ISSN0104-4869

REVISTA DO INSTITUTO DE

INFORMÁTICA

PUC-CAMPINAS - VOLUME 5 - Nº 2 JULHO/DEZEMBRO/1997

CONTEÚDO/CONTENTS

Artigos/Articles

Comunicações Institucionais/Institutional Communications

PUC CAMPINAS

PONTIFÍCIA UNIVERSIDADE CATÓLICA

Revista do Instituto de Informática da PUC-Campinas. -
Volume 5, n.2 (1997) - . Campinas:
Semestral
1. Informática-Periódico

CDD 001.61
CDU 681.3

Revista do Instituto de Informática da PUCCAMP

Publicação Semestral

Editor-Executivo: Dr. Frank Herman Behrens

Capa
Máscara de circuito integrado cedida pela FCTI - Fundação Centro Tecnológico para Informática.

Correspondência:
A/C:
Instituto de Informática - PUC-Campinas
Caixa Postal 317 - Campus I
Rodovia D. Pedro I - Km 136
CEP:13020-904 - Campinas - SP
FAX: (0XX19) 756-7094
E-mail: ii@zeus.puccamp.br
ou frank@zeus.puccamp.br

A "Revista do Instituto de Informática" tem uma tiragem de 2000 exemplares. É distribuída gratuitamente às Universidades, Centros de Pesquisa, Órgãos Governamentais e Empresas que nos solicitam.

Composição e Impressão
Gráfica PUC-Campinas

ISSN 0104 - 4869

PUC CAMPINAS
PONTIFÍCIA UNIVERSIDADE CATÓLICA

INSTITUTO DE INFORMÁTICA

REVISTA DO INSTITUTO DE INFORMÁTICA DA PUC-CAMPINAS

Rev. Inst. Inform. PUC-Campinas	V. 5	n. 2	p. 1-40	julho/dezembro, 1997

CONTEÚDO/CONTENTS

Artigos/Articles

Comunicações Institucionais/Institutional Communications

EDITORIAL

Neste número, a **Revista do Instituto de Informática** apresenta aos leitores artigos técnicos em distintas áreas da informática, abrangendo aspectos de *hardware*, *software* e educação.

No primeiro artigo deste número, "Projeto de Sistemas de Computação", o autor enfoca etapas de planejamento, desenvolvimento, avaliação e revisão de um ciclo de projeto integrado de sistemas de computação, sob a óptica do *hardware*, discutindo uma metodologia mais atual para abordar este problema, bem como apresentando uma excelente revisão de pontos essenciais a serem considerados em atividades de desenvolvimento nesta área de conhecimento.

Apresentamos aos leitores, também, o artigo "Abordagens ao Processamento Simbólico da Linguagem Natural", trabalho que discute aspectos lingüísticos relevantes a serem considerados na solução do problema do processamento e reconhecimento de linguagens naturais por parte de sistemas computacionais, através da aplicação de três possíveis abordagens simbólicas, baseadas em regras, em princípios e em casos.

Por fim, em nossa seção de Comunicações Institucionais, o artigo, "Educação à Distância Mediada por Computador (EDMC) – Projeto Pedagógico para uma Rede de Pós-graduação", apresenta e discute a potencialidade de aplicação de EDMC em cursos de Pós-graduação, tendo como base a experiência preliminar do Curso de Mestrado em Informática com ênfase no Gerenciamento de Sistema de Informação, ora oferecido por nossa Instituição, tradicionalmente na forma de curso presencial, mas também com a opção de curso à distância, com aulas virtuais segundo esta abordagem.

Finalmente, agradecemos a confiança e a colaboração de todos os participantes do Conselho Editorial e do Conselho Consultivo, bem como daqueles que direta ou indiretamente auxiliaram-nos nesta edição.

Esperamos cada vez mais apresentar à comunidade profissional e acadêmica de Informática um veículo útil à troca de idéias e à disseminação de conhecimentos.

Frank Behrens
Editor-executivo

PROJETO DE SISTEMAS DE COMPUTAÇÃO

COMPUTER SYSTEM DESIGN

José Gonzaga SOUZA JÚNIOR[*]

RESUMO

O projeto de sistemas de computação na atualidade é baseado na aplicação de técnicas, metodologias e conhecimento oriundos das áreas de software, arquitetura e hardware os quais, longe de formar um conjunto coeso, apresentam um carater heterogêneo facilmente visível, seja nas divisões internas entre as diversas metodologias de projeto para uma mesma área, seja no carater estanque de cada área. Entretanto, com o advento dos processadores RISC e a conseqüente mudança no paradígma de projeto arquitetural, cresce a demanda pela integração entre as áreas, fato que já é realidade para arquitetura e hardware como comprovam os recentes projetos de processadores VLSI, mas que tende rapidamente a envolver também o projeto dos sistemas de software associados. Duas razões despontam como fortes inibidoras desta integração: o enfoque de projeto evolucionário, dominante para sistemas de software, que inviabiliza um ciclo de projeto integrado com as demais áreas onde as metodologias baseiam-se em um enfoque descendente e; a própria inexistência de um paradígma sedimentado para o projeto arquitetural, hoje bastante influenciado pela metodologia de síntese vertical preponderante em projetos de hardware. Este trabalho identifica as principais diferenças entre as metodologias dominantes em cada área e propõe as condições básicas para o estabelecimento de um enfoque homogêneo de projeto, fator essencial à obtenção futura de um ambiente integrado de projeto.

Palavras-chave: sistemas de computação, projeto de software, projeto arquitetural de sistemas de computação, síntese vertical de hardware, co-projeto de hardware e software

1. INTRODUÇÃO

A evolução dos computadores teve como núcleo e catalisador, a arquitetura de von Neumann, em função da qual se desenvolveram e foram consolidadas as áreas de software e hardware, hoje camadas efetivas dentro da estrutura hierárquica dos sistemas de computação. O surgimento de opções à máquina de von Neumann propiciou o amadurecimento da área de arquitetura, que atualmente conta com um amplo espectro de modelos computacionais, alguns dos quais testados em protótipos de laboratório [1,2]. Entretanto, a grande maioria dos computadores tem sido concebida aplicando-se a metodologia de projeto sumariada em Gurd et al [3], que preconiza o projeto de sistemas de computação em três níveis que se complementariam: o "nível de máquina", correspondendo ao nível do conjunto de instruções; o "nível da linguagem", definido pelo escopo das lingua-

(*) Professor do Instituto de Informática da PUC-Campinas e Pesquisador da Fundação Centro Tecnológico para Informática - CTI (gonzaga@im.cti.br).

gens de programação e; o "nível da aplicação", orientado para domínios de aplicação específicos.

Este enfoque, embora adequado para a validação conceitual de soluções, não explora de forma eficiente o espaço de projeto[1] existente na atualidade [4] propiciado pela variedade de tecnologias de hardware e software disponíveis, estando sustentado primariamente no aumento de desempenho do hardware, fruto da crescente velocidade e compactação dos circuitos integrados. A demanda por maior integração por sua vez ocasionou um brutal aumento na complexidade dos circuitos[2], como resultado direto das dificuldades de empacotamento geradas pela redução do ciclo de relógio, culminando no desenvolvimento dos processadores VLSI. Com os processadores VLSI[3], formados por poucos tipos de elementos funcionais, com caminhos de dados e de controle simples e regulares e uso extensivo de "pipelining" [5,6,7], caracterizou-se a mudança no paradígma original de projeto arquitetural, com o estabelecimento de uma simbiose entre arquitetura e implementação, na forma sumariada por Hennessy [8]:

1. *Preferência pelo uso de estruturas paralelas.* "Pipelining" e arranjos paralelos de múltiplas unidades funcionais são usados para remediar a menor velocidade de operação dos circuitos MOS. A maior adequação da tecnologia para a replicação de estruturas regulares também penaliza arquiteturas formadas por módulos complexos.
2. *Minimização da comunicação.* O custo de implementação é bem maior para arquiteturas que requeiram alto grau de comunicação dos módulos internos entre si e com o exterior. Em particular busca-se concentrar o tráfego de alta velocidade nos limites da pastilha devido a largura de banda da comunicação extra-integrado ser muito menor e mais susceptível a interferências e atrasos de propagação.

A integração entre os projetos arquitetural e de hardware permite uma melhor exploração do espaço de projeto dos sistemas de computação, tendo entretanto como limitante a pouca interação com o projeto dos sistemas de software associados que acarreta efeitos negativos no desenvolvimento dos sistemas como um todo [9].[4] No item a seguir é apresentada uma panorâmica das principais metodologias de projeto adotadas para software, hardware e arquitetura e no item 3 são analisadas as condições para que estas metodologias evoluam para um cenário de maior integração.

2. METODOLOGIAS DE PROJETO

De forma geral, o projeto de sistemas de computação assistido por computador está, como resposta ao nível de complexidade atingido, concentrando-se no domínio da descrição do problema em contra-posição ao enfoque tradicional, orientado para a definição da solução, estratégia que tem garantido a tratabilidade computacional dos problemas, dentro de um cenário de complexidade crescente. Para projetos de sistemas de software isto implica em uma demanda por qualidade e produtividade com a conseqüente pesquisa em métricas e padrões. Nos projetos de hardware o objetivo é a síntese automática de circuitos integrados, feita a partir de uma descrição comportamental, e que gere soluções não inferiores [10][5] para um dado conjunto de especificações [11]. Para que seja viável, esta filosofia de projeto não prescinde do desenvolvimento de metodologias de projeto as quais encontram-se em estágios distintos de amadurecimento de acordo com a área e o nível de abstração envolvidos, como visto a seguir.

(1) Espaço euclidiano cujos eixos correspondem ao conjunto definido como métrica da qualidade do projeto.

(2) A complexidade sendo função da dimensão mínima e também do tamanho do integrado [28].

(3) O desbalanceamento inicial entre os avanços na tecnologia para fabricação de circuitos integrados e a capacidade de uso eficiente da área em silício, desembocou na chamada "crise do VLSI" nos anos 70 que levou alguns pesquisadores a preverem um estrangulamento iminente dos computadores sequenciais por conta da impossibilidade de obtenção de mais desempenho, fato que só foi superado a partir da metodologia de projeto proposta por Mead e Conway, embora continue válido o argumento de que, cedo ou tarde, tornar-se-á impossível sustentar o aumento de desempenho com base na redução constante do ciclo de relógio dos computadores, em uma dada tecnologia de integração.

(4) Estatísticas existentes para sistemas embutidos demonstram que embora 90% dos circuitos integrados para aplicações específicas sejam projetados corretamente apenas 50% do sistemas finais, compostos por hardware e software, têm a funcionalidade esperada.

(5) Uma solução não inferior é aquela que sem ser ótima, não pode entretanto ser superada por nenhuma outra desde que cada melhoramento sob o ponto de vista de algum dos critérios de sua métrica, provoque degradação ao menos em relação a algum outro critério. Em geral há um infinito número de pontos que compõem uma solução não inferior, definindo uma "superfície não inferior", dentro do espaço de projeto.

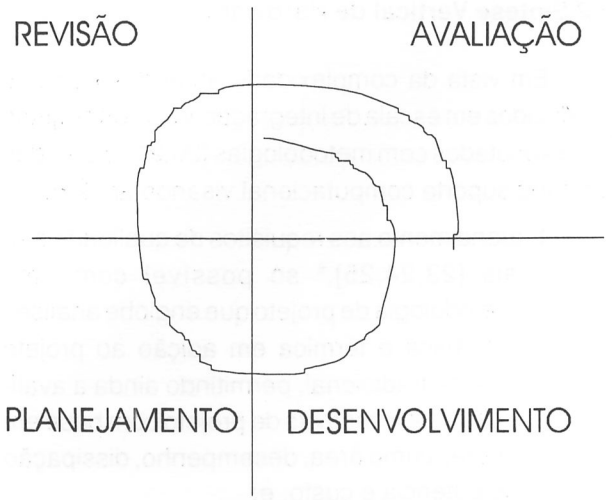

Figura 1: Metodologia de Projeto em Espiral. Proposta em 1986 por Boehm, esta metodologia se opõe ao modelo clássico de desenvolvimento em cascata, por introduzir a análise de riscos no ciclo de projeto e por romper com o clássico enfoque descendente (top-down) de desenvolvimento de software. Como visto no diagrama, Boehm modela o processo por meio de quatro etapas cíclicas: No quadrante denominado revisão, é feita a determinação de objetivos, identificação de restrições e a análise de alternativas; o quadrante de avaliação comporta a análise de riscos e interação com o usuário; o quadrante do desenvolvimento contempla a geração do nível seguinte de refinamento da solução e; o quadrante de planejamento contém as atividades de definição de novas metas. O processo de produção de software é feito de modo incremental com cada passagem pelo conjunto de quadrantes gerando uma versão mais refinada da solução, ou o abandono do projeto.

2.1 Projeto Integrado de Software

O projeto de sistemas de software encontra-se em transformação, migrando de uma forma linear de pensamento para uma visão iterativa ou evolucionária do software [12].

2.1.1 Modelos Sequenciais

O paradígma de engenharia de software dominante ainda é sequencial sendo os sistemas desenvolvidos segundo modelos de processo em que séries de passos sequenciais tratam a complexidade dos problemas manipulando-os por partes. O modelo sequencial mais antigo, conhecido como "codificação-e-correção" e usado nos primórdios da computação, é puramente empírico, com as atividades de depuração e teste sendo planejadas e executadas após a codificação ter sido feita [13]. O crescimento dos programas exigiu um planejamento prévio com a conseqüente caracterização do desenvolvimento de software na forma de um ciclo representado pelo "modelo em cascata" [14] proposto por Royce [15] em 1970, onde o ciclo de produção é dividido em estágios correspondendo às atividades de análise de viabilidade, especificação de requisitos, projeto, codificação, integração, testes e manutenção, sendo previstas realimentações entre estágios adjacentes.

2.1.2 Modelos Iterativos

Nesta categoria encontram-se os modelos baseados em linguagens de 4ª geração, prototipagem rápida e o modelo da espiral [16,13], os quais podem ser usados também de forma combinada [12]. Em qualquer dos casos, faz-se uso extensivo de ambientes integrados de projeto (I-CASE),[6] cuja arquitetura básica é vista a seguir, como forma de obtenção de ganhos em produtividade e qualidade pela exploração de macro-aspectos da metodologia de projeto como o uso de repositórios de software e prototipagem rápida [17,12].

Como já notado, a principal mudança diz respeito ao abandono da forma sequencial de projeto caracterizada pelo modelo de produção de software em cascata, substituindo-o por modelos iterativos ou evolucionários. A essência desta mudança é a troca da mecânica de projeto orientada pela especificação por outra dirigida para a identificação e prototipagem dos "elementos de risco" do sistema. Pressman [12] sumaria as tendências de mudança através de uma proposta de ambiente para desenvolvimento de software construído a partir do modelo de espiral proposto por Boehm, comentado na Figura I, e suportado por tecnologias integradas de prototipagem e orientação a objeto.

Dentro do diagrama em espiral, o desenvolvimento do projeto seria feito com base no paradígma de orientação a objeto, o qual oferece como vantagens sobre os demais, a facilidade de derivação de componentes e o re-uso de software. A etapa de avaliação, como é inerente ao modelo proposto por Boehm, faria uso extensivo dos recursos de

(6) "Integrated Computer-Aided Software Engineering".

prototipagem, seja com o objetivo experimental, exploratório ou mesmo estrutural [18].[7]

2.1.2.1 Arquitetura de um Ambiente de I-CASE

Exceto por detalhes de implementação, os ambientes de I-CASE seguem a arquitetura básica vista e comentada na figura. São essenciais para a operação destes ambientes a presença de mecanismos de execução, através dos quais é possível o disparo, suspensão ou término de processos; e de mecanismo de comunicação, responsáveis pela gerência da comunicação entre processos, permitindo o uso integrado das ferramentas. Em geral, o ambiente é organizado de forma distribuída, com base em um sistema operacional multi-tarefas sob o qual várias ferramentas podem estar ativas concomitantemente. O elemento central do ambiente é o repositório de software [19], um banco de dados relacional ou orientado a objeto ao qual estão associados serviços exclusivos para CASE, como acomodação de tipos de dados complexos, modelos para a imposição de integridade, interfaces ricas em semântica, entre outros.

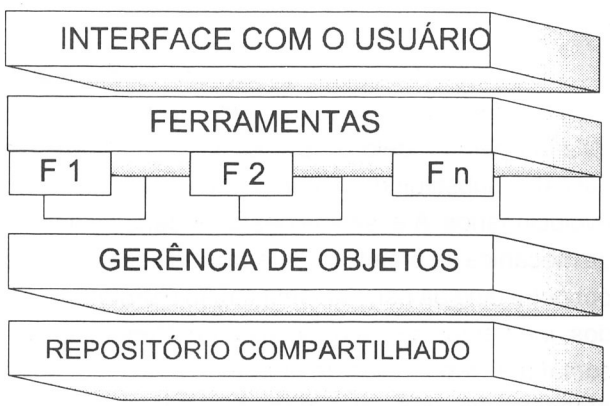

Figura 2: Arquitetura de um Ambiente de I-CASE. A estrutura em camadas é formada por ferramentas de interface homem-máquina baseadas em um protocolo de apresentação comum; por um conjunto de ferramentas de CASE suportadas por serviços de gerência de ferramentas comuns; por uma camada responsável pelas funções de gerência de configuração e; por um repositório de software contendo a base de dados do ambiente.

2.2 Síntese Vertical de Hardware

Em vista da complexidade atual dos circuitos produzidos em escala de integração VLSI, os projetos são executados com metodologias [20,21] baseadas em forte suporte computacional visando ao [22]:

1. atendimento aos requisitos de qualidade atuais [23,24,25],[8] só possível com uma metodologia de projeto que englobe análises mecânica e térmica em adição ao projeto elétrico tradicional, permitindo ainda a avaliação de alternativas de projeto com base em fatores como área, desempenho, dissipação de potência e custo, e;
2. manutenção do ciclo de projeto dentro de limites de tempo condizentes com as expectativas de um mercado altamente competitivo [20,26].[9]

A automação de projeto em hardware é a capacidade de um sistema de projeto de perfazer as atividades de síntese de uma forma autônoma e correta por construção, entre um nível do processo de projeto para outro [27,28].[10] Síntese refere-se, em termos gerais, ao processo de transformar uma descrição de projeto em um dado nível de abstração para uma representação equivalente de nível mais baixo [22], acrescentando-se detalhes estruturais e geométricos, com o objetivo de aproximar-se da realização física do sistema [29]. Os diversos níveis em que o processo de síntese é aplicado no projeto de hardware são sumariados a seguir.

2.2.1 Síntese ao Nível do Sistema

Etapa de síntese aplicável quando, usando-se a capacidade atual de integração, sistemas inteiros são acondicionados em uma única pastilha. Neste caso as técnicas de síntese funcional nem sempre são

(7) Riddle e Williams definem três tipos de prototipagem: o primeiro, chamado estrutural ou evolucionário, teria como meta a criação do software por meio de implementações de complexidade sucessiva, que agreguem funcionalidade; o segundo tipo definido como experimental teria objetivo investigatório, permitindo a comparação entre diferentes implementações; por fim o tipo exploratório seria adotado como apoio para a identificacão e formulação do problema.

(8) Ao contrário do que se poderia supor em um primeiro momento, a sistematização do projeto visando atingir níveis de qualidade profissional é foco de intensa pesquisa acadêmica, notadamente em hardware.

(9) Dado o nível de integração atual se mantidos os padrões e ferramentas do início da década de 80, o tempo de proieto seria comparável ao tempo de vida do produto com um circuito integrado de 10^5 transitores demandando 60 homens/ano para o projeto e igual quantidade para a depuração. Extrapolando-se para um integrado com 10^7 transistores o esforço, apenas no projeto, seria de $6 \cdot 10^3$ h/a.

(10) Para a maioria dos métodos de síntese tratados neste item, valeria ainda a definição mais restrita de McFarland et al: "A tarefa de síntese é, a partir da especificação do comportamento requerido de um sistema, e de um conjunto de restrições e objetivos a serem atingidos, encontrar uma estrutura que implemente este comportamento, satisfazendo ainda as restrições e os objetivos".

adequadas ao nível de abstração da descrição comportamental de entrada por não enfocar questões como por exemplo, o desmembramento do sistema em processos assíncronos, o redimensionamento do bloco operacional para variação do grau de paralelismo, e o agrupamento de registradores isolados em "register files" [27]. Este nível de síntese pode ser também usada nos casos de sistemas implementados em mais de um integrado, que sejam gerados a partir de uma única descrição precedendo, em ambos os casos, à etapa de síntese funcional. A síntese ao nível do sistema como descrita por Lagnese e Thomas [31] determina, a partir de uma descrição comportamental, e segundo as etapas descritas a seguir, as características de projeto necessárias à síntese funcional, como o número de integrados ou módulos, e o tráfego global entre estes.

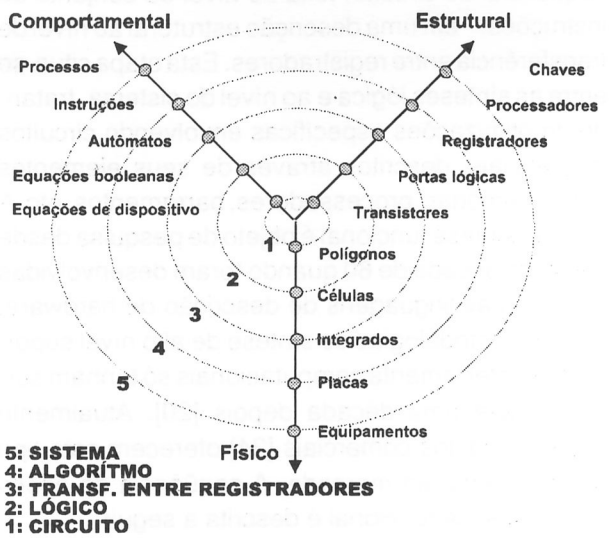

Figura 3: Diagrama de Gajski [26,1]. Em 1983 Gajski e Kuhn propuseram o esquema de representação posteriormente denominado diagrama de Gajski ou diagrama **Y**, que contempla os três eixos ortogonais usados para a representação de sistemas digitais, correspondendo aos domínios comportamental, estrutural e físico. A partir do centro, cada eixo cresce em abstração, podendo ser identificados pontos de mesmo Nível de abstração nos três eixos correspondendo a níveis de abstração definidos, embora estes permitam um "continuum" de representação [29].

2.2.1.1 Partição Arquitetural

Uma tarefa essencial que corresponde a determinação das relações de dados e de controle relevantes para a extração da estrutura implícita no comportamento do sistema, com o objetivo de:

1. otimização da área através da redução do hardware usado em cada partição;
2. aumento do desempenho pela redução do tráfego entre partições, obtido com o agrupamento das partes da arquitetura que se interagem mais pesadamente. Adicionalmente pode haver também uma otimização em área se houver redução nos dutos de comunicação;
3. detecção de paralelismo e concorrência ao nível do sistema, facilitando o escalonamento na etapa de síntese funcional e;
4. introdução de considerações físicas e estruturais precocemente no projeto, adotando assim um metodologia mista, não puramente ascendente ou descendente.

2.2.1.2 Síntese

Como nas demais etapas de síntese, a metodologia de síntese ao nível do sistema aplica métodos de otimização por critério múltiplo, neste caso baseadas na técnica de "agrupamento em múltiplos estágios".[11] As técnicas de agrupamento agregam objetos em conjuntos segundo alguma métrica de proximidade a qual, aplicada iterativamente, produz uma estrutura hierárquica em árvore para a disposição dos objetos, permitindo então que seja usado um critério de corte para a determinação dos grupos. A técnica de agrupamento em múltiplos estágios é o uso combinado das técnicas de agrupamento com a adoção de métricas de proximidade diferentes e desacopladas entre si, para facilitar sua tratabilidade computacional, e evitar a propagação de erros devido a aproximações e estimativas feitas em uma dada métrica. As métricas de agrupamento adotadas por Lagnese e Thomas são derivadas dos seguinte tipos básicos:

- *Transferência de controle.* Visa a redução da passagem de controle entre grupos, buscando agrupar-se os operadores em fluxos contínuos, de acordo com a probabilidade **P** de ativação sequencial destes operadores, como no exemplo da equação 1, onde $\mathbf{P}_{AB}$ repre-

(11) Adaptação para o português da expressão inglesa "multi-stage clustering".

senta a probabilidade de ativação sequencial dos grupos de operadores **A** e posteriormente **B**.

$P_{AB} = P_B / P_A$ (equação 1)

- *Transferência de dados*. Tem como meta reduzir o fluxo de dados entre grupos o que é feito com base na definição de grupos que tenham o valor mínimo para a razão $\mathbf{G}_{dado}$ que corresponde a fração de dados comuns entre grupos, como no exemplo da equação 2, válida para dois grupos de operadores **A** e **B**:

$$G_{DADO\_AB} = \frac{Comum(A, B)}{Total_A + Total_B} \quad \text{(equação 2)}$$

- *Redundância de Hardware*. Estágio executado após os agrupamentos por dado e controle, consistindo na redução do hardware pelo compartilhamento de partes do bloco operacional usadas por grupos de operadores similares, quais sejam, aqueles grupos que contenham os mesmos tipos de operadores e não sejam simultaneamente acionados.

Para a árvore obtida após a aplicação de cada estágio, a determinação da linha de corte faz uso de considerações físicas fornecidas adicionalmente à descrição comportamental. Os critérios de corte são baseados em considerações de área, conexões de dados e escalonamento.

- *Área*. Este critério tem como objetivo produzir agrupamentos dentro de limites máximos e mínimos de área, evitando a produção de objetos com dimensões por demais díspares, o que dificulta o leiaute automático posterior. É também usada informações do projetista relativas a área dos objetos para uma dada tecnologia-alvo.
- *Conexões*. Tem como meta a redução das interconexões entre grupos de operadores, através do que se obtém uma redução entre as interconexões das partições.[12]
- *Escalonamento*. Neste critério são introduzidas considerações de tempo, através da limitação pelo projetista do comprimento máximo das seqüências de operadores presentes em um dado grupo. Como restrição adicional é imposta a existência de apenas uma unidade funcional de cada tipo dentro da seqüência.

2.2.2 Síntese Funcional

A síntese funcional de circuitos digitais, também chamada síntese de alto nível [27,29,32,33], é a transformação automática de uma especificação comportamental em linguagem de descrição sequencial do circuito, feita ao nível do conjunto de instruções,[13] em uma descrição estrutural ao nível de transferência entre registradores. Esta etapa situa-se entre as sínteses lógica e ao nível do sistema, tratando de otimizações específicas envolvendo circuitos sequenciais, descritos através de seus elementos como memórias, processadores, barramentos, etc. A área de síntese funcional é objeto de pesquisa desde o final da década de 60 quando foram desenvolvidas as primeiras linguagens de descrição de hardware, embora metodologias de síntese de alto nível suportadas por ferramenta computacionais só tenham surgido quase uma década depois [20]. Atualmente alguns produtos comerciais [34] oferecem este tipo de ferramenta ao mercado. A seqüência de ações para a síntese funcional é descrita a seguir:

2.2.2.1 Compilação

Compilação da descrição comportamental, convertendo-a em um grafo orientado, representando o fluxo de dados e de controle.[14] Durante a compilação, além das otimizações de código comuns em software, são tomadas ações específicas orientadas para a

(12) Apenas a descrição comportamental não permite a aplicação deste critério, pois como também notado por Lagnese e Thomas [31], não há o mapeamento um-a-um entre o fluxo de dados na descrição comportamental ao nível do sistema e os dutos na implementação em hardware, o que faz este critério mais fraco que os demais.

(13) Chamado em alguns textos como nível algorítmico. A nomenclatura adotada neste texto é um pouco mais específica, e leva em conta o contexto particular do projeto de sistemas de computação, sendo consistente com as convenções encontradas na literatura especializada.

(14) As informações dados e de controle são geradas em grafos separados em alguns sistemas [32].

implementação final em hardware como aumento do paralelismo sempre que não houver dependência de dados, e redução do número de níveis no grafo.

2.2.2.1.1 Escalonamento

Escalonamento no tempo das operações descritas no grafo associando cada operação a um passo de controle, de forma que uma unidade de hardware só possa ser acionada uma vez em cada passo.[15] As ações de escalonamento e alocação convertem um comportamento em uma representação estrutural e podem ser aplicadas de forma intercambiável. Se, por exemplo, o bloco operacional for alocado primeiro, a partir deste é feito o escalonamento das operações.

2.2.2.1.2 Alocação

A alocação corresponde a atribuição de hardware para os elementos do grafo: unidades funcionais para a execução das operações, unidades de armazenamento para as variáveis, e barramentos para a comunicação.

2.2.2.1.3 Geração do Bloco de Controle

Geração da lógica de controle com base no escalonamento das operações no bloco operacional resultante da alocação.

2.2.3 Síntese Lógica

A síntese de funções lógicas tem como objetivo a implementação em hardware de uma dada rede de circuitos lógicos combinatórios, expressos originalmente através de equações booleanas, tabelas, máquinas de estado algorítmicas, ou linguagens de descrição de hardware, entre outros. Os algorítmos de minimização baseados na teoria convencional de chaveamento de circuitos, e orientados para a obtenção da solução ótima em termos do número de portas, conduzem a soluções do tipo NP-completa [35].[16] não sendo adequados para a implementação em computador tendo em vista o rápido aumento da complexidade destes algorítmos a medida que o número de variáveis de entrada cresce. A síntese lógica assistida por computador se faz necessária devido a dois fatores associados:

1. possibilidade tecnológica de integração de grande quantidade de lógica em um único circuito integrado, permitindo com isto o aumento da velocidade de operação dos circuitos e;
2. a conseqüente complexidade das fases de minimização, simulação e teste da rede lógica gerada. Deste modo, a síntese automática de um circuito correto por construção e que, mesmo sem ser o mínimo possível, atenda aos requisitos especificados apresenta-se como vantajosa [36].

Tomando-se como ponto de partida um conjunto de funções booleanas a tarefa de síntese consistiria na sua decomposição em sub-funções com um número limitado de variáveis, seguido pelo agrupamento destas através das variáveis comuns em blocos básicos, com os critérios de agrupamento visando minimizar os caminhos críticos e o número total de blocos [37,38,39], buscando assim obter soluções que apresentem atraso mínimo, área mínima ou uma combinação de ambos [11].

2.2.3.1 Síntese em Dois Níveis

As técnicas de síntese lógica são divididas em síntese de dois níveis e multi-nível. A minimização lógica em dois níveis produz uma expressão booleana formada por somas-de-produtos,[17] sendo orientada para a implementação de funções de controle em PLAs,[18] onde a minimização do número de mintermos corresponde a solução de menor área.

(15) Desta forma, a síntese funcional produz um circuito final síncrono, com cada passo de controle correspondendo a um período de relógio.

(16) Em relação ao tempo de execução, os algoritmos para a solução de um problema podem ser de "tempo exponencial", onde o tempo de execução é uma função exponencial do número de entradas; ou de "tempo polinomial", onde o tempo de execução é uma função polinomial do número de variáveis. Problemas com solução polinomial conhecida são ditos de classe **P**. Estes são um sub-conjunto dos problemas que, a princípio, têm uma solução polinomial embora só a solução exponencial seja conhecida, chamados de problemas classe **NP**. Dentro da classe NP existem alguns problemas gerais, aos quais os demais podem ser reduzidos, chamados de **NP-Completos**, e que não apresentam solução ótima.

(17) Na soma-de-produtos cada parcela da expressão representa um ponto onde a função assume o valor lógico 1.

(18) "PLA- programmed logic array". Estrutura usada para o mapeamento de lógica combinatória irregular. Apresenta nas suas saídas a forma canônica da soma-de-produtos das entradas.

2.2.3.2 Síntese Multi-Nível

A síntese lógica multi-nível por sua vez representa uma forma mais geral de solução possibilitando o re-uso de blocos de lógica comuns e permitindo resolver problemas de feixe-de-entrada e de saída [40],[19] bem como de potenciais "circuit hazards" decorrentes da solução em dois níveis [36]. A síntese lógica multi-nível é foco de intensa pesquisa na atualidade [38,41], mas em termos gerais os diversos métodos de síntese obedecem a seqüência de passos:

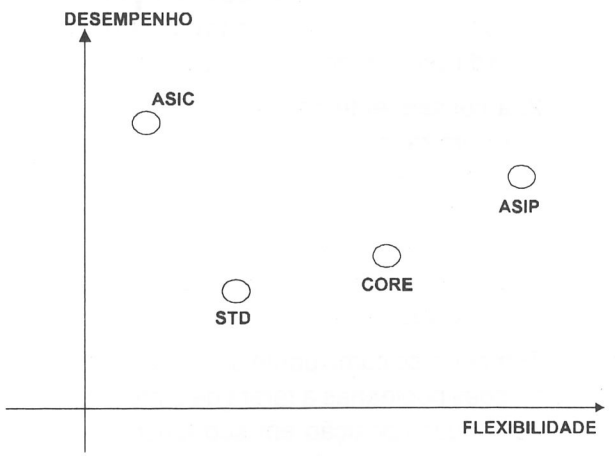

Figura 4: Espaço de projeto de processadores dentro da metodologia de co-projeto [46]. Dentro do escopo do projeto de sistemas digitais de aplicação específica, o desenvolvimento do bloco de controle varia de acordo com a opção de processamento adotada. Soluções que maximizam o desempenho as expensas do tempo de projeto e da perda de flexibilidade são implementadas com base em circuitos integrados para aplicações específicas (ASICs); Uma solução que aumenta ainda mais o tempo de projeto porém permite a implementação de computação em software, flexibilizando o uso do sistema, é obtida com o uso de processadores com conjuntos de instruções dedicados (ASIPs); redução adicional no tempo de projeto, com a consequente perda de desempenho é conseguida se for usado um núcleo de processador já existente como parte do controlador (CORE); finalmente, a solução mais flexível e de custo e ciclo de projeto menores é obtida se for usado um processador de propósito geral padrão (STD).

1. representação da função lógica como uma soma-de-produtos, como no caso da síntese de dois níveis;
2. fatoração desta expressão de onde se obtém sua representação multi-nível,[20] que não representa necessariamente a solução ótima, para o que não há algorítmo geral conhecido [11]. A expressão fatorada permite a implementação lógica da função com um menor número de transistores e conseqüentemente menor área, causando entretanto maior atraso [26] e;
3. mapeamento da função resultante na biblioteca de células da tecnologia a ser usada na implementação em hardware, visando a geração de um circuito otimizado em área e dentro das limitações de tempo de propagação especificadas.

2.3 Projeto Arquitetural

Metodologias de projeto ao nível arquitetural, englobando hardware [27,31][21] e software, são ainda fracamente suportadas por ferramentas computacionais, disponíveis atualmente apenas para sistemas com arquitetura fixa e pequena ou média complexidade [42,43,44], tipicamente usados em aplicações específicas, e projetados para exercer uma função determinada dentro de um sistema maior [45].

2.3.1 Co-Projeto Hardware-Software

A área de co-projeto hardware-software[22] lida com o desenvolvimento de metodologias unificantes de projeto automatizado para sistemas compostos por software, e a hardware programável de propósito geral e de aplicação específica, e componentes eletromecânicos, englobando também questões relativas ao empacotamento mecânico do sistema [42], visando uma maior eficiência em relação às metodologias tradicionais de projeto devido ao [46]:

1. melhor suporte ao sistema operacional e programas aplicativos, bem como ao ciclo evolutivo dos produtos, tendo em vista a metodologia integrada de projeto e;
2. melhor desempenho do sistema final pelo balanceamento da computação entre

(19) Tradução para as expressões "fan-in" e "fan-out".

(20) Ao contrário do que ocorre com a soma de produtos, o complemento da expressão fatorada é uma expressão que possui o mesmo número de literais que a função original.

(21) A faceta de hardware das questões arquiteturais no entanto, tem sido favorecida pela existência de metodologias e ferramentas, desenvolvidas originalmente para suporte ao projeto de circuitos integrados que, devido a sua alta integração, permitem hoje a inclusão de pequenos sistemas computacionais em uma só pastilha.

(22) Em inglês "hardware-software codesign".

módulos de hardware programável e de software. Os enfoques básicos são a síntese de hardware programável dedicado para a aceleração da execução de funções,[23] e a transferência de funções não críticas para o software, visando a redução de custos e aumento da ftexibilidade.

2.3.1.1 Metodologia de Projeto

A tarefa inicial em um co-projeto é a obtenção de uma arquitetura para o sistema dado um conjunto inicial de especificações, que discrimine os elementos de hardware eletrônico e mecânico, e de software, sobre os quais são então aplicadas metodologias específicas de projeto, geralmente realimentadas de algum modo que permita a consistência final frente às especificações originais. A definição da arquitetura envolve dois aspectos [42] distintos e sequenciais:

1. escolha do modelo computacional ao nível arquitetural, a partir do qual os elementos de hardware e software serão organizados e;
2. criação de uma instância específica do modelo computacional que atenda aos requisitos de projeto.

Devido a forte inter-relação entre as metodologias de co-projeto e de síntese funcional, as pesquisas atuais têm se concentrado no universo dos sistemas reativos,[24] particularmente em "embedded systems",[25] onde a geração de sistemas baseados em um controlador síncrono comandando um bloco operacional, organizados em uma arquitetura pré-definida, produz resultados eficientes. A opção "a priori" por uma arquitetura-alvo praticamente suprime considerações referentes ao item I, restringindo o espaço de projeto a decisões quanto a natureza dos elementos processadores como visto na figura 4.

2.3.1.2 Aplicações

Várias propostas de metodologia de projeto para a área de co-projeto têm sido apresentadas. Gupta e De Micheli [44] propõem uma metodologia para síntese de sistemas compostos por microprocessadores e ASICs orientada a uma implementação em barramento comum, a partir de uma especificação comportamental a qual são aplicadas restrições de tempo, o que também é feito por Thomas et al [43], os quais no entanto se concentram na definição de mecanismos de sincronização. Um ambiente orientado à simulação para aplicações em processamento digital de sinais que admite a convivência de ferramentas heterogêneas é proposto por Kalavade e Lee [47]. Srivas-tava e Brodersen [42,48] apresentam uma metodologia mais abrangente, para a síntese de sistemas em uma organização de barramentos hierárquicos, suportada por um mecanismo próprio de comunicação entre processos.

Tabela 1: Panorâmica das metodologias para o projeto de sistemas de computação assistido por computador.

CAMADA	METODOLOGIA	APLICAÇÃO	ENFOQUE DE PROJETO
SOFTWARE	método da espiral	gerência de projeto	iterativo
	prototipagem	projeto de software	iterativo
	proj. orientado a objeto	projeto de software	misto
ARQUITETURA	co-projeto	hardware & software	descendente
	síntese de sistemas	projeto de sistemas/VLSI	descendente
HARDWARE	síntese funcional	projeto VLSI	descendente
	síntese lógica	projeto VLSI/LSI	descendente
	projeto físico	leiaute de microeletrônica	descendente

(23) Um exemplo é o uso de co-processadores baseados em "programmable active memories-PAMs", estas formadas por um conjunto de "fuse programmable gate arrays-FPGAs" e memórias RAM, os quais são carregados com porções críticas do software compiladas para execução em hardware.

(24) Um sistema reativo executa funções em resposta a estímulos externos de entrada, dentro de janelas específicas de tempo.

(25) "Embedded Systems" podem ser classificados como sistemas digitais para aplicação específica cujo projeto é sujeito a restrições de tempo associados a eventos assíncronos [44].

Como deficiência comum em todos os trabalhos, a partição entre hardware e software é sempre feita manualmente o que, adicionado ao uso de arquiteturas fixas, limita bastante o escopo dos ambientes de co-projeto.

3. INTEGRAÇÃO ENTRE AS METODOLOGIAS DE PROJETO

A complexidade dos projetos é enfrentada atualmente por duas linhas básicas de ação [26,30], que se distinguem quanto ao papel designado para as ferramentas computacionais:

1. a primeira linha postula que o projetista é o centro da atividade de projeto, cabendo às ferramentas uma função auxiliar, incrementando a produtividade do elemento humano através do suporte a captura de projeto, análise, verificação, etc;
2. a segunda linha de ação preconiza a possibilidade de codificação do conhecimento humano em termos de algorítmos ou regras, permitindo que o processo de projeto seja totalmente automatizado, por meio de atividades de síntese e compilação.

Em linhas gerais, metodologias de projeto evolutivas relacionam-se com o ambiente de projeto na forma descrita em 1, enquanto que metodologias descendentes[26] são baseadas em ambientes como o descrito em 2, conforme sumariado na Tabela 1. Para que haja uma melhor exploração do espaço de projeto, através da maior integração entre as diferentes metodologias existentes para software, arquitetura e hardware, três pontos são fundamentais:

Foco no domínio da descrição do problema. Como visto em 2.1, 2.2 e 2.3.1, as metodologias de projeto aplicadas aos sistemas de computação, antes confinadas ao "domínio da solução", evoluíram para a situação sumariada na tabela, onde a ênfase é concentrada no "domínio do problema" com o projeto sendo orientado pela descrição deste, tendo em vista a complexidade crescente dos sistemas, esta uma função de dois fatores básicos:

1. Uso de estruturas elementares em hardware. Apesar da complexidade apresentada pelos circuitos VLSI, estes ainda são concebidos, em sua forma final, em termos de transistores [49,50], os mesmos elementos básicos no projeto eletrônico há quase 50 anos. As metodologias de síntese de hardware, embora trabalhem em níveis de abstração que permitem a manipulação de módulos e blocos funcionais que muitas vezes independem da implementação, organizam fisicamente estes blocos em estruturas baseadas em arranjos bidimensionais de transistores, do que resulta uma solução complexa ao se fazer o mapeamento entre funcionalidade e implementação.
2. Linguagem de montagem composta por operações com baixo significado semântico. Com o fracasso das arquiteturas de processadores HLLCA [51,52][27] e a subseqüente popularização dos processadores RISC, baseados em conjuntos de instrução simplificados e com baixo significado semântico associado, ocorre nos sistemas de software fenômeno análogo ao descrito no item 1 para os circuitos VLSI, com o aumento crescente da complexidade do software.

Uso de metodologias de projeto evolutivas. O enfoque dominante em projetos de hardware (e conseqüentemente em arquitetura) é hoje marcadamente descendente, dificultando em muito a interação com a instância de software associada ao sistema, normalmente projetada segundo uma metodologia iterativa, e reduzindo a própria qualidade do projeto de hardware como argumentado McFarland e Kowalski [53]. Ambientes para o projeto evolutivo de

(26) Tradução para "top-down" segundo Wagner et al [29].

(27) Uma derivação em relação às máquinas CISC foi proposta ao final dos anos 70 quando alguns projetistas desenvolveram arquiteturas voltadas diretamente para a programação em alto nível, genericamente classificadas como "Arquiteturas de Computador Orientadas às Linguagens de Alto Nível", em inglês HLLCA-High Level Language Computer Architecture, que buscavam mapear os algorítmos diretamente na arquitetura da máquina [52] por meio da extensão do uso dos operadores existentes nos conjuntos de instruções também sobre as estruturas complexas manipuladas ao nível das linguagens de alto nível ou; pela introdução das estruturas de dados na arquitetura e uso de operações simples em sua manipulação. Sob o ponto de vista conceitual as máquinas HLLCA, como o intel iAPX 432, apresentam apenas diferenças de grau em relação às máquinas CISC existentes. Entretanto, as arquiteturas e implementações resultantes da aplicação de seus postulados apresentavam uma relação custo-desempenho altamente desvantajosa, o que terminou por determinar seu uso comercial apenas de forma residual.

hardware, voltados a nichos específicos de aplicação [54,55,56] são objeto de pesquisa na atualidade, o que pode servir para popularizar o enfoque iterativo no projeto de hardware.

Exploração do espaço de projeto ao nível arquitetural. O atrelamento do projeto arquitetural às metodologias de projeto de hardware, limita artificialmente o espaço de projeto ao uso de arquiteturas [57] baseadas na Máquina de von Neumann em sua versão original ou mesmo em derivações multiprocessadas.[28] No caso de aplicação das técnicas de síntese funcional a restrição é ainda maior, pois os circuitos sintetizados são formados por um bloco operacional ("datapath") comandado por um controlador síncrono [32]. Este cenário, embora indesejado, não apresenta perspectivas de alteração a curto prazo tendo em vista a inexistência de um paradígma independente de projeto arquitetural de escopo geral, que seja suportado por ferramentas computacionais.

4. CONCLUSÃO

Este trabalho apresentou uma visão panorâmica suscinta das principais metodologias de projeto aplicadas aos sistemas de computação procurando identificar suas características-chave e limitações, e propondo as pré-condições a serem satisfeitas, sob o ponto de vista do autor, para uma efetiva integração destas metodologias em um ambiente de projeto homogêneo e iterativo.

5. REFERÊNCIAS

[1] John Gurd, Ian Watson & John Glauert, *A Multilayered Data Flow Computer Architecture.* University of Manchester, Department of Computer Science, (March, 1981)

[2] Arvind, Kim Gostelow & Wil Plouffe, *An Assinchronous Programming Language and Computing Machine* University of California, Irvine, Department of Computer Science, (December, 1978).

[3] John Gurd, Wim Bohm & Yong Teo, *Performance Issues in Dataflow Machines* (Elsevier Science Publishers B. V., 1987), 285-297.

[4] Daniel Gasjki & Jih-Kwon Peir, "Essential Issues in Multiprocessor Systems," *IEEE Computer* (June, 1985): 9-27.

[5] Philip Treleaven, "Fifth Generation Computer Architecture Analysis," em *Fifth Generation Computer Systems* (JIPDEC, North Holland Publishing Company, 1982), 265-275.

[6] Wayne Wolf, "Hardware-Software Codesign," *IEEE Design & Test of Computers* (September, 1993)

[7] Carver Mead & Lynn Conway, *Introduction to VLSI Systems* (Addison Wesley, 1980)

[8] "VLSI Processor Architecture," *IEEE Transactions on Computers* C-33/12 (December, 1984): 1221-1246.

[9] Asaware Kalavade, *System Level Codesign of Mixed Hardware-Software Systems (PhD Thesis)* University of California, (Berkeley, 1996)

[10] Stephen Director et al, "A Design Methodology and Computer Aids for Digital VLSI Systems," *IEEE Transactions on Circuits and Systems* CAS-28/7 (1981): 634-644.

[11] R. Brayton et al, "Multilevel Logic Synthesis, "*Proceedings of the IEEE* (February, 1990): 264-300.

[12] Roger Pressman, *Soflware Engineering: A Practitioner's Approach.* (McGraw-Hill, 1994).

[13] Barry Boehm, "A Spiral Model for Software Development and Enhancement," *IEEE Computer* 21 /5 (May, 1988): 61-72.

[14] Alan Davis, "A Strategy for Comparing Alternative Software Development Life Cycle Models,"*IEEE Transactions on Software Engineering* (October, 1988)

[15] W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," *Proceedings of WESCON* (August, 1970)

[16] Barry Boehm, "Improving Software Productivity," *IEEE Computer* (September, 1987)

[17] A. Sage & J. Palmer, *Software Engineering* (John Wiley and Sons, 1990)

[18] W. Riddle & G. Williams, "Software Environments Workshop Report," *ACM SIGSOFT Software Engineering Notes* (1986)

(28) Para uma visão detalhada dos modelos computacionais ao nível arquitetural veja-se Treleaven et al [57].

[19] G. Forte, "Rally round the Repository," *CASE Outlook* (1989)

[20] Stephen Trimberger et al, "A Structured design methodology and associated software tools," *IEEE Transactions on Circuits and Systems* CAS-28/7 (July, 1981): 61 R-633.

[21] Jun Gu & Kent Smith, "A strucutred approach for VLSI circuit design," *IEEE Computer* (November, 1989)

[22] R. Cavin & J. Hilbert, "Design of integrated circuits: directions and challenges.," *Proceedings of the IEEE* (February, 1990)

[23] G. De Micheli, "High-Level Synthesis of digital Circuits," *IEE Design & Test of Computers* (October, 1990)

[24] G. De Micheli, "Hardware-softtware codesign," *IEE Micro* (August, 1994)

[25] K. Mueller-Glaser & J. Bortolazzi, "An approach to computer-aided specification," *IEEE Journal of Solid-State Circuits* (April, 1990)

[26] N. Dutt & D. Gajski, "Design synthesis and silicon compilation," *IEEE Design & Test of Computers* (December, 1990)

[27] M. McFarland et al, "The High Level synthesis of digital Systems," *IEEE Journal of Solid-State circuits* 25/2 (1990)

[28] R. Burger, "The impact of ICs on computer technology," *IEEE Computer* (October, 1984)

[29] Flávio Wagner et al, *Métodos de Validação de Sistemas Digitais* (VI Escola Brasileira de Computação,1988)

[30] D. Gajski & Robert Kuhn, "New VLSI Tools", *IEEE Computer* (June 1985)

[31] Elisabeth Lagnese & Donald Thomas. "Architectural Partitioning for System Level Synthesis of Integrated Circuits", *IEEE Transactions on Computer-Aided Design*, (July, 1991)

[32] Raul Camposano. "From Behaviour to Strucutre: High-Level Synthesis", *IEEE Design & Test of Computers*, (October, 1990)

[33] Cheng-Tsung Hwang et al. "A formal approach to the scheduling problem in high level synthesis", *IEEE Transactions on Computer-Aided Design*, (April, 1991)

[34] Synopsys. *High Level Design Training Workbook*, 1996

[35] Douglas Lewin. *Computer Aided Design for Microcomputer Systems* in: *Microcomputer System Design: an Advanced Course*, (Springer-Verlag, 1982)

[36] Douglas Lewin. *Computer Aided Design of Digital Systems*, (Crane, Russak & company, Inc., 1977)

[37] Peirre Abouzeid et al. "Input-driven partitioning methods and application to synthesis on table-lookup-based FPGAs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, (July, 1993)

[38] L. Bouchet & G. Saucier, "Multi-level Synthesis on PALs and on PAGs", in: *IFIP Working Conf. Logic Architecture Synthesis*, (May, 1990)

[39] R. Munoz & C. Stroud, "Automatic partitioning of programmable logic devices", *VLSI System Design*, (October, 1987)

[40] J. Millman & C. Halkias. *Eletrônica*, (McGraw-Hill do Brasil, 1981)

[41] Yen-Chen Wei et al, "Multiple-level partitioning: na application to the very large-scale hardware simulator", *IEEE Journal of solid State Circuits*, (May, 1991)

[42] Mani Srivastava & Robert Brodersen, "SIERRA: A unifiede ftamework for rapid-prototyping of system level hardware and software", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, (June, 1995)

[43] D. Thomas et al, "A model and methodology for hardware-software cod esign", *IEEE Design & Test of Computers*, (September, 1993)

[44] R. Gupta & G. De Micheli, "Hardware-Software Cosynthesis for Digital Systems", *IEEE Design & Test of Computers*, (September, 1993)

[45] Frank Vahid et al, "A VHDL ftont-end for embedded systems", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, (June,1995)

[46] G. De Micheli, "Computer-Aided Hardware-Software Codesign", *IEEE Micro*, (August, 1994)

[47] A. Kalavade & E. Lee, "A hardware-software codesign methodology for DSP applications",

IEEE Design & Test of Computers, (September, 1993)

[48] M. Srivastava & R. Brodersen, "Using VHDL for high-level, mixed-mode system simulation. *IEEE Design & Test of Computers*, (September, 1992)

[49] Stan Hurst, "Custom microelectronics technologies and developments: a survey", *Journal of Semicustom ICs*, (March 1990)

[50] Randall Geiger et al, *VLSI Design Techniques for Analog and Digital Circuits*, (McGraw-Hill, 1990).

[51] G. Meyers, *Advances in Computer Architecture*, (J. Wiley and Sons, 1982)

[52] J. Browne, "Understanding execution behaviour of software systems", *IEEE Computer*, (July 1984)

[53] M. McFarland & T. Kowalski, "Incorporating; bottom-up dcsil,m into hardware synthesis," *IEEE Transactions on Computer-Aided Design* 9/9 (September, 1990): 938-950.

[54] V. Madisetti & J. Debardelaben, "A RASSP Approach to Hardware/Software Codesign," *The HASSP Digest* 2/4 (1995)

[55] V. Madisetti, "Vive la Defference," *The RASSP Digest* 2/4 (1995)

[56] D. Gajski et al, *A Design Methodology and Environment for Interactive Behavioural Synthesis* Dept. of Information and Computer Science, University of California, Irvine, (June, 1996).

[57] Philip Treleaven et al, "Data-Driven and Demand-Driven Computer Architecture," *ACM Comp. Surveys* (vtarch, 1982).

ABORDAGENS AO PROCESSAMENTO SIMBÓLICO DA LINGUAGEM NATURAL

APPROACHES TO SYMBOLIC NATURAL LANGUAGE PROCESSING

João Luís Garcia ROSA*

RESUMO

Segundo Pereira e Grosz (1993), o Processamento de Linguagem Natural (PLN) é dividido basicamente em três abordagens simbólicas: baseada em casos, baseada em princípios e baseada em regras. Na Inteligência Artificial, existe há muito tempo um debate entre as abordagens baseadas em regras e as abordagens baseadas em casos. Já na Lingüística e na comunidade de PLN, há um debate entre abordagens baseadas em regras e baseadas em princípios. Este debate contrasta, por exemplo, descrições de regras de estrutura de frase da sintaxe da linguagem natural em que as regras são específicas da linguagem, com abordagens baseadas em princípios nas quais um conjunto de princípios independentes da linguagem são modulados por certos estabelecimentos de *parâmetros*, por exemplo, com respeito a ordem da palavra, para caracterizar linguagens naturais particulares. O objetivo deste trabalho é discutir estas três abordagens de forma comparativa.

Palavras Chave: Processamento de Linguagem Natural, Lingüística Computacional, Abordagem Simbólica, Inteligência Artificial.

ABSTRACT

According to Pereira and Grosz (1993), Natural Language Processing (NLP) is divided basically into three symbolic approaches: case-based, principle-based, and rule-based approach. In Artificial Intelligence, there is for a long time a debate between rule-based approaches and case-based approaches. In Linguistics and in the NLP community, there is a debate between rule-based and principle-based approaches. This debate contrasts, for instance, natural language syntax descriptions of phrase structure rules in which rules are language dependent, with principle-based approaches in which a set of language-independent principles are modulated by some *parameters*, for instance, in relation to the order of words, to characterize particular natural languages. The aim of this paper is to discuss these three approaches in a comparative way.

Keywords: Natural Language Processing, Computational Linguistics, Artificial Intelligence

(*) Professor do Instituto de Informática da PUC-Campinas e doutorando em Lingüística Computacional pelo IEL-Unicamp. E-mail: joao@zeus.puccamp.br.

1. INTRODUÇÃO

Pereira e Grosz (1993) dividem o Processamento de Linguagem Natural (PLN) em três abordagens simbólicas: baseada em casos, baseada em princípios e baseada em regras. Mas, o que são casos, princípios e regras? Um *caso* é uma associação entre uma situação prototípica e a informação relevante à tarefa que a segue. Por exemplo, um caso pode representar uma sentença da linguagem natural envolvendo o verbo principal *dar* e alguma outra informação desta sentença, por exemplo, que depois da ação descrita pelo verbo, o agente da ação não tem mais a posse do "paciente" da ação. O raciocínio baseado em casos envolve a descrição de analogias entre situações observadas recentemente e casos relevantes e o uso da informação de tarefa associada para determinar as inferências apropriadas às novas situações.

Um *princípio* é uma restrição aos tipos de situações possíveis: permite que um sistema infira características de situações adicionais a partir de outras características observadas. Por exemplo, um princípio na sintaxe da linguagem natural requer que cada sintagma nominal em uma sentença preencha exatamente uma posição argumental de um item lexical com posição argumental tal como um verbo ou uma preposição. Tal princípio restringe as associações possíveis entre itens lexicais com posição argumental e sintagmas nominais e portanto restringe a faixa de significados que podem ser expressos por uma determinada sentença.

Uma *regra* especifica como certas características de, ou relacionamentos entre, situações seguem de outras. Por exemplo, de novo na sintaxe da linguagem natural, uma regra de algumas línguas estabelece que um sintagma nominal (SN) seguido por um sintagma verbal (SV), havendo concordância em gênero e número, pode formar uma sentença (S), com o SN como sujeito e o SV como predicado.

2. O RELACIONAMENTO ENTRE REGRAS E CASOS

O contraste entre as abordagens baseada em regras e baseada em casos está essencialmente na fonte de generalidade de um sistema. Em sistemas baseados em regras, a generalidade vem da escolha de primitivas descritivas que permitem grandes coleções de situações com resultados similares a serem identificados e trabalhados por regras; em contraste, a generalidade em um sistema baseado em casos vem dos procedimentos de recuperação de caso e unificação (*matching*) que determinam o resultado para uma situação nova a partir de resultados para casos similares armazenados. Permitindo noções de unificação parcial ou aproximada, os sistemas baseados em casos são freqüentemente capazes de agir mesmo quando seu conhecimento de caso não unifica totalmente com a situação sob análise. Por outro lado, as regras previamente projetadas podem resumir e identificar eficientemente os itens comuns em grandes conjuntos de casos, tornando então o conhecimento do sistema mais largamente aplicável.

A utilidade de uma abordagem baseada em casos depende crucialmente da eficiência dos mecanismos de aquisição e uso da informação específica sobre a distribuição das situações de interesse. No PLN, tais situações envolvem objetos lingüísticos tais como palavras ou unidades fonéticas em determinados contextos. Enquanto as abordagens baseadas em casos devem ser avaliadas por sua habilidade de aprender casos relevantes, generalizá-los apropriadamente e aplicá-los, a falta de seleção de caso e de métodos de generalização efetivos força os praticantes atuais a criarem a maior parte da informação de caso a mão. Dado isto, os problemas mais importantes enfrentados por estes sistemas são a escolha dos traços (*features*) de caso relevantes à seleção de caso, reconhecimento dos casos que se aplicam a uma situação dada e a construção de interpretações para enunciados (*utterances*) complexos a partir de combinações de casos apropriados unificando partes do enunciado.

Enquanto as abordagens ao PLN baseadas em casos têm muito da sua inspiração a partir das idéias da ciência cognitiva que trata da organização da memória e inferência do senso comum, as abordagens baseadas em regras derivam na maior parte das tradições fortes da lingüística e da teoria de linguagens formais. Estas origens têm levado a arquiteturas de sistema centradas sobre as noções da descrição estrutural e da transdução estrutura a estrutura. Por exemplo, as regras de estrutura de frase são usadas para descrever a sintaxe da linguagem natural e regras adicionais em cascata são então usadas para transformar tais descrições estruturais, através de

uma sucessão de representações intermediárias, em uma representação do conteúdo das sentenças originais. Várias representações têm sido usadas, incluindo fórmulas lógicas, redes semânticas e quadros (*frames*). Enquanto as arquiteturas baseadas em regras têm produzido sistemas de processamento de linguagem muito expressivos, elas têm encontrado sérias dificuldades na área da robustez, isto é, a habilidade de produzir saída útil mesmo diante de regras muito específicas ou ausentes e de tratar com fenômenos não composicionais, ou seja, situações nas quais a saída apropriada numa situação complexa não pode ser derivada por uma regra simples a partir das saídas para suas partes.

3. O RELACIONAMENTO ENTRE REGRAS E PRINCÍPIOS

Um outro conjunto de dificuldades com sistemas baseados em regras no PLN surge da rigidez e especificidade das regras. Por exemplo, com a exceção de alguns sistemas recentes que usam formalismos de regra baseados em restrições declarativas e estratégias sofisticadas de aplicação de regras, as considerações baseadas em regras do mapeamento sintaxe-significado são tipicamente unidirecionais; portanto, evita-se o uso das mesmas regras para interpretação e geração da linguagem. Mais fundamentalmente, os sistemas a regras são específicos da linguagem e da construção, portanto requerem esforço maior para serem transportados para outras línguas ou mesmo para outras partes da mesma língua ou outros domínios.

Estas dificuldades podem ser vistas como sintomas da restrição da noção usual de regra, que força uma definição gerativa do relacionamento entre análises e interpretações. Por exemplo, um sistema que mapeia análise sintática para fórmulas lógicas que representam significados da sentença, teria tipicamente uma regra da gramática estabelecendo que uma sentença como "um estudante fez todo o teste" é composta de um sintagma nominal sujeito ("um estudante") seguido por um sintagma verbal predicado ("fez todo o teste"). Associado com esta regra da gramática haveria uma regra de interpretação estabelecendo que o significado da sentença é igual ao significado do sujeito aplicado ao significado do predicado. No nosso exemplo, o significado do sujeito poderia ser uma fórmula que pode ser explicada como "verdadeira para qualquer propriedade que tem algum estudante", e o significado do predicado como uma fórmula que podemos explicar como "propriedade de fazer todo o teste". A interpretação resultante para a sentença poderia então ser explicada como "existe um estudante que tem a propriedade de ter feito todo o teste". Esta interpretação força o quantificador do sujeito ter escopo mais largo do que o quantificador do objeto. Mas, para adequadamente manipular linguagem natural, o processo de interpretação precisa considerar escopos alternativos antes de escolher aquele que é contextualmente mais apropriado.

A causa fundamental deste problema é que regras privilegiam conexões gerativas particulares entre evidência e interpretação. Em contraste, a evidência específica que pode ser extraída de uma situação natural tal como um enunciado (isto é, a estrutura sujeito-predicado no exemplo anterior) é muito indeterminada para ser confiavelmente modelada como uma transdução entre um domínio de descrições estruturais e um domínio de interpretações.

Em contraste às abordagens baseadas em regras, nas abordagens baseadas em princípios, os princípios fornecem restrições fundamentais gerais entre tipos de evidências em diferentes níveis de descrição. A análise da linguagem, interpretação ou geração é vista não como um processo de rescrita, mas como uma busca pela hipótese que melhor explica a evidência observada; o espaço de busca é implicitamente definido pelos princípios e por restrições de domínio.

Numa visão baseada em princípios, em processos com componente perceptual como o processamento da linguagem, as restrições impostas pelo sistema natural fundamental são uma fonte crucial de princípios. Na linguagem natural, uma visão correspondente (devido à teoria lingüística de princípios e parâmetros de Chomsky e seus seguidores) assegura que os princípios, ancorados em critérios determinadores evolucionários tal como aprendizagem, eficiência comunicativa e carga cognitiva, provêem um sistema de regularidades "legais" que definem os espaços de representações possíveis nos vários níveis relevantes de descrição e as restrições entre estes níveis. Na linguagem, estes níveis incluem sintaxe, semântica, discurso e prosódia, ain-

da que o trabalho de PLN baseado em princípios tem se concentrado somente nos níveis sintático e lexical. Abstratamente, os princípios não são apenas independentes de modelos de processamento específicos mas também de línguas particulares ou domínios de discurso. Entretanto, para ser usado na prática, um sistema baseado em princípios deve ser "preenchido" com conhecimento sobre objetos particulares - línguas, palavras, conceitos - nos termos dos princípios. Noções de aprendizagem têm um papel importante na concepção das teorias baseadas em princípios; entretanto, algoritmos efetivos para aprendizagem do conhecimento específico necessário ainda não estão disponíveis. Por ora, o conhecimento específico deve ser descrito à mão como também é o caso dos sistemas baseados em regras e baseados em casos, mesmo que a generalidade dos princípios em algumas instâncias permita especificação mais concisa do conhecimento específico.

Enquanto nas abordagens baseadas em regras tem-se uma regra para cada construção, nas abordagens baseadas em princípios, considera-se alguns princípios apenas, que combinados, atendem a qualquer construção. Os princípios considerados são (Berwick, 1992): teoria X-Barra, filtro de caso, critério temático, *move*-a, teoria de vestígios e teoria de ligação. Apesar de, aparentemente, este tipo de abordagem parecer ineficiente computacionalmente, é bem interessante pelas seguintes razões:

1. o mesmo conjunto (pequeno) de princípios pode ser recombinado várias vezes, de diferentes formas, resultando em muitas sentenças de superfície, e variando os parâmetros, em diferentes dialetos e línguas;
2. princípios abstratos e heterogêneos, estabelecidos como um conjunto de restrições declarativas, ao contrário de uma representação mais uniforme como um conjunto de regras livres de contexto;
3. ênfase na importância do léxico, fonte de restrições de papel temático e variação de línguas particulares.

4. *PARSER* BASEADO EM PRINCÍPIOS

Segundo Crocker (1991), as abordagens tradicionais ao PLN podem ser consideradas baseadas em construção. Isto é, elas empregam regras específicas da linguagem orientadas à superfície, ou na forma de Redes de Transição Aumentadas (ATN, de Woods, 1970), gramáticas lógicas ou algum outro formalismo de gramática ou *parsing* (Pereira e Warren, 1980). Os problemas de tais abordagens são claros, pois envolvem grandes conjuntos de regras, freqüentemente *ad hoc*, e sua adequação com respeito à gramática da linguagem é difícil de assegurar. Em contraste, esforços na pesquisa lingüística têm observado certas regularidades nas linguagens naturais. De fato, uma tentativa inicial foi caracterizar esses universais lingüísticos, que definiriam a classe das linguagens naturais, resultando na teoria da Gramática Universal (GU). A melhor teoria da GU desenvolvida por Chomsky e outros foi um paradigma de Princípios e Parâmetros, freqüentemente referido como a teoria da Regência e Ligação (GB para *Government and Binding*). GB é uma teoria dedutiva e modular da gramática que trabalha com vários níveis de representação relacionados por uma regra transformacional, a *move*-α. A aplicação de *move*-α é restringida pela interação de vários princípios que agem como condições às possíveis representações ou derivações. Associados com os princípios estão os parâmetros que dão conta das variações entre as línguas. Então a gramática para uma determinada língua é especificada pelo estabelecimento de parâmetros apropriados e por um léxico.

As abordagens baseadas em princípios não são apenas independentes de modelos de processamento específico mas também de linguagens ou domínios de discurso particulares.

Crocker (1996), no contexto da teoria da gramática dos princípios e parâmetros assumida, discute a noção "baseado em princípios": um modelo que usa os princípios da gramática diretamente na recuperação de uma análise sintática. Isto é, ele *não* usa uma gramática compilada, transformada[1]. Ou seja, existem várias teorias de performance[2] que podem ser consideradas baseadas em princípios, no sentido de

(1) Um exemplo disto é o *parser* de Marcus (1980), que computa uma estrutura de superfície, incluindo relações antecedente-vestígio. O *parser* faz isto sem tornar explícito o uso dos princípios da gramática, mas no entanto ele os obedece. Este *parser* é 'fracamente baseado em princípios'.

(2) *Competência* se refere ao nosso conhecimento da língua e *performance* ao modo como usamos este conhecimento.

que elas usam os princípios da gramática *on-line*, mas que tomam decisões na base de critérios 'não lingüísticos', tal como eficiência computacional ou complexidade representacional. Tais modelos claramente contrastam com teorias de processamento que operam de acordo com estratégias baseadas em gramática, sugerindo um relacionamento mais próximo entre *parser* e gramática. Uma teoria de performance que é baseada em princípios e incorpora estratégias que são baseadas em gramática (no sentido descrito), é chamada de 'fortemente baseada em princípios'.

5. APLICAÇÕES DO *PARSING* BASEADO EM PRINCÍPIOS

Cornell (1993) mostra que o interesse de se trabalhar com abordagens baseadas em princípios se justifica também pelas conseqüências teóricas interessantes, como por exemplo, uma visão mais clara do lugar da teoria GB dentro do universo do formalismo lingüístico. Ele trabalha com sistemas de princípios gramaticais através do problema mais simples do *parsing* com sistemas de condições de licenciamento[3]. O autor desenvolve um formalismo para expressar as Gramáticas de Licenciamento, isto é, gramáticas construídas inteiramente de condições de licenciamento, e explora a interpretação de tais gramáticas como sistemas de restrições (sua interpretação tradicional) e como sistemas de produção, tratados como regras de rescrita.

O trabalho de Stabler Jr. (1993) trata da abordagem baseada em princípios onde os mesmos são estabelecidos como axiomas de uma linguagem de primeira ordem. Stabler usa a resolução no seu *parser* baseado em lógica de primeira ordem que usa a teoria lingüística transformacional de Chomsky: gera árvores para DS, SS e LF a partir da entrada PF[4]. Stabler frisa que o Prolog puro não pode resolver todos os problemas da linguagem natural. O Prolog puro usa a resolução SLD, ou seja, cláusulas definidas onde somente um literal pode ser positivo[5]. Em outras palavras, numa implicação, só se pode ter um conseqüente. Ele propõe uma teoria enriquecida chamada de G1, onde pode-se concluir que certas construções *não* são sentenças, coisa que só cláusulas de Horn não conseguem. Para isto, Stabler tenta especificar as propriedades computacionais de seu modelo, ou seja, necessidade de componentes não-Horn para o problema do *parsing*:

1. princípio da categoria vazia: nenhum constituinte pode estar vazio.
2. filtro de caso: não falta caso para nenhum SN não vazio.
3. subjacência: nenhuma cadeia pode cruzar mais de uma "barreira".
4. c-comando.

Uma outra aplicação para o *parsing* baseado em princípios é o trabalho de Mchale (1995). Neste artigo, o autor propõe uma abordagem combinada entre um *parser* baseado em princípios e um dicionário legível por máquina semanticamente. O *parser* é implementado com a GB do Chomsky e sua cobertura sintática é uma função do tamanho e riqueza de seu léxico (extraído do *Longman's Dictionary of Contemporary English*) e enriquecido semanticamente usando o *Roget's International Thesaurus*.

Sua pesquisa investiga:

(1) o impacto de usar um dicionário legível por máquina como o léxico para um *parser* baseado em princípios;

(2) a extração automática de papéis temáticos deste dicionário; e

(3) métodos para enriquecer estes papéis usando o *Roget's*.

Crocker (1996) também aplica o conceito de *parsing* baseado em princípios no seu processador de sentenças. Baseado no conceito de modularidade da mente e na natureza da competência e performance lingüísticas, o autor defende duas hipóteses fundamentais sobre a arquitetura e princípios básicos do mecanismo de processamento de sentenças humano: a modularidade, que diz que o processador de sentenças constitui um sistema distinto dentro da

(3) Condição de licenciamento é uma condição da sintaxe das linguagens naturais que permite que uma determinada construção sintática possa ser considerada como gramatical.

(4) DS=Estrutura-D; SS=Estrutura-S; LF=Forma Lógica; e PF=Forma Fonética.

(5) Cláusulas de Horn são disjunções de literais da lógica que contêm no máximo um literal positivo. Cláusulas Definidas são cláusulas de Horn que possuem um literal positivo, ou seja, uma única conclusão definida para a implicação. Um programa Prolog puro consiste de cláusulas definidas.

faculdade da linguagem e a incrementalidade, que diz que a operação do processador de sentenças e seus módulos constituintes é determinada pelo princípio da compreensão incremental, que assegura que o máximo uso da informação lingüística é possível, assim que cada palavra de um enunciado é encontrada.

Ao invés das teorias de sintaxe serem baseadas em sistemas de regras grandes e complexos, estas teorias dependem da interação de um pequeno conjunto de princípios universais que são parametrizados entre as línguas. Fong (1992a) descreve um sistema de *parsing* baseado em princípios que alcança cobertura lingüística substancial e eficiente enquanto mantém um nível de representação de princípios próximo ao usado na literatura da lingüística. Usando um pequeno conjunto de vinte e cinco princípios, o sistema demonstra dar conta corretamente de centenas de construções diferentes a partir de um livro texto introdutório de lingüística. Também ilustrando a natureza universal das teorias baseadas em princípios, o mesmo conjunto de princípios demonstra cobrir exemplos de dados do japonês assim como do inglês. O autor também investiga o problema da configuração de princípios para *parsers* eficientes. O sistema de *parsing* incorpora parâmetros de controle flexíveis independentes das definições de princípios. Estes parâmetros de controle efetivamente definem uma família de *parsers* que incorporam o mesmo conhecimento lingüístico, mas com diferentes características de performance. Fong, através da investigação do efeito das variações nos estabelecimentos de controle, obtém uma caracterização das propriedades computacionais relevantes dos princípios que determinam as configurações de controle mais apropriadas para um *parsing* eficiente.

Merlo (1995) discute o problema relacionado à eficiência dos *parsers* baseados na GB. Ela argumenta que a GB não é uma teoria computacionalmente modular e por isso, os *parsers* baseados nesta teoria lingüística não são eficientes. Ela diz que um *parser* eficiente e confiável pode ser construído tirando vantagem da forma como os princípios são estabelecidos. Para sustentar este ponto de vista, duas características de um *parser* implementado são discutidas. Primeiro, configurações e informação lexical são pré-compiladas separadamente em duas tabelas (uma tabela X-Barra e uma tabela de co-ocorrência lexical). Segundo, a pré-computação de traços sintáticos (papéis theta, caso, etc.) resulta em computação eficiente de cadeias, porque reduz muitos problemas de formação de cadeia para uma computação local, evitando busca extensiva da árvore para um antecedente ou *backtracking* extensivo. A autora mostra também que este método de construção de dependências de longa distância pode ser computado incrementalmente.

Fong (1992b) construiu um *parser* baseado em lógica, o PO-*PARSER* (*parser* de ordenação por princípios) para investigar e demonstrar os efeitos da ordenação de princípios. O PO-*PARSER* foi propositadamente construído de uma forma altamente modular para permitir uma flexibilidade máxima na exploração de ordenações alternativas de princípios. Por exemplo, cada princípio é representado separadamente como uma operação atômica do *parser*. Uma estrutura é considerada bem formada apenas se ela passa por todas as operações do *parser*. O escalonamento das operações do *parser* é controlado por um mecanismo dinâmico de ordenação que tenta evitar trabalho desnecessário através da eliminação de estruturas mal formadas o mais rápido possível. Apesar da preocupação principal ser a exploração das propriedades computacionais dos princípios para construir *parsers* mais eficientes, o PO-*PARSER* é também capaz de tratar uma grande variedade de fenômenos lingüísticos, como os princípios da teoria theta, teoria de caso, teoria de ligação, subjacência, princípio da categoria vazia (ECP), movimento em nível da forma lógica etc.

Tradicionalmente, a informação semântica em léxicos computacionais é limitada a noções tais como restrições selecionais ou restrições específicas de domínio, codificada numa representação 'estática'. Esta informação é tipicamente usada no PLN por um simples mecanismo de manipulação de conhecimento limitado à habilidade de unificar instâncias de palavras relacionadas estruturalmente. O dispositivo mais avançado para estruturar a informação lexical é a herança, para os níveis objeto (itens lexicais) e para os meta níveis (conceitos lexicais) do léxico. Pustejovsky e Boguraev (1993) apresentam uma visão de um léxico computacional através da descrição de uma teoria de semântica lexical. Esta teoria faz uso de uma representação do conhecimento que oferece um vocabulário mais rico e mais expressivo para a informação lexical.

6. *PARSER* BASEADO EM CASOS

Para conectar o texto de entrada ao conhecimento e metas prévios do entendedor, deve-se ter um modelo no qual o acesso ao conhecimento e metas prévios seja uma parte integral do processo de *parsing*. O *parsing* baseado em casos atende a esse requisito. O objetivo de um *parser* baseado em casos é reconhecer quais estruturas de memória já existentes são mais relevantes à entrada, onde esta "relevância" é determinada pelos planos e metas do entendedor. Esta abordagem difere dos modelos tradicionais de *parsing* que tenta construir uma análise sintática ou uma estrutura de significado conceptual para um texto. O *parsing* baseado em casos é primariamente um processo de *reconhecimento* (Martin, 1989).

Como o *parsing* baseado em casos objetiva uma meta diferente dos outros *parsers*, o algoritmo para um *parser* baseado em casos também é diferente. Certos aspectos do algoritmo codificam conhecimento sintático ou conceptual, mas o algoritmo tem a principal preocupação de prover acesso às estruturas de memória preexistentes o mais cedo possível no curso do PLN. Esse acesso às estruturas de memória é essencial ao entendimento. Portanto, a organização da memória é fundamental a um *parser* baseado em casos. Este deve ser a ponte entre os itens lexicais primitivos da entrada e as estruturas de memória direcionadas identificadas como a saída do processo de entendimento. O *parsing* baseado em casos depende dos planos e metas idiossincráticos do entendedor.

Um texto pode - e deve - se referir a muitas estruturas de memória, sendo que cada estrutura é uma caracterização diferente da entrada nos termos relacionados à meta. A noção de um único significado para um texto é abandonada no *parsing* baseado em casos.

Os *parsers* convencionais, que constróem uma representação do significado de um texto de entrada, geralmente retornam uma representação como a saída do processo de *parsing*. Um *parser* baseado em casos, entretanto, pode ser chamado a reconhecer múltiplas estruturas de memória no curso do processamento de um texto de entrada. O que é significativo sobre essas estruturas de memória não são suas representações individuais, mas suas conexões com outras estruturas de memória que podem também ser relevantes ao texto. O conjunto de expectativas e referências do *parser* determina quais estruturas de memória serão reconhecidas. É este conjunto de expectativas e referências que constitui um resultado do processo do *parsing* bem sucedido.

A saída de um *parser* baseado em casos pode ser caracterizada como *um novo estado de memória*. Algumas estruturas terão sido referenciadas por um texto de entrada ou processo de inferência e algumas serão esperadas. Ainda que novas estruturas de memória sejam adicionadas, quando a informação específica já não estiver na memória, é o estado de referência e expectativas que constitui a saída real do sistema.

Um *parser* baseado em casos usa itens lingüísticos tais como palavras individuais para direcionar o processo de busca aos conceitos na memória. A tarefa de busca na memória consiste em conectar essas referências espalhadas, achando as unidades organizacionais para a memória que melhor organizem a entrada.

O conhecimento do processamento de um *parser* baseado em casos está na forma de *expectativas*. As expectativas são baseadas na idéia de senso comum de que as pessoas são capazes de fazer previsões sobre o que deve acontecer no futuro baseado no que aconteceu no passado e na sua experiência anterior.

As expectativas são derivadas de exemplos estereotípicos do uso da linguagem, que apontam para as unidades organizacionais da memória. Estes são os índices para um *parser* baseado em casos. Portanto, estes *parsers* usam exemplos específicos de uso da linguagem para indexar estruturas de memória.

A hipótese fundamental do *parsing* baseado em casos é que o acesso ao conhecimento prévio na forma de estruturas de memória dinâmicas, específicas do domínio, é crucial nos estágios mais iniciais do entendimento da linguagem natural. É a idéia do *parsing* através da lembrança. Realizar o *parsing* a partir de casos significa lembrar instâncias passadas do uso da linguagem tal que possam ser reconhecidas de novo e lembrar conceptualizações passadas tal que possam ser chamadas novamente. O *parsing* baseado em casos é muito diferente da análise conceptual. Questões de organização de memória, indexação e busca na memória são de importância central para um *parser* baseado em casos porque

este deve operar dentro de um modelo de memória existente.

A tarefa de *parsing* é um problema de busca na memória. Conceitos não são construídos a partir de pedaços derivados da entrada. Ao invés disso, já existem conceitos que preenchem muitas necessidades do entendedor. A tarefa é usar os indícios supridos pela tarefa para localizar os conceitos mais relevantes e modificá-los quando necessário para refletir as diferenças entre o que é visto e o que já é conhecido. Já que um *parser* baseado em casos faz uso da memória, ele pode fazer uso das expectativas derivadas desta memória. Estas expectativas dirigem o processo de *parsing*.

O *parser* baseado em casos difere de outras abordagens, principalmente em relação aos seguintes pontos:

1. Ao invés de acessar uma gramática geral da sintaxe da linguagem para determinar os elementos relacionados de um enunciado, um *parser* baseado em casos captura as formas idiossincráticas nas quais a linguagem é usada para referenciar determinados conceitos na memória.
2. Ao invés de construir conceptualizações para representar o significado de um texto de entrada, um *parser* baseado em casos faz uso dos elementos de análise conceptual para direcionar o processo de busca à memória aos conceitos que organizam estes elementos.
3. Ao invés de depender da memória para resolver ambigüidades de possíveis interpretações depois do *parsing* realizado, um *parser* baseado em casos usa as metas e expectativas na memória para resolver ambigüidades da entrada durante o processo de *parsing*.

A saída de um *parsing* baseado em casos inclui mudanças nas estruturas de memória e no contexto de expectativas na memória. A moderna teoria lingüística busca "capturar as generalizações significativas" no uso da linguagem. Isto tem levado quase que exclusivamente à busca dos padrões sintáticos. O *parsing* baseado em casos, entretanto, se preocupa mais com a caracterização de como o texto se refere a conceitos.

7. APLICAÇÕES DO *PARSING* BASEADO EM CASOS

Jones (1993) propõe um novo método para o processamento de linguagem natural, chamado de processamento baseado em analogias ou em exemplos (na verdade, outro nome que se dá para a abordagem baseada em casos). O paradigma baseado em exemplos é comparado e contrastado com o processamento convencional baseado em regras e o autor discute as vantagens e desvantagens de ambos. Um dos principais temas de seu trabalho é como melhor representar exemplos em um ambiente baseado em exemplos tal que alguma capacidade gerativa possa ser dada a um sistema sem recurso às técnicas baseadas em regras, portanto reduzindo a inflexibilidade associada com a computação lingüística dirigida analogamente. É sugerido que, no campo do processamento de linguagem baseado em analogias, os pesquisadores assumam que as técnicas baseadas em regras convencionais devam ser usadas em algum ponto da arquitetura de tais sistemas. Onde o processamento baseado em regras não é usado, tenta-se realizar uma arquitetura baseada em analogias.

Kolodner (1993) cita em sua biblioteca de casos a aplicação Parse-O-Matic de Goodman (1991), que trata do *parsing* de linguagem natural. O sistema usa uma biblioteca de casos de 50.000 micro-casos para criar uma representação semântica de questões comuns da língua inglesa. Resolução de ambigüidade do sentido da palavra e morfológica, referência pronominal e elisão são tratados de uma maneira integrada. Um micro-caso é uma ação individual realizada no curso do raciocínio. O sistema tem como ênfase o uso de casos codificados como seqüências temporais de micro-casos, comparação de eficiência, velocidade e tempo de engenharia de conhecimento para criação e manutenção de sistemas baseados em casos e baseados em regras. Este sistema foi construído na metade do tempo necessário para construir um sistema baseado em regras com eficiência comparável.

Jones e Boguraev (1987) descrevem uma aplicação de representações baseadas em casos para o processamento da linguagem. Eles usam um analisador de linguagem que constrói representações de significado expressando papéis de caso semântico; especificamente, o analisador de Boguraev

(1979) constrói árvores de dependência com sentidos de palavras definidos por fórmulas 'primitivas de categoria semântica', e com rótulos de caso, isto é, 'primitivas de relação semântica', nas estruturas dos constituintes do verbo (e outras categorias). No seu estudo de casos, os autores trazem uma lista de casos, usando exemplos de sentenças, para ser usada por aqueles que desejam a aplicação no PLN.

Martin (1989) implementou um *parser* baseado em casos, o micro DMAP, que usa uma arquitetura de passagem de marcadores para identificar estruturas de memória relevantes a partir do texto de entrada e das expectativas na memória. Dois tipos de marcadores são usados no sistema: marcadores de *ativação*, que capturam informação sobre o texto de entrada e as estruturas de memória referenciadas correntemente, e os marcadores de *previsão*, que indicam quais estruturas de memória podem se tornar referenciadas. Se um marcador de previsão é *passado* a um conceito e um marcador de ativação é subseqüentemente também passado a este conceito, diz-se que existe uma *interseção* a este conceito. Os marcadores podem carregar mais informação do que a representada pelo conceito, neste caso o conceito pode ser *refinado* a um conceito mais específico. Por último, o conceito refinado é *reconhecido*.

8. CONCLUSÃO: CASOS, REGRAS E PRINCÍPIOS NOS SISTEMAS DE PLN

A faixa das organizações de sistemas constituída por princípios, casos e regras forma um *continuum* multifacetado no qual muitas opções diferentes podem ser consideradas (Pereira e Grosz, 1993). Em uma faceta, os princípios podem ser vistos como fornecedores do conhecimento inicial crucial na especificação do espaço de casos possíveis e representações apropriadas, adquiridas ou recuperadas. Os mecanismos baseados em casos podem ser usados como uma reserva, que entram em ação quando os princípios conhecidos são insuficientes para derivar a interpretação de uma situação particular ou para decidir entre interpretações alternativas compatíveis com os princípios. Em uma outra faceta, a informação derivada de caso pode ser altamente abstraída pelos projetistas de sistemas para as restrições específicas da linguagem tal como a ordem da palavra e sistemas flexionais ou representações e restrições específicas do domínio tais como aquelas que especificam as propriedades sintáticas, semânticas e de domínio de determinadas entradas lexicais. Em ainda uma outra faceta, as regras podem ser vistas como codificações orientadas computacionalmente de determinadas instâncias de princípios apropriados às tarefas ou situações particulares; como a computação direta a partir dos princípios é em geral muito difícil, as regras podem ser preferidas por razões computacionais.

9. REFERÊNCIAS

BERWICK, R. C. (1992), Principles of Principle-Based *Parsing*. In Berwick, R. C. & Abney, S. P. & Tenny, C. (Eds.), *Principle-Based Parsing: Computation and Psycholinguistics*. 1-37. Kluwer Academic Publishers.

BOGURAEV, B. K. (1979), Automatic Resolution of Linguistic Ambiguities. *Technical Report 11*, Computer Laboratory, University of Cambridge. *apud* Jones & Boguraev (1987), pág. 65.

CORNELL, T. L. (1993), Description Theory, Licensing Theory, and Principle-Based Grammars and *Parsers. Dissertation Abstracts International* vol. 54, no. 2, August, 500-A.

CROCKER, M. (1996), *Computational Psycholinguistics - An Interdisciplinary Approach to the Study of Language*. Kluwer Academic Publishers.

CROCKER, M. W. (1991), A Principle-Based System for Syntactic Analysis. *Canadian Journal of Linguistics* 36 (1): 1-26. March.

FONG, S. (1992a), Computational Properties of Principle-Based Grammatical Theories. *Dissertation Abstracts International* vol. 52, no. 10, April, 5364-B.

FONG, S. (1992b), The Computational Inplementation of Principle-Based *Parsers*. In Berwick, R. C. & Abney, S. P. & Tenny, C. (Eds.), *Principle-Based Parsing: Computation and Psycholinguistics*. 65-82. Kluwer Academic Publishers.

GOODMAN, M. (1991), A case-based, inductive architecture for natural language processing. Unpublished paper presented at AAAI Spring Symposium on Machine Learning of Natural Language and Ontology. *apud* Kolodner (1993), pág. 608.

JONES, K. S. & Boguraev, B. (1987), A Note on a Study of Cases. *Computational Linguistics*, Volume 13, Numbers 1-2, January-June, 65-68.

JONES, D. B. (1993), The Processing of Natural Language by Analogy with Specific Reference to Machine Translation. *Dissertation Abstracts International* vol. 53 no. 8, February, 4218-B.

KOLODNER, J. (1993), *Case-Based Reasoning*. Morgan Kaufmann Publishers, Inc.

MARCUS, M. P. (1980), *A Theory of Syntactic Recognition for Natural Language*. The MIT Press.

MARTIN, C. E. (1989), Case-Based *Parsing*. In Riesbeck, C. K. & Schank, R. C. (Eds.), *Inside Case-Based Reasoning*. 319-372. Lawrence Erlbaum Associates, Publishers.

MCHALE, M. L. (1995), Combining Machine-Readable Lexical Resources with a Principle-Based *Parser*. *Dissertation Abstracts International*, vol. 57/02-A, page 491.

MERLO, P. (1995), Modularity and Information Content Classes in Principle-Based *Parsing*. *Computational Linguistics* vol. 21, Number 4, 515-541.

PEREIRA, F. C. N. & Grosz, B. J. (1993), "Introduction" (to the Special Volume on Natural Language Processing). *Artificial Intelligence* 63, 1-15.

PEREIRA, F. C. N. & Warren, D. H. D. (1980), Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence* 13, 231-278.

PUSTEJOVSKY, J. & Boguraev, B. (1993), Lexical Knowledge Representation and Natural Language Processing. *Artificial Intelligence* 63, 193-223.

STABLER JR., E. P. (1993), *Parsing* as non-Horn Deduction. *Artificial Intelligence* 63, 225-264.

WOODS, W. A. (1970), Transition Network Grammar for Natural Language Analysis. *Communications of the ACM*, vol. 13, no. 10, October, 591-606.

EDUCAÇÃO À DISTÂNCIA MEDIADA POR COMPUTADOR (EDMC) – PROJETO PEDAGÓGICO PARA UMA REDE DE PÓS-GRADUAÇÃO

COMPUTER MEDIATED DISTANCE LEARNING – POST GRADUATED INTEREGIONAL NET

Waldomir LOYOLLA*
Maurício PRATES*

RESUMO

Este trabalho procura mostrar que a Educação à Distância Mediada por Computador (EDMC) se adequa perfeitamente à implantação de cursos de pós-graduação, em particular em um país de grandes extensões territoriais como o Brasil. Como uma resposta a esta adequação, apresentam-se algumas diretrizes estratégicas de projeto para a implantação real de um curso de Mestrado em Informática (Gerenciamento de Sistemas), formatado como uma rede inter-regional para quatro Universidades Católicas: PUC-Campinas, Universidade Católica de Brasília (UCB), Universidade Católica de Goiânia (UCG) e Universidade Católica Don Bosco de Campo Grande (UCDB).

Palavra-chave : Educação à Distância; Mediação por Computador; Gerenciamento de Sistemas; Cursos de Pós-Graduação; Ferramenats Pedagógicas; Ferramentas Tecnológicas; Web; Internet.

ABSTRACT

This paper highlights that Computer Mediated Distance Learning fits very well for implementing graduated courses in countries that present large territorial extension, as Brazil does. As an answer to this suitability some sterategic project guidelines are presented for the real implementation of a Master Degree Course in Information Systems Management formated as an inter-regional academic network involving four ctholic universities: Pontifical Catholic University of Campinas (PUC-Campinas), Catholic University of Brasilia (UCB), Catholic University of Goiania (UCG) and Dom Bosco Catholic University of Campo Grande (UCDB).

Keywords: Distance Learning; Computer Mediation; Systems Management; Post-Gaduated Master Courses; Pedagogical Tools; Technological Tools; Web; Internet.

1. INTRODUÇÃO

A Educação à Distância é mais antiga do que parece, pois já contabiliza mais de um século de existência. Seus primórdios remontam ao ano de 1881 quando William Harper, primeiro reitor e fundador da Universidade de Chicago, ofereceu, com absoluto sucesso, um curso de Hebreu por correspondência. Em 1889 o Queen's College do Canadá deu início a uma série de cursos à distância, sempre registrando grande procura pelos mesmos devido, principalmente, a seu baixo custo e às grandes distâncias que separam os centros urbanos daquele país.

(*) Professores do Instituto de Informática da PUC-Campinas.

Daquela época em diante, a Educação à Distância foi sendo desenvolvida utilizando-se dos mais variados ferramentais pedagógicos possíveis, dependendo de fatores tais como: as características da escola e dos professores, o tipo de curso ministrado, da distribuição geográfica entre escola e alunos e, principalmente, a tecnologia disponível e a relação custo/benefício para o uso da mesma. Em função, principalmente, da tecnologia de transmissão de informação adotada, a evolução da Educação à Distância pode ser dividida em três fases cronológicas, ou gerações **[Saba 97] [Roberts 96]**. A primeira foi a **geração textual**, que se baseou no auto-aprendizado com suporte apenas em simples textos impressos, o que ocorreu até a década de 1960. A Segunda foi a **geração analógica**, que se baseou no auto-aprendizado com suporte em textos impressos intensamente complementados com recursos tecnológicos de multimídia tais como gravações de vídeo e áudio, o que ocorreu entre as décadas de 1960 e de 1980. A terceira é a atual **geração digital** que se baseia no auto-aprendizado com suporte quase que exclusivamente em recursos tecnológicos altamente diferenciados, que podem ser balizados pelos seguintes fatores **[Wilson 97] [Spodik 97]**:

- A eficiência e o baixo custo dos modernos sistemas de telecomunicação digital e via satélite;
- A alta interatividade e o baixo custo dos modernos computadores pessoais;
- A amplitude e o custo acessível das redes computacionais locais e remotas, tais como as intranets e a internet.

As novas tecnologias, altamente interativas, permitiram o surgimento dos sistemas de EDMC – Educação à Distância Mediada por Computador, que põem criticamente em cheque a eficiência pedagógica do sistema educacional convencional, baseado no uso exclusivo da sala de aula, totalmente síncrono, ou seja, exigindo presenças físicas e simultâneas de instrutor e alunos. O uso do ferramental pedagógico atualmente disponível pela EDMC permite o oferecimento de condições assíncronas de aprendizado, que podem, e devem, ser combinadas parcialmente com o ferramental do sistema convencional, este em menor escala, permitindo uma combinação estreita de grande flexibilidade e alta eficiência no aprendizado final. Uma outra particularidade do EDMC é que as modernas tecnologias, atualmente disponíveis, permitem o oferecimento de múltiplas combinações de ferramentas pedagógicas, modernas e tradicionais, com inegável e significativo melhoramento da relação custo/benefício de implantação e manutenção dos programas de pós-graduação nestes moldes.

Em todo o mundo, em particular nos países denominados de "primeiro mundo", a EDMC está em franca expansão, sendo largamente implantada por meio de programas de grande porte. Isto tem ocorrido com mais intensidade nos países de grande extensão territorial, como Canadá, Estados Unidos e Austrália, que estão na fronteira avançada do uso extensivo dos processos de EDMC, com um sem número de programas, a maioria promovido por suas melhores e maiores universidades e empresas. Alguns países da América Latina, como México e Venezuela, também possuem programas significativos de EDMC. **[Lewis & Romiszowski 97] [Klemm & Utsumi 97]**.

No Brasil o interesse pela educação à distância se destacou quando, no início dos anos 60, o Bispo D. Fernandes do Paraná convidou a Profª Eda de Souza, atualmente coordenadora da Cátedra UNESCO de Educação à Distância da Universidade de Brasília, a elaborar um projeto para educação à distância destinado à zona rural paranaense, segundo os moldes de outro já colocado em funcionamento por D. Eugênio Sales em Natal-RN. No Brasil ainda são poucas as iniciativas do emprego de EDMC, em particular na pós-graduação stricto-sensu, destacando-se neste mister as iniciativas da Universidade Federal de Santa Catarina (UFSC), da Pontifícia Universidade Católica de Campinas (PUC-Campinas) e da Universidade Católica de Brasília (UCB), estas duas motivo de relato no presente trabalho **[Gomes 97]**.

O presente trabalho procura mostrar a viabilidade da implantação de cursos de pós-graduação stricto sensu, baseado na EDMC. Na seção 2 identificam-se os principais projetos de EDMC em andamento em todo o mundo, e também no Brasil, mostrando o alcance, abrangência e importância do uso da EDMC como alternativa pedagógica. Na seção 3 discutem-se as principais tecnologias digitais necessárias para o uso da EDMC, relacionando-as com as ferramentas pedagógicas convencionais. O relato da implantação de uma rede de pós-graduação stricto sensu baseada no uso da EDMC é apresentado na seção 4.

2. FERRAMENTAL PEDAGÓGICO PARA EDMC

A composição pedagógica da EDMC não deve apenas resolver as questões das grandes distâncias. Deve também, e principalmente, buscar suprir as necessidades de interatividade do aluno com o tema de estudo bem como valer-se do ferramental tecnológico disponível como forma de aperfeiçoar os aspectos pedagógicos do ensino, permitindo cumprir os principais fatores de uma educação centrada no aprendizado interativo, dinâmico e contextualizado **[Spennemann 97]**.

Normalmente, o processo contextualizado do aluno passa pelas seguintes fases:

1ª) Recepção de idéias, conceitos e informações;

2ª) Re-concepção das idéias, integrando-as com sua própria experiência

3ª) Exemplificação das idéias, integrando-as com sua própria experiência

4ª) Generalização a partir das idéias já integradas

5ª) Geração de questões a partir das idéias já integradas

6ª) Conexão das idéias com o discurso

Levando em conta estes fatores, para efeito de análise comparativa as ferramentas pedagógicas básicas a serem referenciadas são as seguintes:

- textos didáticos;
- aulas expositivas;
- orientação de trabalhos de pesquisa;
- avaliação de trabalhos.

Nessa linha, as ferramentas pedagógicas a serem analisadas são divididas em dois grandes grupos, quais sejam, convencionais e não-convencionais. As ferramentas convencionais, largamente utilizadas no ensino tradicional, são as seguintes:

- textos didáticos em papel;
- aulas expositivas com presença pessoal simultânea de professor e alunos;
- orientação de pesquisas e dissertações com presença pessoal simultânea de orientador e um aluno por vez.
- avaliação de trabalhos e seminários com presença pessoal simultânea de professor e grupos de alunos.

As ferramentas não-convencionais, ou virtuais, utilizadas pela EDMC, podem ser também divididas em dois sub-grupos:

- **Essenciais** – que se caracterizam pela alta confiabilidade, pela facilidade de uso e pelo baixo custo;
- **Complementares** – que se caracterizam pelo diferencial tecnológico oferecido, que de maneira geral apresentam custos mais elevados.

São consideradas ferramentas **essenciais** de EDMC as seguintes:

- textos didáticos disponibilizados em Home Pages na Internet (sem animação);
- aulas expositivas disponibilizadas em Home Pages na Internet (com animação);
- orientação de pesquisas através de correio eletrônico (e-mail) e diálogo remoto (chat);
- avaliação de trabalhos e seminários através de correio eletrônico (e-mail) e diálogo remoto (chat).

São consideradas ferramentas **complementares** de EDMC as seguintes:

- textos didáticos gravados em CD-ROM (sem animação);
- aulas expositivas disponibilizadas em CD-ROM (com animação) e/ou sistemas de teleconferência;
- orientação de pesquisas e dissertações através de sistemas de vídeo ponto-a-ponto (VDp-p) e diálogo remoto (chat);
- avaliação de trabalhos e seminários através de sistemas de tele-conferência e/ou sistemas de vídeo ponto-a-ponto (VDp-p).

A implantação de cursos de EDMC deve considerar a viabilidade de uso da tecnologia em todos as fases: produção da informação, disponibilização da informação e obtenção da informação. O atual momento tecnológico tem disponibilizado os recursos necessários para suprir o ferramental pedagógico, em todas as fases do ensino, a uma relação custo-benefício muito vantajosa. A seguir discute-se o

ferramental tecnológico, tanto o essencial como o complementar, atualmente disponível para a elaboração de cursos por EDMC.

Para a produção de textos didáticos, atualmente dispõe-se de browsers e processadores de texto que operam com a linguagem HTML (Hyper Text Markup Language) que permite a elaboração de textos com grande capacidade de agregação de informação correlata através de hyperlinks. Alguns dos browsers mais utilizados atualmente, além de servirem para navegação pela Web são editores de documentos em HTML e podem ser obtidos gratuitamente, o que em muito barateia a criação e edição de textos didáticos usando esta linguagem. Além disto, muitos programas geradores de imagens animadas podem ser encontrados gratuitamente na rede, permitindo a edição de textos didáticos mais elaborados, inclusive utilizando de recursos gráficos que também podem ser disponibilizados através de sua inclusão em páginas elaboradas em linguagem HTML.

A produção de aulas expositivas, para disponibilização através de computadores, requer uma composição de elementos didáticos que devem ser integrados: parte expositiva e parte interativa. A parte expositiva corresponde à digitalização de certos períodos de vídeo em que o professor apresenta a estrutura da aula e um resumo de sua composição, elabora uma resenha do aprendido e propõe novos estudos. Esta parte expositiva é intercalada com a parte interativa, onde são apresentadas informações elaboradas com os mais diversos programas para composição de apresentações. A integração destas partes inclusive permite um ritmo mais intenso da aula, incentivando o aluno ao aprendizado e evitando períodos de pouco interesse por parte do aluno.

A interação pessoal entre aluno e professor ocorrida na orientação de pesquisas e mesmo na avaliação de outros trabalhos pode ocorrer de variadas formas. A forma mais tradicional, já largamente utilizada desde longa data, corresponde ao uso do correio eletrônico (e-mail), através do qual documentos podem ser transferidos de uma pessoa a outra. Recentes avanços ocorridos nos programas de correio eletrônico permitiram a inserção, no corpo de uma mensagem, de documentos gerados pelos mais variados programas de edição de texto, vídeo, apresentações, etc. Com isto aumentou-se a eficiência da troca de informações através deste meio de comunicação. Outros avanços recentes têm permitido também a interação de voz e vídeo ponto-a-ponto através de browsers operando na Web. Isto tem, em muito, ampliado a capacidade de interação entre professor e orientados. Há que se notar o fato que os citados browsers podem ser obtidos, em sua maioria, gratuitamente na própria Web. Para a disponibilização dos diferentes elementos didáticos do curso, (textos, aulas, avaliações, etc.) a Web apresenta-se como mecanismo indiscutível, com boa eficiência para este tipo de documento, com alta disponibilidade de acesso praticamente a qualquer interessado e com baixo custo.

Para a obtenção dos diferentes elementos didáticos do curso, (textos, aulas, avaliações, etc.) diversos tipos de browsers podem ser obtidos gratuitamente na rede, que permitem não só a visualização de textos editados nas mais variadas linguagens (não apenas HTML) mas até a visualização, e mesmo interações, com animações através da adição de diversos módulos, que também podem ser obtidos gratuitamente. Quanto às ferramentas complementares, estas representam alternativas mais dispendiosas, mas que também podem ser adotadas nas diferentes fases da implantação e execução de cursos por EDMC. O ferramental de EDMC, conforme discutido acima, pode ser sintetizado pela matriz de composição da Figura-1.

3. DIRETRIZES BÁSICAS DE PROJETO

A rápida evolução da ciência e da tecnologia tem levado os profissionais a buscar um aprimoramento constante e dinâmico, de forma a atender às exigências crescentes de um mercado de trabalho altamente competitivo. De forma geral, os cursos isolados e de curta duração, que procuram oferecer atualização tecnológica, não têm conseguido dar aos profissionais uma necessária formação sólida, com uma base consistente e versátil. No caso específico do setor acadêmico, instituições de ensino superior, buscando a excelência do ensino através da qualificação de seus mestres, têm demandado cursos de pós-graduação nas mais variadas áreas. Por outro

lado, o mercado de trabalho não-acadêmico apresentam uma demanda reprimida de recursos humanos.

Como ponto inicial, a proposta de um curso de pós-graduação, em particular daqueles a serem mediados por computador, deve demonstrar claramente um conjunto de justificativas que, adequadamente, localize o universo acadêmico e organizacional em que o curso se insere. Também devem ser esclarecidos os principais fatores que direcionam à adoção de uma difusão de conhecimento mediada por computador. Não apenas os aspectos geográficos devem ser apresentados mas também aqueles de âmbito pedagógico e tecnológico que incentivem a adoção de tal prática. A seguir, alguns destes fatores são abordados.

3.1 Fatores TEMPO e DISTÂNCIA

Na busca por cursos de pós-graduação, as instituições têm se defrontado com alguns sérios problemas para que seus Recursos Humanos possam se engajar nestes cursos. Dentre estes problemas pode-se citar ao menos dois que afligem a maioria das organizações: tempo e distância, conforme comentados a seguir. Com frequência os profissionais interessados em um curso de pós-graduação não dispõe de tempo para frequentar, com regularidade, às aulas de um determinado curso. A falta de regularidade às aulas pode acabar por levar o candidato ao abandono do curso, já que boa parte do curso pode ter sido perdida.

FERRAMENTAS PEDAGÓGICAS	CONVENCIONAIS	VIRTUAIS	
		ESSENCIAIS	COMPLEMENTARES
TEXTOS DIDÁTICOS	EM PAPEL	HOME PAGE (sem animação)	CD-ROM (sem animação)
AULAS EXPOSITIVAS	PRESENÇA PESSOAL DOCENTE+ALUNOS	HOME PAGE (com animação)	TELE-CONFERÊNCIA e/ou CD-ROM (c/anim.)
ORIENTAÇÃO DISSERTAÇÃO	PRESENÇA PESSOAL ORIENTADOR+ALUNO	E-MAIL e/ou CHAT	VÍDEO POINT TO POINT
AVALIAÇÃO SEMINÁRIOS	PRESENÇA PESSOAL DOCENTE+ALUNOS	E-MAIL e/ou CHAT	TELE-CONFERÊNCIA e/ou VÍDEO P-T0-P

Figura 1 - Matriz de composição do ferramental de EDMC

Um outro fator restritivo para a pós-graduação de muitos profissionais é a distância que separa sua base de trabalho de regiões onde existam cursos que preencham suas necessidades ou interesses. Em países onde as distâncias geográficas são grandes estas dificuldades são agravadas, principalmente quando não existe uma distribuição uniforme dos quadros de docentes para cursos de pós-graduação. A EDMC se apresenta, atualmente, como uma resposta eficaz e de relação custo/benefício extremamente favorável para o oferecimento de cursos de pós-graduação que eliminem, ou ao menos minimizem, as restrições mencionadas acima.

Há que salientar-se o fato que, nesta proposta, o ensino mediado pelo computador de maneira alguma prescinde nem da figura do instrutor, como regente do curso, nem de uma interação pessoal entre professores e alunos. O que se busca é uma valorização da interatividade maior do aluno com professor, colegas e informação útil para sua formação, e uma socialização maior do ensino através da possibilidade de maior difusão do conhecimento detido por professores e pesquisadores com reconhecido notório saber que, com frequência, não podem se deslocar por todo o país para formar novos profissionais.

Embora a atual legislação federal, através da Lei de Diretrizes e Bases da Educação, contemple todos os níveis educacionais com a possibilidade de Educação à Distância, a presente proposta adequa-se, em princípio, a cursos de pós-graduação onde os

fatores limitadores acima discutidos são frequentes e também onde se pode contar com um provável maior grau de responsabilidade e interesse do aluno pela informação a ser recebida.

3.2 Modulação e Estrutura Básica das Disciplinas

Baseando-se no ferramental necessário para a realização de EDMC e na significativa disponibilidade de recursos para sua implementação, propõe-se aqui uma modulação e estrutura básicas para que disciplinas de pós-graduação sejam oferecidas em um programa de EDMC. Esta modulação e estrutura busca uma integração dos recursos computacionais e de telecomunicações atualmente disponíveis de forma a prover educação à distância com a maior interatividade possível entre professores/alunos, alunos/alunos e alunos/informação, como requerido para um eficiente programa de ensino centrado no aprendizado. As disciplinas deverão obedecer um mesmo padrão de modulação e estrutura básica, baseados nos seguintes parâmetros:

P = presença pessoal simultânea de professor e alunos;

V = presença pessoal de alunos e presença virtual de professor através de sistema de videoconferência;

I = presença virtual de professor e alunos através de aulas gravadas em Home Page e/ou CD-ROM com possibilidade de animação;

O = orientação de trabalhos e pesquisas com presença virtual de aluno e professor através de sistemas de correio eletrônico, chat e vídeo ponto-a-ponto.

Cada disciplina será composta de duas partes, estrategicamente distribuídas, quais sejam:

- **Módulo 1** – Conceitos, onde serão apresentados e discutidos os conceitos teóricos e básicos do tema da disciplina;
- **Módulo 2** – Laboratório, onde os conceitos deverão ser aplicados em trabalhos, pesquisas e estudos orientados, individuais ou em grupo.

Considerando-se que cada hora-aula no período noturno corresponde a 45 minutos, cada aula padrão será modulada em 3 horas e 40 minutos, ficando cada disciplina parametrizada conforme a Figura-2.

4. EXEMPLO DE IMPLEMENTAÇÃO

A implantação da primeira fase do Projeto de Rede de Pós-Graduação foi iniciada em fevereiro de 98, quando uma opção de EDMC foi oferecida a alunos do Mestrado em Informática da PUC-Campinas, localizados a mais de 200 Km de Campinas (Cuiabá-MS, Uberaba-MG, Três Corações-MG e Jaú-SP). Os alunos freqüentam aulas presenciais numa proporção de ¼ em relação às aulas virtuais, realizadas por meio de uma combinação de aulas remotas síncronas (do tipo sala de discussão ou chat) e assíncronas (do tipo news-group) através da Internet.

O material didático de cada disciplina é disponibilizado em uma Home-Page especialmente projetada para o Curso de Mestrado em Informática (**http://www.mestrado.nar.puccamp.br**). A segunda fase do projeto teve início em agosto de 1998, com a implantação do Curso de Mestrado em Informática da UCB (**http://www.mestradoinfo.pos.ucb.br**) inspirado no programa da PUC-Campinas. As fases finais de envolvimento das outras duas Universidades estão previstas para 1999 e 2000. Os resultados obtidos no primeiro semestre de 1998 com o projeto têm demonstrado claramente a viabilidade operacional, econômica e pedagógica do sistema de EDMC como opção alternativa e complementar aos programas de pós-graduação presenciais.

4.1 Justificativas

A área de Informática tem sido caracterizada como uma daquelas em que os profissionais mais têm buscado aprimoramento constante e dinâmico, como forma de resposta à competitividade e dinamismo do mercado. Em especial nesta área, cada vez mais, tem sido requerida uma visão abrangente da informática em geral e dos processos de gestão de recursos de informática em particular, tanto para profissionais acadêmicos como aqueles do mercado não-acadêmico. No caso específico do setor aca-

dêmico da região centro-oeste do país, conta-se com dezenas de instituições de ensino superior que têm demandado cursos de pós-graduação na área de ciência da computação.

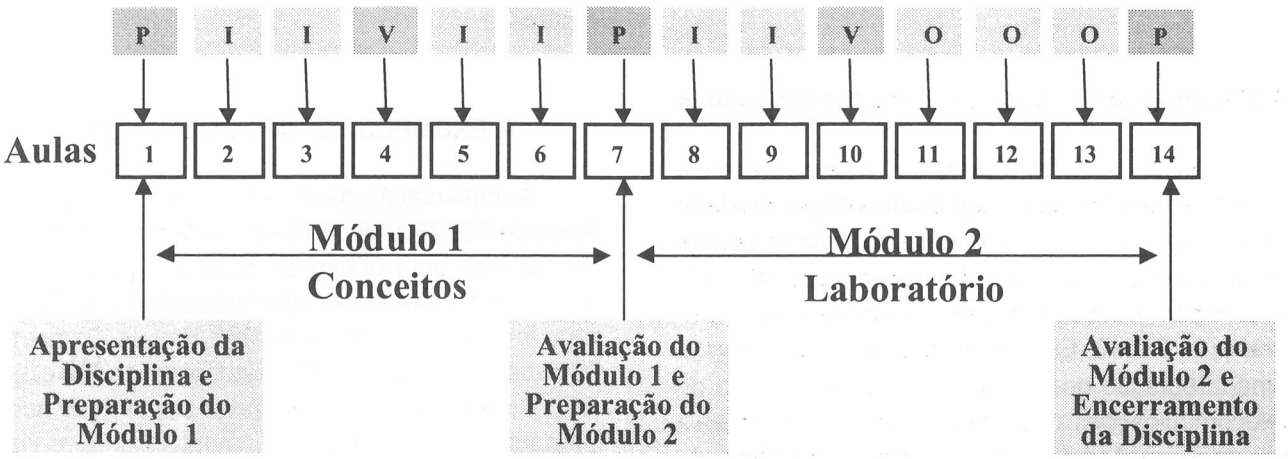

Figura 2 - Modulação das aulas em cada disciplina

No mercado de trabalho não-acadêmico da região, mais particularmente no Distrito Federal, situam-se os principais órgãos do governo e as sedes das grandes empresas estatais. Neste ambiente, apresenta-se uma demanda reprimida de recursos humanos capacitados para o exercício da alta gerência organizacional, que possam apresentar um sólido conhecimento da área da informática junto à capacitação para a solução de complexos problemas gerenciais e tecnológicos.

4.2 Objetivos

Diante das peculiaridades do mercado de trabalho de informática das regiões envolvidas, a Rede de Pós-Graduação, esquematizada na Figura-3, pretende disseminar o Curso de Mestrado em Informática, com o objetivo geral de formar profissionais altamente capacitados na Informática em geral e no Gerenciamento de Sistemas em particular. Dessa forma, os objetivos específicos da Rede de Pós-Graduação, envolvendo quatro Universidades Católicas, podem ser assim enumerados:

- Desenvolver a formação de recursos humanos com a qualificação especial para o exercício da alta gerência organizacional e tecnológica, na área da informática em geral e dos sistemas de informação em particular;
- Desenvolver a formação de recursos humanos com a qualificação especial para realização de pesquisas científicas e tecnológicas e docência de nível superior, na área da informática em geral e dos sistemas de informação em particular;
- Atingir um público alvo do interior do Estado de São Paulo, da região do Sul de Minas e da região Centro-Oeste que, por sua vinculação profissional, têm severas restrições geográficas e de horário para a participação presencial
- Desenvolver estudos e pesquisa na área de informática, visando o avanço dos conhecimentos científicos e tecnológicos, na área do desenvolvimento e da implementação de sistemas de Informação, assim como a integração destes conhecimentos com as mais modernas técnicas gerenciais;
- Desenvolver projetos para a aplicação de tecnologias de ponta que levem à otimização do desempenho profissional na área do gerenciamento de sistemas em organizações públicas e privadas.

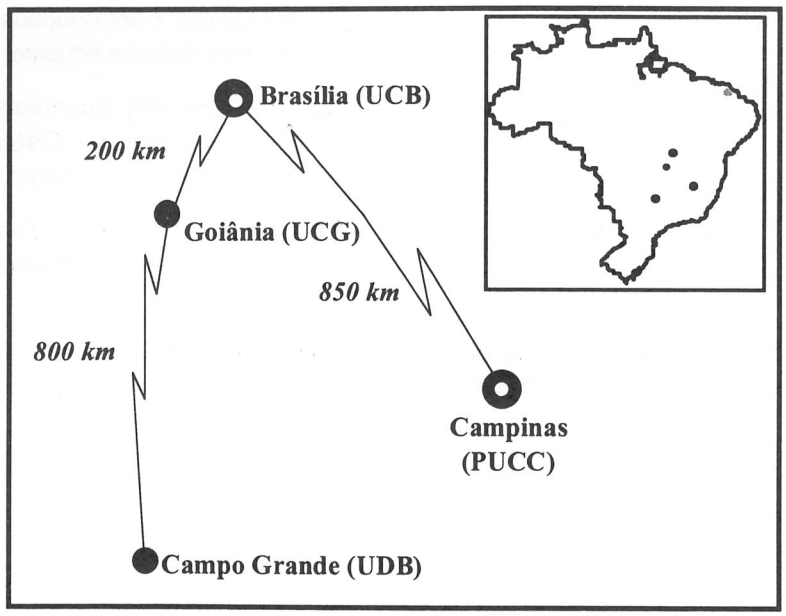

Figura 3 - Esquema da Rede Acadêmica de Pós-Graduação

5. CONCLUSÕES

A Educação à Distância Mediada por Computador - EDMC tem sido largamente utilizada nos mais variados níveis educacionais em várias partes do mundo. Nelas, este processo se instalou com vigor, principalmente em países de grandes dimensões como o Brasil, tais como Canadá, Austrália, Estados Unidos, México e China. O atual estágio das tecnologias de telecomunicação e computação tem, em muito, melhorado a relação custo/benefício para a implantação deste tipo de mecanismo de ensino.

No Brasil ainda são poucos os projetos e programas de EDMC ou afins, e as pressões por ofertas de Pós-Graduação e Educação Continuada são as mesmas e em condições similares àquelas existentes em países onde mais tem florescido a EDMC. Tais fatores acumulados permitem a avaliação que, no Brasil, o uso de EDMC deve crescer de forma vertiginosa, ainda mais quando se tem o estímulo explícito como o da nova LDB – Lei de Diretrizes e Bases da Educação vem dar à EDMC através de seu Artigo n.º 80, onde fica bem explicitado que a Educação à Distância deve se dar "em todos os níveis" da educação **[Demo 97] [Monlevade 97]**. Nessa linha, a experiência que está em curso, da implantação de uma Rede Acadêmica de Pós-Graduação envolvendo quatro Universidades Católicas, vem de encontro às necessidades do mercado profissional e aos ditames das autoridades educacionais do País.

A proposta aqui apresentada se adequada perfeitamente à implantação de EDMC para programas de pós-graduação. Neste projeto da rede de Pós-Graduação pôde-se confirmar uma relação custo/benefício extremamente favorável, uma vez que estimado um completo retorno do investimento praticamente ao final de apenas uma turma (dois anos) e a partir daí uma estimativa de retorno por volta de 50% em relação aos custos de manutenção do curso.

REFERÊNCIAS BIBLIOGRÁFICAS

[Demo 97] Demo, P. *A Nova LDB – Ranços e Avanços*, 111 pags., Papirus, Campinas, 1997

[Gomes 97] Gomes, M.T.. *Seja um Alunauta e Garanta seu Futuro*, http://www.2.uol.com.br/exame, 02 pags., 04/07/97

[Gomide 96] Gomide, S. *Volta às Aulas Virtual no Brasil*, Internet World, março de 1996, 62-67

[Klemm & Utsumi 97] Klemm, W.R. & Utsumi, T. *Affordable and Accessible Distance Education: A Consortium Iniciative*, http://www.usq.edu.au, 06 pags., 14/07/97

[Lewis & Romiszowski 97] Lewis, J.H. & Romiszowski, A. *Networking and the Learning Organization: Issues and Scenarios for the 21st Century,* http://www.usq.edu.au, 14 pags., 14/07/97

[Monlevade 97] Monlevade, J. *Educação Pública no Brasil – Contos e Descontos*, 191 pags., Idéia, Brasília, 1997

[Roberts 96] Roberts, J.M. *The Story of Distance Education: A Practitioner's Perspective*, Journal of the American Society for Information Science, 47 (11): 811-816, 1996

[Saba 97] Saba, F. *Introduction to Distance Education*, http://www.distance-educator.com, 04 pags., 14/07/97

[Spennemann 97] Spennemann, D.H.R. *On-Line Study Packages for Distance Education*, http://www.csu.edu.au, 15 pags., 15/07/97

[Spodik 97] Spodik, E.F., *The Evolution of Distance Learning–4. Tools Available for Distance Education*, http://sqzm14.ust.hk, 02 pags., 14/07/97

[Wilson 97] Wilson, J.M. *Distance Learning for Continuous Education*, http://www.educon.edu, 05 pags., 14/07/97

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS
INSTITUTO DE INFORMÁTICA

CHAMADA DE TRABALHOS

REVISTA DO INSTITUTO DE INFORMÁTICA

ARTIGOS TÉCNICOS - CIENTÍFICOS - OPINIÕES TENDÊNCIAS

MAIORES INFORMAÇÕES E SOLICITAÇÃO DO FORMATO PARA APRESENTAÇÂO DE TRABALHOS

ii@zeus.puccamp.br
frank@zeus.puccamp.br

ou

Correspondência: Rodovia D. Pedro I km 136
Caixa Postal 317 CEP 13020-904 Campinas SP.
BRASIL
Att. Marilda dos Reis Gomes
Tel.: (0XX19) 756-7094 Srtª Marilda
(0XX19) 756-7195 Prof. Frank Behrens

Revista do Instituto de Informática
Publicação Semestral do Instituto de Informática
PUC-Campinas

NORMAS AOS COLABORADORES

1 - Serão aceitos trabalhos técnicos, científicos, tendências e opiniões;

2 - Os artigos deverão conter, sequencialmente:
- Título em português;
- Título em inglês;
- Nome(s) do(s) autor(es) (colocados por extenso, o último nome maiúsculo, e seguidos de*, para especificações profissionais do(s) autor(es) no rodapé da primeira página);
- Resumo e Abstract (máximo de 200 palavras);
- Introdução (precedida do número 1);
- Corpo do artigo (itens numerados seqüencialmente, a partir do número 2);
- Conclusões;
- Referências Bibliográficas (conforme utilizado nesta edição); Desenhos, gráficos e fotografias serão denominados Figuras, numerados sequencialmente (algarismos arábicos) e constantes do corpo do trabalho. As tabelas serão denominadas Tabelas, numeradas sequencialmente (algarismos arábicos) e constantes do corpo do trabalho;

3 - Os trabalhos, digitados com, no máximo, 30 000 caracteres (aproximadamente dez páginas), deverão ser elaborados em Microsoft Word;

4 - Os trabalhos poderão ser apresentados em três cópias impressas, devendo constar a identificação do(s) autor(es) em uma folha a parte, para permitir a avaliação dos mesmos pelo Conselho Editorial. Os trabalhos, uma vez aceitos, deverão ser encaminhados como descrito no item 3.;

5 - Os trabalhos podem ser enviados em disquete ou pela rede, anexos a uma mensagem indicando o editor de textos usado;

6 - Os trabalhos serão publicados após pareceres favoráveis de membros do Conselho Editorial da Revista;

7 - Quaisquer outros esclarecimentos poderão ser feitos pelo Editor - executivo da Revista;