descritas em diferentes níveis de abstração. Este recurso pode ser utilizado num processo de refinamento passo-a-passo, no qual agências de uma rede vão sendo seletivamente descritas em níveis inferiores de projeto. O ambiente define compatibilidades entre tipos de dados de sinais declarados em interfaces de agências primitivas descritas em diferentes níveis de projeto, de modo a permitir a interconexão destes sinais.

Uma característica importante deste modelo em relação ao processo de integração de ferramentas é a sua larga *granularidade*. Agências são objetos genéricos que em

princípio contêm agregados de elementos ou funções mais primitivos. Assim, por exemplo, uma versão primitiva de uma agência descrita em NILO contém em seu interior uma interconexão de portas lógicas, enquanto uma versão primitiva é descrita em LAÇO através de um grafo de controle semelhante a uma rede de Petri contendo diversos lugares e transições.

Sendo assim, no nível do modelo de dados não é conhecida a estrutura ou função interna de uma versão primitiva. O banco de dados armazena objetos com a granularidade de "agências" e suas interconexões. As representações internas das agências primitivas são geradas pelas ferramentas de descrição (compiladores e editores gráficos) e são armazenadas como arquivos binários no banco de dados, cujos conteúdos só são interpretados pelas próprias ferramentas.

3 Integração de ferramentas em AMPLO

Nesta seção será analisado o processo de integração ao ambiente das ferramentas de descrição (editores e compiladores) e de simulação originalmente previstas no projeto AMPLO, baseadas no modelo de dados apresentado na seção anterior e limitadas às linguagens NILO, KAPA e LAÇO.

3.1 Recursos para a integração

AMPLO foi pensado no início como um ambiente a ser construído a partir de um conjunto pré-definido de ferramentas, a saber compiladores, editores gráficos e simuladores para as linguagens de descrição de hardware LAÇO, KAPA e NILO. Foi então definido um modelo uniforme para representação e gerência de dados em níveis discretos de projeto, estreitamente vinculado a uma linguagem para a especificação de versões compostas (REDES). REDES e o modelo de dados são orientados aos tipos de dados disponíveis em NILO, KAPA e LAÇO para a definição de sinais de interface das agências. Para este modelo (na realidade um esquema conceitual particular para o conjunto definido de ferramentas) foi implementado um conjunto de funções de acesso aos dados, que operam sobre um sistema de arquivos [8]. Estas funções permitem que os dados sejam acessados como objetos complexos e garantem determinadas restrições de integridade. O banco de dados do AMPLO, portanto, faz uma realização direta do modelo de dados especificado.

Além da interface de dados, foi construído um pacote gráfico (PG) [9] contendo funções comuns aos diversos editores e que auxiliou o desenvolvimento homogêneo de suas interfaces com o usuário. Uma linguagem e um pacote de funções (LINUS e PIU) [10] permitem a definição de telas, cardápios e outros recursos interativos, assim como seu seqüenciamento na construção do diálogo com qualquer ferramenta.

Está sendo implementado um simulador multi-nível [11,12,13] que permite a validação de qualquer descrição gerada ao longo de um processo de projeto por refinamentos sucessivos, na qual diferentes módulos estejam descritos em diferentes níveis de projeto.

Tendo em vista que o ambiente é dedicado à simulação, o modelo conceitual foi estendido para incluir objetos auxiliares relevantes à gerência e controle do processo de simulação (estímulos, resultados, etc). Um sub-ambiente especializado de simulação está em implementação oferecendo recursos gráfico-interativos que permitem a manipulação destes objetos e são comuns aos três níveis de projeto [14,15,16].

Finalmente, foi construída uma interface de alto nível com o usuário, denominada LAGO [15,16,17], que permite acesso às diversas ferramentas e navegação gráfico-interativa sobre os objetos armazenados no banco de dados através de um *browser* dedicado ao modelo de dados. Em princípio, LAGO ofereceria recursos especiais para gerência de equipes de projeto e de transações de projeto, que no entanto não foram ainda implementados, tendo ficado dependentes de uma extensão adicional ao sistema de banco de dados.

3.2 Processo de integração das ferramentas de descrição

Utilizando os recursos antes apresentados, as ferramentas de descrição (editores gráficos e compiladores) foram desenvolvidas e integradas da seguinte maneira:

- suas interfaces-usuário foram especificadas e construídas com auxílio dos pacotes PG e PIU;
- o acesso aos dados foi codificado dentro de cada ferramenta na forma de chamadas às funções de acesso ao banco de dados;
- as ferramentas foram integradas a LAGO, de modo que sua ativação se dá unicamente dentro do contexto apropriado.

Note-se que, em função da larga granularidade do modelo de dados, a interpretação das representações internas das versões primitivas das agências é feita dentro das próprias ferramentas. Assim, o editor gráfico NILO, o compilador NILO e o simulador NILO, por exemplo, compartilham o conhecimento a respeito das estruturas de dados internas da linguagem NILO. Estas ferramentas acessam a base de dados de forma bastante simplificada, através de funções de tipo "busca versão primitiva" (que traz um arquivo binário da base de dados que corresponde à versão desejada) e "cria versão primitiva".

Já o editor REDES tem uma interação bem maior com a base de dados, pois é através dele que são geradas as versões compostas. Basicamente, o usuário realiza as seguintes tarefas:

- busca da base de dados uma *alternativa* A_i (i.e. uma definição de interface) para uma *agência* A, ou cria uma nova *alternativa* A_i para A (neste caso cria sinais de interface e seus atributos)
- busca da base de dados uma *versão composta* $A_{i,j}$ para A_i, a partir da qual será criada, por edição, uma nova *versão composta* $A_{i,k}$, através das seguintes ações (opcionalmente o usuário pode criar uma nova *versão* unicamente a partir de uma alternativa buscada na base de dados ou recém-criada):
  - cria componentes que são ocorrências de outras *agências* já previamente definidas na base de dados (pode-se optar entre ocorrências de *alternativas* ou de *versões* destas *agências*)
  - coloca estas ocorrências em determinadas posições dentro do espaço delimitado para $A_{i,k}$ (i.e. define os valores de atributos geométricos de posicionamento para cada componente)
  - conecta sinais das interfaces dos componentes com outros sinais, sejam da interface de outros componentes, sejam da interface de $A_{i,k}$ (i.e. cria objetos Conexões com seus atributos geométricos de traçado)

3.3 Processo de integração dos simuladores

Os simuladores não são na realidade diretamente integrados à base de dados através das mesmas funções de acesso utilizadas pelos editores e compiladores. Os simuladores devem operar sobre **modelos de simulação** que são redes de versões primitivas (exigência feita pela natureza de seus algoritmos). Para a construção destes modelos a partir das descrições de agências contidas na base de dados, é utilizada uma ferramenta especial denominada "construtor de modelos", que implementa a seguinte seqüência de passos, utilizando recursos para exibição e seleção de objetos disponíveis no *browser* de LAGO:

- seleção de uma *agência* A
- seleção de uma *alternativa* A_i para A
- seleção de uma *versão* $A_{i,j}$ para A_i
- se $A_{i,j}$ for uma *versão composta*, então, para cada componente de $A_{i,j}$, se ele for uma ocorrência de uma *alternativa* B_k de outra agência B, selecionar uma *versão* para B_k, i.e. repetir estes dois ultimos passos para o componente (se o componente for uma ocorrência de uma *versão*, a seleção não é necessária, mas de qualquer forma o último passo deve ser repetido sobre cada componente se a *versão* for composta).

Paralelamente a este processo de *configuração*, o construtor de modelos vai montando a rede de *versões primitivas* necessária à simulação. No final do processo, é criado na base de dados o objeto *modelo de simulação*.

O ambiente de simulação possui um interpretador de uma linguagem de comandos para controle de sessões que se comunica com os simuladores através de um sistema de mensagens que isola os simuladores completamente da interação com o usuário.

Tanto o construtor de modelos como o interpretador da linguagem de comandos foram construídos com a utilização dos pacotes PG e PIU, com o que garante-se a uniformidade na interação com o usuário.

O ambiente de simulação apresenta portanto as seguintes características em relação à integração com as interfaces de dados e com o usuário:

- o construtor de modelos tem uma grande interação com a base de dados e com o usuário, ao fim da qual ele cria um objeto "modelo de simulação";
- no início de uma sessão de simulação, são buscados um objeto "modelo de simulação" e as representações internas (versões primitivas) das agências contidas no modelo;
- os simuladores são ativados pelo interpretador da linguagem de comandos, dele recebendo as estruturas de dados necessárias, buscadas da base de dados, e através dele fazendo a comunicação com o usuário;

- os simuladores propriamente ditos não tem nenhuma interação com o usuário nem com a base de dados.

Com a extensão à base de dados para construção do ambiente de simulação, novos objetos auxiliares (estímulos, sessões, resultados) foram criados. Estes objetos são acessados por outras ferramentas auxiliares, como o gerador de estímulos e o analisador de resultados.

4 Crítica aos recursos para integração de novas ferramentas

Um *framework* não deve se restringir a um conjunto fixo e pré-determinado de ferramentas. Uma de suas grandes vantagens deve residir no apoio à integração de novas ferramentas, tanto aquelas que vierem a ser desenvolvidas especificamente para o ambiente como outras porventura já existentes mas que tenham sido desenvolvidas de forma não integrada.

Nesta seção o ambiente AMPLO será analisado à luz do processo de integração de novas ferramentas em três diferentes situações:

- integração de uma ferramenta perfeitamente compatível com o modelo de dados atual;
- integração de um novo nível de projeto adicionalmente aos três níveis atualmente suportados;
- integração de uma ferramenta que exija uma alteração no modelo de dados.

Como se verá, a primeira situação é quase plenamente suportada pelo ambiente com os recursos hoje disponíveis. A segunda situação exige adaptações em diversas ferramentas, que poderiam no entanto ser implementadas num prazo de tempo razoável. A terceira situação, no entanto, só é possível com uma substituição completa do sistema de banco de dados, além de adaptações em recursos já existentes.

4.1 Integração de novas ferramentas ao modelo já existente

A interface de dados é composta por um conjunto de funções que permite a manipulação dos objetos do modelo de dados. A interface com o usuário é construída com o auxílio dos pacotes PG e PIU. Qualquer nova ferramenta de projeto, que esteja limitada ao uso dos níveis de projeto definidos no modelo de dados, pode ser construída utilizando as funções definidas nestas duas interfaces. Pode-se imaginar por exemplo a construção de uma ferramenta de síntese automática que:

- selecione uma agência, alternativa e versão primitiva, através de um processo de interação com o usuário e com a base de dados semelhante ao encontrado no *browser*;
- a partir de uma versão primitiva descrita em KAPA, por exemplo, gere uma versão primitiva descrita em NILO; tendo em vista a granularidade do modelo, esta geração exige a chamada de uma única função "cria versão primitiva".

A integração desta ferramenta ao ambiente não pode ser completa devido a uma única restrição. A interface LAGO foi construída para um conjunto pré-estabelecido de ferramentas (editores, compiladores e simuladores), e assim não pode ser utilizada

para a chamada a uma nova ferramenta. A ativação desta, na situação atual da implementação do ambiente, precisa ser feita portanto externamente a LAGO.

4.2 Integração de novos níveis de projeto

A característica mais importante do modelo de dados em relação ao processo de integração é a distinção entre versões "primitivas" e "compostas". Esta distinção define a granularidade do modelo e influencia a implementação de todo o ambiente:

- uma linguagem REDES é dedicada à descrição de versões compostas;
- os editores e compiladores NILO, KAPA e LAÇO criam versões primitivas;
- os simuladores trabalham sobre redes de versões primitivas – um processo mestre é responsável pelas interações entre as agências na rede, enquanto processos escravos avaliam a função interna das agências primitivas [11,18], de tal modo que para cada linguagem para descrição de versões primitivas há um processo escravo especializado.

O conjunto de níveis de projeto atualmente suportado pelo ambiente obviamente restringe a aplicação deste. O suporte a novos níveis de projeto, dentro do princípio de associar níveis de projeto às versões primitivas, é portanto altamente desejável. Para que isto seja possível, no entanto, devem ser extensíveis a novos níveis de projeto:

- o modelo de dados, e portanto a implementação das funções de acesso ao banco de dados;
- a linguagem REDES, e portanto o editor e o compilador REDES;
- a linguagem LAGO, e em particular o *browser*;
- o simulador multi-nível;
- as ferramentas auxiliares do ambiente de simulação.

Essencialmente, todos estes recursos e ferramentas devem passar da situação atual, onde o repertório de níveis de projeto (e de tipos de dados existentes em cada nível de projeto) é fixo, para uma nova situação onde o repertório seja variável, o que exige alterações em suas implementações. Estas alterações visam tornar a linguagem REDES, o *browser*, as funções de acesso ao banco de dados e as ferramentas auxiliares do ambiente de simulação automaticamente configuráveis para um conjunto de níveis de projeto que esteja definido na própria base de dados. Esta definição deve ser feita por um usuário qualificado (um administrador da base de dados) através de recursos especiais disponíveis em LAGO. Na seção 5 estas alterações serão examinadas com maior detalhe.

O atual mecanismo de simulação multi-nível [11] já está sendo implementado de forma a permitir uma rápida adaptação para novos níveis de projeto [13]. Os processos

mestre e escravos se comunicam através de "adaptadores" nos quais se concentram todos os detalhes de conversão entre tipos de dados de níveis de projeto diversos. Como exercício de aplicação deste princípio, está em desenvolvimento um processo escravo para a linguagem VHDL [19], para o que são exigidas restrições à linguagem de modo a ela se adequar ao princípio da distinção entre versões primitivas e compostas [20]. A integração de VHDL se dará unicamente no nível do simulador, tendo em vista as demais limitações do ambiente, acima citadas.

Feitas as adaptações devidas no ambiente, o processo de integração de um novo nível de projeto passaria pelas seguintes etapas:

- definição do novo nível, de seus tipos de dados e da compatibilidade destes com tipos de dados de outros níveis;
- construção do processo de simulação escravo para o novo nível;
- construção dos adaptadores de simulação entre os tipos de dados do novo nível e os tipos de dados dos níveis já existentes.

O mecanismo de simulação mestre-escravo existente não comporta simulação analógica, com o que fica excluída a integração do nível elétrico no ambiente (ou pelo menos no sub-ambiente de simulação).

4.3 Integração de ferramentas com extensões ao modelo

O modelo conceitual do AMPLO é restrito a um conjunto de ferramentas pré-estabelecido, destinadas à descrição e simulação de sistemas em três níveis de abstração particulares. Este modelo deveria ser aberto à inclusão de características necessárias para outras aplicações importantes, como síntese e teste. Outras aplicações podem exigir novos objetos como vetores de teste ou atributos como restrições de projeto (área, velocidade,etc).

Quando da extensão do modelo de dados inicialmente especificado, com vistas à integração do ambiente de simulação, por exemplo, foi necessária a codificação manual de um novo conjunto de funções para manipulação dos novos objetos e relações acrescentados (modelos de simulação, estímulos, sessões, resultados). Esta forma de extensão do modelo de dados é altamente inconveniente, pois exige um grande trabalho de codificação de novas funções de acesso, além de modificação das funções já existentes para considerar as relações com os novos objetos e atributos.

A solução para o problema está no suporte a um modelo de dados semântico genérico, sobre o qual possam ser definidos esquemas conceituais dedicados a diferentes aplicações. O sistema de banco de dados oferece funções de acesso a objetos do modelo de dados básico e um mecanismo para a construção de esquemas conceituais. O sistema então automaticamente realiza o mapeamento do esquema conceitual, que é visto pelas ferramentas de projeto, para o modelo de dados básico.

Na prática, existem dois graus de generalidade possíveis para este modelo de dados semântico básico:

- um modelo de dados genérico para aplicações de CAD quaisquer, baseado seja em objetos com larga granularidade, como no sistema BDPAC [21], seja em objetos bastante primitivos, tal como previsto nos sistemas CWS [22] e NELSIS [23];

- um modelo de dados dedicado à aplicação de CAD para sistemas digitais, tal como previsto no sistema GARDEN [24,25].

Em qualquer um destes casos, a solução implica a substituição completa do sistema de banco de dados hoje existente no AMPLO, no qual o esquema conceitual é fixo, por um novo sistema, que suporte o modelo de dados genérico e o mecanismo para definição de esquemas conceituais. Este relatório não entrará em detalhes a respeito desta solução. Ela está sendo estudada em conjunto com a migração do ambiente AMPLO para uma nova plataforma, baseada em estações de trabalho. Nesta nova situação, as ferramentas do AMPLO poderão ser integradas no novo ambiente definindo-se neste um esquema conceitual que reproduza o modelo hoje suportado em AMPLO.

Além das alterações na interface de dados, a extensão para novas aplicações exige ainda modificações em LAGO, para que as novas ferramentas possam ser ativadas dentro do contexto do ambiente e para que o *browser* considere os novos objetos, atributos e relações do esquema conceitual. No momento, por exemplo, embora o ambiente de simulação tenha sido implementado, o *browser* não permite a visualização dos objetos auxiliares criados para suporte à simulação Para suporte a esquemas conceituais variados, o *browser* deve ser dinamicamente configurável, o que exige seu completo re-projeto.

Extensão para o nível de layout Uma extensão muito importante no ambiente diz respeito à consideração do nível de *layout* de circuitos impressos ou de máscaras de fabricação de circuitos integrados. O projeto físico de um sistema digital exige a manipulação de atributos geométricos variados que deveriam ser considerados no modelo de dados e nas ferramentas de projeto. Exemplos são:

- dimensões das interfaces das agências;

- localização de cada componente dentro de uma versão composta;

- localização de cada sinal na interface da agência;

- traçado exato das conexões entre sinais nas interfaces de componentes de uma versão composta;

- no caso de circuitos integrados, camada do *layout* em que está implementado cada sinal de interface e cada conexão (uma conexão pode ser formada por segmentos implementados em camadas diversas).

Exceto pelas camadas do *layout*, os demais atributos acima listados já são hoje armazenados na base de dados, tendo em vista que as linguagens possuem formas gráficas que são geradas por editores. No entanto, estes atributos não estão definidos nas formas textuais das linguagens. Na interface de dados, distingue-se entre funções de manipulação de objetos descritos graficamente e de objetos descritos textualmente. Esta distinção deve ser removida, de modo que os atributos geométricos sejam definidos também nas descrições geradas textualmente, o que implica obviamente adaptações nas linguagens. Quanto à alteração no modelo de dados, ela poderá ser facilmente implementada no momento da migração do ambiente para estações de trabalho, quando esquemas conceituais variados poderão ser definidos de acordo com a aplicação.

5 Adaptações para a integração de novos níveis de projeto

Nesta seção são listadas as adaptações principais que devem ser realizadas no ambiente para que ele suporte facilmente a integração de novos níveis discretos de projeto, assim como de ferramentas para estes níveis, em particular ferramentas de descrição (editores e compiladores) e de simulação (na realidade um processo escravo para o novo nível).

5.1 Adaptações em LAGO

Há três classes principais de adaptações a serem feitas na linguagem de acesso global LAGO visando à extensibilidade do ambiente a novos níveis de projeto:

- mecanismos para a definição dos novos níveis de projeto;
- adaptações para adequação de LAGO a um repertório variável de níveis de projeto;
- alterações no *browser*.

Definição de níveis de projeto A definição dos novos níveis de projeto disponíveis numa dada configuração do ambiente deve ser feita em LAGO através de um usuário privilegiado (um administrador da base de dados). A existência de usuários com diferentes privilégios já está prevista e implementada em LAGO. Devem ser criados cardápios adicionais com funções necessárias a esta definição:

- inserção de um novo nível – devem ser fornecidos
  - o nome da linguagem de descrição de hardware;
  - a lista de tipos de dados disponíveis nesta linguagem para a declaração de sinais de interface;
  - tabelas de compatibilidade de cada um dos tipos de dados da nova linguagem com os tipos de dados das demais linguagens integradas ao ambiente;
- remoção de um nível – sugere-se que esta remoção seja apenas virtual, permanecendo a definição do nível na base de dados; o nível "removido" passa a não constar dos cardápios apresentados ao usuário;
- re-inserção de um nível previamente removido.

Um tipo de dados é definido por:

- um identificador;

- um tipo básico (*boolean* ou *integer*) do qual ele é derivado;
- restrições aplicáveis aos demais atributos dos sinais de interface – p.ex. um sinal de tipo *A* só pode ser de entrada (atributo "sentido") e um sinal de tipo *B* (derivado do tipo *boolean*) só pode ter largura de um bit (atributo "largura em bits").

Esta forma simplificada de definição de tipos de dados é válida no contexto de linguagens de descrição de hardware nas quais os tipos de dados encontrados são o *integer* e outros mapeáveis para *boolean*, como *terminal*, *bus* e *clock*, tal como ocorre na família de linguagens do AMPLO.

Para a implementação destas funções em LAGO, deve ser estendido o modelo de dados com a definição de objetos *níveis de projeto*. Um nível de projeto tem como atributos:

- um nome (de uma linguagem de descrição de hardware);
- uma lista de tipos de dados para os sinais de interface, incluindo identificador, tipo básico e restrições em relação aos atributos "sentido" e "largura em bits" dos sinais de interface.

Um outro objeto *tabelas de compatibilidade* contém a definição das compatibilidades entre os tipos de dados das diferentes linguagens suportadas pelo ambiente.

Esta extensão ao modelo de dados exige uma re-implementação do sistema de arquivos sobre o qual operam as funções de acesso, com a inclusão de novos arquivos relativos aos novos objetos e com a adaptação dos arquivos já existentes a um repertório variável de níveis de projeto e de tipos de dados.

Para a manipulação destes novos objetos devem ser implementadas as seguintes funções de acesso na interface de dados:

- definição de um novo nível de projeto;
- remoção de um nível de projeto;
- re-inserção de um nível de projeto;
- extensão da tabela de compatibilidade para um novo nível.

Adaptação a um repertório variável de níveis de projeto A implementação da linguagem de acesso global LAGO foi dedicada ao conjunto de linguagens originalmente previstas. Assim, as funções disponíveis nos vários cardápios consideram apenas a ativação das ferramentas de descrição e simulação para os níveis primitivos NILO, KAPA e LAÇO.

A adaptação necessária consiste portanto em tornar LAGO configurável para um repertório variável de níveis de projeto, o que exige as seguintes alterações:

- no início de uma sessão AMPLO, LAGO busca na base de dados a lista de níveis de projeto da configuração corrente do ambiente;
- LAGO configura dinamicamente todos os cardápios nos quais constem a lista de níveis de projeto.

Para que isto seja possível, a interface de dados precisa ser acrescida de uma função de acesso para a busca da lista de níveis de projeto definidos para a configuração corrente do ambiente. Para LAGO, apenas a informação sobre os nomes dos níveis de projeto é necessária, já que os tipos de dados não são manipulados neste ponto do ambiente.

Adaptações no browser O *browser*, que permite a navegação sobre os objetos e relações armazenados no banco de dados, foi projetado para o conjunto de níveis pré-estabelecido.

Na exibição e apontamento de agências, alternativas e versões, nenhuma alteração precisa ser feita, já que o *browser* exibe o atributo "nível de projeto" de uma versão primitiva ou de uma alternativa de acordo com o definido para o objeto na base de dados.

O *browser* oferece no entanto funções de *pruning* que permitem "recortar" a árvore de objetos exibida na tela de acordo com critérios que podem envolver o nível de projeto, tal como "exibir apenas versões primitivas descritas em NILO". Todos os cardápios referentes a esta funcionalidade devem então ser tornados configuráveis para a lista de níveis de projeto buscada da base de dados.

Chamada de outras ferramentas Na subseção 4.1 foi constatado que LAGO foi projetado unicamente para a chamada de editores gráficos, compiladores e simuladores. Se, além da extensão do ambiente para novos níveis de projeto, deseja-se tambem estendê-lo para novas ferramentas, como síntese e teste (desde que estas não exijam modificações no modelo de dados, o que implica outras questões, discutidas na subseção 4.3), LAGO deveria prever um repertório variável de ferramentas. Uma vez que ferramentas não são consideradas no modelo de dados, sugere-se aqui uma solução diversa, na qual o cardápio que contém a seleção de uma atividade de projeto, e que hoje conta com quatro itens (edição de textos, compilação, edição gráfica e simulação) passe a ser configurável para uma lista variável de atividades que esteja definida num arquivo de configuração a ser carregado junto com o ambiente quando da invocação deste. A seleção de uma outra atividade neste cardápio leva à execução imediata de uma ferramenta que então assume o diálogo com o usuário, tal como já é feito hoje para as ferramentas existentes.

5.2 Adaptações no sistema de banco de dados

Na subseção anterior já foram introduzidas:

- as modificações no modelo de dados para a consideração de um repertório variável de níveis de projeto;

- as funções que devem ser acrescentadas à interface de dados para a criação e busca da lista de níveis de projeto.

Além destas adaptações, todas as funções de acesso à base de dados devem ser tornadas configuráveis para o repertório corrente de níveis de projeto. Exemplos de funções que são influenciadas por este repertório são:

- criação de uma versão primitiva – função deve aceitar qualquer nível de projeto definido na base de dados;

- criação de uma nova alternativa – função deve aceitar não só qualquer nível de projeto definido como também os tipos de dados definidos neste nível para os sinais de interface;

- redefinição de nível de uma alternativa – o mesmo que acima.

Deve-se notar que a compatibilidade de tipos de dados de sinais de agências distintas conectados entre si não é verificada pela interface de dados, e sim no nível da linguagem REDES, pelo editor gráfico ou pelo compilador desta linguagem.

5.3 Adaptações no editor e no compilador REDES

A implementação da linguagem REDES (e por extensão de seu compilador e de seu editor gráfico) foi dedicada ao conjunto de linguagens originalmente previstas para a descrição de versões primitivas. Assim, as funções disponíveis nos vários cardápios consideram apenas:

- a seleção de alternativas e versões descritas em NILO, KAPA e LAÇO para vinculação às ocorrências da versão composta sendo criada;

- a escolha de NILO, KAPA ou LAÇO para a alternativa e/ou versão composta sendo criada, e por conseqüência de tipos de dados destas linguagens para os sinais da interface do novo objeto.

A adaptação necessária consiste portanto em tornar REDES configurável para um repertório variável de níveis de projeto, o que exige as seguintes alterações no editor e no compilador:

- no início de uma sessão com o editor ou o compilador, este busca na base de dados a lista de níveis de projeto da configuração corrente do ambiente, incluindo todas as informações relativas aos tipos de dados (repertório, restrições sobre os atributos), e a tabela de compatibilidades de tipos de dados;

- no caso do editor, este configura dinamicamente todos os cardápios nos quais constem a lista de níveis de projeto e os tipos de dados selecionáveis para os sinais de interface;

- o editor e o compilador realizam a verificação semântica das interconexões dentro da versão composta, de acordo com a tabela de compatibilidades de tipos de dados.

As modificações no editor REDES exigem também adaptações no PIU, já que no momento ele só aceita a especificação de cardápios com itens fixos, definidos através da linguagem LINUS.

5.4 Adaptações no ambiente de simulação

O ambiente de simulação é formado por um conjunto de ferramentas que são afetadas de diferentes maneiras pelas extensões necessárias à integração de novos níveis de projeto. Conforme foi explicado na subseção 4.2, a integração de um novo nível discreto de projeto exige também a construção de um processo escravo para simulação deste nível e de adaptadores para a conversão de tipos de dados.

Construtor de modelos O construtor de modelos de simulação já teve sua função detalhada na subseção 3.3. Ele apresenta dois modos de operação no processo de configuração das ocorrências de alternativas encontradas em versões compostas:

- no modo automático, o usuário especifica critérios para a seleção de alternativas e versões (por exemplo, selecionar sempre a versão primitiva mais recente para NILO);

- no modo interativo, as opções são exibidas na tela e a escolha é feita por apontamento.

Não há adaptações a fazer no caso da construção interativa, já que as opções de versões exibidas são buscadas na base de dados, que fornece o nome do nível de linguagem associado a cada versão.

No caso da construção automática, o cardápio que contém as opções de níveis de projeto para o critério de seleção deve ser tornado configurável dinamicamente. O construtor de modelos, ao ser ativado, deve então buscar da base de dados a lista de níveis de projeto corrente.

Gerador de estímulos As linguagens de descrição textual e gráfica de estímulos (LDTE e LDGE) são orientadas aos tipos de dados encontrados nos níveis de projeto. No caso da descrição textual de estímulos, o compilador da LDTE deve ser configurável para os tipos de dados definidos na base de dados. No caso da descrição gráfica, por outro lado, o editor da LDGE deve ter cardápios configuráveis de seleção

de tipos de dados. Tanto o compilador como o editor gráfico devem buscar na base de dados não só a lista de identificadores de tipos de dados corrente como também as informações sobre restrições aplicáveis à largura em bits admitida para estes tipos de dados. A função de verificação de consistência dos valores atribuídos aos sinais deve ser configurada por estas restrições.

Vinculação de estímulos a modelos O módulo de vinculação de estímulos a sinais de entrada de modelos de simulação não é afetado pelo repertório variável de níveis de projeto e tipos de dados. Embora este módulo verifique se os tipos de dados de um sinal e de um estímulo a este vinculado são idênticos, os tipos de dados podem ser quaisquer.

Interpretador da linguagem de comandos O interpretador da linguagem de comandos que controla as sessões de simulação oferece recursos que estão relacionados aos tipos de dados dos sinais:

- alterar o valor de um sinal, seja de forma permanente (comando "force") ou através de uma atribuição não persistente (comando "put");
- especificar pontos de parada para a simulação condicionados ao valor de uma expressão booleana envolvendo valores de sinais do modelo.

O interpretador deve, a partir dos tipos de dados associados aos sinais envolvidos nestas operações, efetuar a verificação da validade dos valores atribuídos. Para esta verificação, devem ser buscadas na base de dados as restrições sobre os atributos definidas para os tipos de dados.

Simulador O simulador propriamente dito deve ser reconfigurado por um processo de "linking" a cada vez que um processo escravo para um novo nível de projeto é acrescentado. Uma configuração do simulador multi-nível deve incluir:

- o processo mestre;
- processos escravos para todos os níveis de projeto;
- adaptadores para conversão entre os tipos de dados existentes nestes níveis de projeto.

Durante a simulação, o processo mestre, ao ativar uma agência, identifica o nível de projeto no qual ela está descrita e aciona o processo escravo correspondente. Sempre que uma agência causa uma variação no valor de um sinal de interface, o processo mestre aciona os adaptadores para conversão entre o tipo de dados do sinal de onde vem a variação e os tipos de dados dos demais sinais a ele conectados. Este procedimento já implementado no processo mestre deve ser adaptado para o caso de um repertório variável de níveis de projeto e de tipos de dados.

Análise de resultados No módulo de análise de resultados, são exibidas seqüências de eventos ocorridos durante uma sessão de simulação em sinais selecionados pelo usuário, numa forma gráfica semelhante à adotada pelo módulo gerador de estímulos. Uma vez que esta forma gráfica é padronizada para todos os tipos de dados, e as informações exibidas são todas elas provenientes da base de dados, este módulo é independente do repertório de níveis de projeto e tipos de dados.

6 Adaptações para nova plataforma

AMPLO foi implementado sobre uma plataforma de hardware e software bastante limitada (PC e DOS). Esta plataforma tem servido ao desenvolvimento isolado das ferramentas, mas será certamente inviável a integração completa, na mesma, de todas as ferramentas previstas no projeto. Certamente será impossível desenvolver nesta plataforma qualquer projeto real, mesmo que de pequeno porte, utilizando todas as facilidades previstas para o ambiente.

A migração do ambiente para uma estação de trabalho exige, no mínimo, a conversão das duas interfaces principais das ferramentas:

- troca dos pacotes básicos para acionamento dos dispositivos periféricos (IS [26,27] e IE [28,29]) por funções disponíveis na estação de trabalho;
- migração das funções de acesso a base de dados para o sistema de arquivos da estação de trabalho (esta tarefa já está em implementação neste momento).

Estas duas tarefas não devem exigir grande volume de trabalho, uma vez que o ambiente foi inteiramente desenvolvido em linguagem C padrão. As ferramentas de projeto não exigirão nenhuma adaptação, já que sua implementação é completamente independente da implementação das interfaces de dados e com o usuário. O ambiente passará então a rodar numa estação de trabalho, sobre um sistema operacional tipo UNIX, conservando todas as suas propriedades atuais.

Este ambiente estaria obviamente deixando de utilizar recursos disponíveis na nova plataforma. Em particular, duas adaptações podem ser feitas:

- re-projeto da interação das ferramentas com o usuário, abandonando-se o estilo imposto pelo PIU, e adotando-se estilos mais adequados (*pull-down menus*, manipulação direta de objetos na tela, múltiplas janelas, etc), suportados por pacotes padrão tais como X-WINDOWS ou MOTIF;
- utilização de recursos para tornar o ambiente multi-tarefa, permitindo sobreposição de janelas executando diferentes ferramentas, e num segundo passo re-projetando algumas ferramentas para aumentar sua eficiência através da execução concorrente (por exemplo permitindo a edição gráfica de um sistema que está sendo simulado).

7 Outras extensões

Embora o objetivo deste relatório tenha sido analisar o ambiente AMPLO especificamente sob o aspecto da integração de ferramentas, que é fundamental em um *framework*, outras extensões devem ser consideradas para completar o ambiente com um conjunto de recursos adequado.

O ambiente atual é mono-usuário e mono-tarefa. Estas restrições devem ser levantadas com dois desenvolvimentos. Um modelo de concorrência deve ser implementado para permitir o compartilhamento de dados entre projetistas num ambiente distribuído, suportando mecanismos de transações longas e de gerência de equipes. Estudos a respeito já foram efetuados e um modelo foi proposto [30], baseado numa hierarquia de bancos de dados. A linguagem LAGO, inclusive, já foi implementada de acordo com este modelo de cooperação. No momento, uma definição definitiva para este modelo está sendo feita no escopo de uma dissertação de mestrado [31].

Em segundo lugar, deve existir um mecanismo para comunicação em tempo real entre ferramentas. A alternativa mais razoável aqui é a utilização de um mecanismo já existente no sistema operacional sobre o qual venha a ser implementada a versão multi-tarefa do ambiente. Uma versão concorrente do simulador poderia ser implementada, na qual o interpretador da linguagem de comandos (responsável pela interação com o usuário), o processo mestre e os processos escravos das agências ativas fossem executados concorrentemente.

Seria desejável que o ambiente permitisse a definição e controle de metodologias de projeto, que seriam gerenciadas automaticamente, tal como proposto por exemplo nos ambientes Ulysses [32], Cadweld [33] e ADAM [34]. Estes ambientes oferecem mecanismos que permitem o disparo controlado de ferramentas, de acordo com seqüências pré-definidas e condições do sistema sendo projetado, testadas automaticamente pelo ambiente.

Outro recurso interessante é a disponibilidade de uma ferramenta dedicada à construção de interfaces com o usuário, tal como um UIMS (*User Interface Management System*), que permite a especificação da implementação da interação com o usuário de forma independente das ferramentas de projeto. O ambiente CWS [22], por exemplo, oferece um UIMS altamente integrado aos demais recursos de integração de ferramentas do *framework*. No ambiente AMPLO, foi especificado um UIMS restrito, dedicado unicamente à implementação da interação com o usuário nos editores gráficos [35], no qual a especificação da interface dos editores é feita de forma gráfico-interativa.

8 Conclusões e trabalhos futuros

Este relatório mostrou que AMPLO apresenta diversas restrições em relação à integração de novas ferramentas. Estas restrições foram classificadas em três categorias, de acordo com a amplitude das modificações necessárias no ambiente para removê-las:

- restrições à integração de novas ferramentas baseadas no modelo de dados e linguagens de descrição de hardware atuais;
- restrição à integração de ferramentas, em particular de descrição e simulação, para novos níveis discretos de projeto;
- restrições à integração de ferramentas que exijam extensões ao modelo de dados.

Foram sugeridas adaptações no ambiente para resolver os problemas das duas categorias iniciais. No entanto, para a solução da terceira categoria de restrições, o ambiente AMPLO deveria ser completamente re-projetado.

Durante o ano de 1991 estará sendo iniciado o desenvolvimento de um novo ambiente, baseado em estações de trabalho, que deverá considerar várias das características desejáveis em um *framework* realmente aberto à integração de novas ferramentas. Este *framework* deverá eliminar de maneira natural todas as restrições à integração de ferramentas existentes no AMPLO, além de acrescentar outros recursos atualmente não existentes. As principais características deste novo *framework* serão:

- implementação em estações de trabalho profissionais de 32 bits, usando sistema operacional tipo UNIX e recursos gráficos de um pacote padrão;
- suporte a um modelo de dados genérico, baseado no modelo GARDEN [24,25] desenvolvido no Centro Científico Rio da IBM Brasil, sobre o qual podem ser desenvolvidos esquemas conceituais para diferentes aplicações;
- suporte a recursos para gerência de metodologias de projeto [36] que incluirão facilidades especiais para integração de ferramentas;
- suporte a um modelo de cooperação [31];
- suporte à execução concorrente de tarefas;
- implementação sobre o sistema de banco de dados KRISYS [37] desenvolvido na Universidade de Kaiserslautern.

Para este novo *framework* deverão ser portadas todas as ferramentas desenvolvidas para o ambiente AMPLO, a saber editores gráficos e compiladores para as linguagens REDES, NILO, KAPA e LAÇO, a linguagem de comandos LAGO e o ambiente de simulação, incluindo o processo mestre e os processos escravos para NILO, KAPA e LAÇO.

Na seção 5 foram listadas várias adaptações nestes recursos do ambiente AMPLO para permitir a integração, pelo próprio usuário, de ferramentas de descrição e simulação dedicadas a novos níveis discretos de projeto. Estas adaptações também afetariam o sistema de banco de dados (SBD) atualmente utilizado. Como na evolução futura do projeto deve-se migrar para um novo SBD, sobre o qual poderão ser definidos diferentes esquemas conceituais, as modificações sugeridas no SBD atual não devem ser implementadas.

Este relatório dedicou-se principalmente aos aspectos de integração de ferramentas relacionados com a interface de dados do ambiente. Esta preocupação focalizada justifica-se pelo fato de, hoje em dia, existirem recursos para a definição de interfaces com o usuário, na forma de *toolkits*, que se não são tão poderosos como UIMSs, permitem ao menos uma rápida implementação da interação das ferramentas com os usuários, assim como uma rápida migração do ambiente para as estações de trabalho. Recursos de alto nível para a definição de interfaces com o usuário e dedicados a ambientes de projeto são no entanto obviamente altamente desejáveis, permanecendo esta como uma área de interesse no projeto, portanto. Na seção 7 foi mencionado o projeto de um UIMS dedicado à implementação dos editores gráficos do AMPLO [35].

Espera-se ainda que o novo *framework* venha a servir de plataforma para a integração de muitas outras ferramentas de projeto atualmente em desenvolvimento na UFRGS, e em particular aquelas desenvolvidas pelo Grupo de CAD para Sistemas Digitais, que visam as áreas de síntese automática, geração de testes, verificação formal e projeto visando testabilidade.

Referências

[1] F.R. Wagner. *Modelos de Representação e Gerência de Dados em Ambientes de Projeto de Sistemas Digitais.* Relatório Técnico CCR121, IBM Rio Scientific Center, Rio de Janeiro, 1991.

[2] F.R. Wagner, C. Freitas, and L.G. Golendziner. The AMPLO system – an integrated environment for digital systems design. In F.J. Rammig, editor, *IFIP Workshop on Tool Integration and Design Environments*, North-Holland, 1988.

[3] L.G. Golendziner and F.R. Wagner. Modeling digital systems as complex objects. In *9th International Symposium on Computer Hardware Description Languages and their Applications*, IFIP, 1989.

[4] F.R. Wagner, C.M.D.S. Freitas, and L.G. Golendziner. *Linguagens de Descrição de Hardware para Suporte à Integração do Processo de Projeto em AMPLO.* Relatório Técnico RP 065, CPGCC / UFRGS, Porto Alegre, 1987.

[5] J.F. Silva Filho, F.R. Wagner, and C. Le Faou. *LAÇO – Uma Linguagem para a Descrição de Hardware no Nível de Sistema.* Relatório Técnico RP 94, CPGCC / UFRGS, Porto Alegre, 1988.

[6] F.R. Wagner. *KAPA – Uma Linguagem para a Descrição de Hardware no Nível de Transferência entre Registradores.* Relatório Técnico RP 068, CPGCC / UFRGS, Porto Alegre, 1987.

[7] F.R. Wagner and C.M.D.S. Freitas. *NILO – Uma Linguagem para a Descrição de Hardware no Nível de Portas Lógicas.* Relatório Técnico RP 066, CPGCC / UFRGS, Porto Alegre, 1987.

[8] K. Becker and L.G. Golendziner. Database support for a CAD environment for digital systems design. In *8th International Conference on Computer Science*, SCCC, Santiago do Chile, 1988.

[9] S.D. Olabarriaga, M.S. Pinho, J.L.D. Comba, and C.M.D.S. Freitas. Ferramentas gráficas do sistema AMPLO. In *I Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens*, SBC, Petrópolis/RJ, 1988.

[10] C.M.D.S. Freitas, B. Copstein, and S.D. Olabarriaga. Ferramentas para especificação e controle da interface com o usuário no sistema AMPLO. In *II Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens*, SBC, Águas de Lindóia/SP, 1989.

[11] F.R. Wagner. Um ambiente integrado para simulação de sistemas digitais. In *IV Simpósio Brasileiro de Concepção de Circuitos Integrados*, SBC, Rio de Janeiro, 1989.

[12] F.R. Wagner. *Modelagem Multi-Nível no Ambiente AMPLO.* Relatório Técnico, CPGCC / UFRGS, Porto Alegre, 1991, a ser publicado.

[13] J.P. Figueiró. Simulador mestre para o sistema AMPLO. Departamento de Informática Aplicada, UFRGS, 1990. (Trabalho de Diplomação).

[14] P.R. Wagner. *Um Ambiente Integrado de Simulação de Sistemas Digitais.* Dissertação de Mestrado, CPGCC / UFRGS, 1991.

[15] F.R. Wagner et al. Recursos de gerência de projeto no sistema AMPLO. In *XVII Seminário Integrado de Software e Hardware*, SBC, Vitória, 1990.

[16] F.R. Wagner et al. Data management facilities in the AMPLO design framework. In F.J. Rammig and R. Waxman, editors, *2nd IFIP Workshop on Electronic Design Frameworks*, North-Holland, 1991.

[17] P.R.G. Luzzardi. *LAGO - A Interface Gráfica de Alto Nível do Sistema AMPLO.* Dissertação de Mestrado, CPGCC / UFRGS, 1991.

[18] F.R. Wagner. A multi-level digital systems simulator based on nets of agencies. In *Conference on Recent Advances in Simulation of Complex Systems*, JSST, 1986.

[19] J.H. Rech. *Um Simulador VHDL para o Sistema AMPLO.* Dissertação de Mestrado, CPGCC / UFRGS, 1991. (em andamento).

[20] F.R. Wagner. *Integrating a VHDL Dialect into the AMPLO Design Framework.* Relatório Técnico, CPGCC / UFRGS, Porto Alegre, 1991.

[21] J. Machado. *Um Modelo de Dados para Ambientes de Projeto.* Dissertação de Mestrado, CPGCC / UFRGS, 1990.

[22] K. Gottheil et al. The CADLAB workstation CWS – an open, generic system for tool integration. In F.J. Rammig, editor, *IFIP Workshop on Tool Integration and Design Environments*, North-Holland, 1988.

[23] P. van der Wolf et al. Data management for VLSI design: conceptual modeling, tool integration, and user interface. In F.J. Rammig, editor, *IFIP Workshop on Tool Integration and Design Environments*, North-Holland, 1988.

[24] E.B. de la Quintana, G.O. Annarumma, and P. Molinari Neto. *GARDEN – the Design Data Interface.* Relatório Técnico CCR-107, IBM Rio Scientific Center, Rio de Janeiro, 1990.

[25] F.R. Wagner and A.H. Viegas de Lima. Design version management in the GARDEN framework. In *28th Design Automation Conference*, ACM/IEEE, 1991.

[26] S.D. Olabarriaga, M.S. Pinho, and J.L.D. Comba. *Interface de Saída com Dispositivos Gráficos.* Relatório Técnico RP 079, CPGCC / UFRGS, Porto Alegre, 1987.

[27] S.D. Olabarriaga and J.L.D. Comba. *Extensão da Interface de Saída com Dispositivos Gráficos.* Relatório Técnico RP 084, CPGCC / UFRGS, Porto Alegre, 1987.

[28] S.D. Olabarriaga. *Interface com Dispositivos de Entrada Gráfica.* Relatório Técnico RP 083, CPGCC / UFRGS, Porto Alegre, 1987.

[29] J.M. de Sá. *Interface de Entrada para Tecladoe e Mouse.* Relatório Técnico RP no prelo, CPGCC / UFRGS, Porto Alegre, 1991.

[30] M.M. Oliveira Neto. *Um Estudo sobre Transações de Projeto e sua Aplicação no Sistema AMPLO.* Trabalho Individual TI 152, CPGCC / UFRGS, Porto Alegre, 1990.

[31] M.A.C. Livi. *Um Modelo de Cooperação para o Sistema AMPLO.* Dissertação de Mestrado, CPGCC / UFRGS, 1991. (em andamento).

[32] M.L. Bushnell and S.W. Director. VLSI CAD tool integration using the Ulysses environment. In *23rd Design Automation Conference*, ACM/IEEE, 1986.

[33] J. Daniell and S.W. Director. An object-oriented approach to CAD tool control within a design framework. In *26th Design Automation Conference*, ACM/IEEE, 1989.

[34] D.W. Knapp and A.C. Parker. A design utility manager: the ADAM planning engine. In *23rd Design Automation Conference*, ACM/IEEE, 1986.

[35] J.D.G. Ramos, F.R. Wagner, and C.M.D.S. Freitas. Um UIMS para o sistema AMPLO. In *XVII Seminário Integrado de Software e Hardware*, SBC, Vitória, 1990.

[36] F.R. Wagner. *Gerência de Metodologias de Projeto no Ambiente GARDEN.* Relatório Técnico, IBM Rio Scientific Center, Rio de Janeiro, 1991. a ser publicado.

[37] S. Dessloch, T. Haerder, N. Mattos, and B. Mitschang. KRISYS: KBMS support for better CAD systems. In *2nd International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, Gaithersburg, MA, 1989.

RP-137: "Integrating a VHDL Dialect into the AMPLO Design Framework", dezembro 1990.
F.R. WAGNER

RP-136: "O Ambiente de Execução do Experimento em Programação Diversitária", novembro 1990.
R. CANANI; S.P. ZANI

RP-135: "Teste Prático do Circuito MPC e o seu Ambiente Técnico", novembro, 1990.
L. ROISENBERG; T.V. WAGNER; D.A.C. BARONE

RP-134: "SÍNTESE AUTOMÁTICA DE PARTES OPERATIVAS - Ferramentas resultantes da implementação do protótipo do sintetizador automático de partes operativas (SAPO), agosto 1990.
C. De ROSE; J.L.S. JÚNIOR

RP-133: "Preliminar Thoughts on Agents and their Development", novembro 1990.
A.C.R. COSTA

RP-132: "Towards a Complete Conceptual model: Petri Nets and Entity-Relationship", agosto 1990.
C.A. HEUSER; E.M. PERES

RP-131: "PC como Terminal do ED680 (Uso e Implementação)", agosto 1990.
S.D. OLABARRIAGA

RP-130: "Ferramenta para Apoio à Análise de Requisitos e Modelagem de Sistemas de Banco de Dados", julho 1990.
M.H. YAMAGUTI; S. LOH; J.M.V. CASTILHO

RP-129: "Biblioteca de Células Tranca. Regras CMP - Parte II", julho 1990.
A. REIS; E. E. GIFFONI; F. MORAES; R. REIS

RP-128: "Nets of Places and Links: a coherent presentation of Petri Nets for systems modeling", julho 1990.
G. RICHTER; C.A. HEUSER

RP-127: "Integração do Método T.A.& A. e do Ambiente YPY na Especificação e Prototipação de um SIE", junho 1990.
N. EDELWEISS; J.P.M. OLIVEIRA

RP-126: "Uma Abordagem para Prototipação de Sistemas de Banco de Dados", junho 1990.
H. STROGULSKI; J.M. CASTILHO; O.J.C. SOUZA

RP-125: "Edição Revisada da Biblioteca de Células Tranca", maio 1990.
F.G. MORAES; A.I. REIS; E.E. GIFFONI; M.S.LUBASZEWSKI; R.F. GOMES; R.A.L. REIS

Relatórios de Pesquisa

RP-149: "Integração de Ferramentas no Sistema AMPLO: Crítica e Proposta de Extensões", março, 1991.
F.R. WAGNER.

RP-148: "Interface de Entrada para Teclado e Mouse", março, 1991.
J.M. DE SÁ.

RP-147: "Servidores - Guia do Usuário - Edição 1", janeiro, 1991.
A.R. TREVISAN, C. LEYEN, G. CAVALHEIRO, J.F.L. SCHRAMM, L.G. FERNANDES, P. FERNANDES, R.M. BARRETO, R. TEODOROWITSCH

RP-146: "Biblioteca de Células TRANCA regras ECP15/1", janeiro, 1991.
C. CRUSIUS, L. FICHMAN, M. KINDEL, C. MARCON, R. REIS

RP-145: "Manual do Usuário do Projeto TRANCA; v 1.0", janeiro 1991.
F.G. MORAES; M. LUBASZEWSKI, R.A.L. REIS

RP-144: "Manual do Sistema TRAMO Projeto TRANCA versão 1.0", janeiro 1991.
M.A. SOTILLE; C.E.S. SOUZA; M.G.R. ARAUJO; M. LUBASZEWSKI; R.A.L. REIS

RP-143: "PILCHA: Projeto e Implementação de um Sistema Digital Discreto dedicado ao controle de acesso direto a memória", janeiro 1991.
F. AZEREDO; L. ROISENBERG; D.A.C. BARONE

RP-142: "Ambiente para Estudo de Fractais - Relatório de Projeto", janeiro 1991.
S.D. OLABARRIAGA; F.S. MONTENEGRO

RP-141: "Técnicas para Compilação em Paralelo", janeiro 1991.
L. C. GARCIA

RP-140: "Interval Solution and Safe Starting region for nonlinear systems", janeiro 1991.
M.A. CAMPOS

RP-139: "Programação de Máquinas Pipeline Vetoriais: Compiladores, Linguagens e Estado-da-Arte, janeiro 1991.
R.M. BARRETO.

RP-138: "TAURA, Um ASIC para Terminal de Video", janeiro 1991.
J.L.A. GUNTZEL; L.O.S. FREIRE; R.P. RIBAS, D.A.C. BARONE.