ISSN 0104-8651

REVISTA DO INSTITUTO DE

INFORMÁTICA

PUCCAMP - VOLUME 2 - Nº 1 MARÇO/SETEMBRO/1994

CONTEÚDO/CONTENTS

Artigos/Articles

Revista do Instituto de Informática

Publicação Semestral

Editora-Executiva: Profª M. Cristina L. F. M. Aranha

Conselho Editorial:

Prof. Otávio R. Jacobini (PUCCAMP) - Presidente
Profª Angela de M. Engelbrecht (PUCCAMP) - Vice-Presidente
Dr. Arthur J. Catto (FCTI)
Drª Beatriz M. Daltrini (UNICAMP)
Dr. Carlos Mammana (FCTI/UNICAMP)
Dr. Edmundo R. M. Madeira (UNICAMP)
Dr. Eduardo O. C. Chaves (UNICAMP/PUCCAMP)
Dr. Geraldo Nonato Telles (UNICAMP/PUCCAMP)
Drª Geraldina Porto Witter (PUCCAMP)
Dr. Heitor Quintella (IBM Brasil)
Dr. Hilton S. Pinto (UNICAMP)
Dr. José Carlos Maldonado (USP)
Dr. José M. da Mata (UFMG)
Dr. Júlio S. Aude (UFRJ)
Dr. Luiz F. B. Baptistella (TELEBRAS)
Dr. Luiz Fernando Soares (PUCRJ)
Dr. Manuel J. Mendes (UNICAMP/PUCCAMP)
Dr. Marcio Luiz de Andrade Netto (UNICAMP)
Dr. Mario Jino (UNICAMP)
Dr. Mario L. Cortes (TELEBRAS/UNICAMP)
Dr. Maurício Magalhães (UNICAMP)
Dr. Maurício Prates (UNICAMP/PUCCAMP)
Dr. Nelson J. Parada (UNICAMP/PUCCAMP)
Dr. Saul G. D'Ávila (UNICAMP)
Drª Vera Sílvia M. Beraquet (PUCCAMP)
Dr. Wanderley L. de Souza (UFPb)

Conselho Consutivo

Profª M. Cristina L. F. M. Aranha
Prof. José Oscar F. de Carvalho
Dr. Maurício Prates
Odair M. da Silva
Prof. Orandi Mina Falsarella

Capa

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

Correspondência:

A/C:
Instituto de Informática - PUCCAMP
C. P. 317 - Campus I - Rod. D. Pedro I - Km 136 - CEP: 13020-904 - Campinas - SP - FAX: (0192) 52.8477

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 PUCCAMP

ISSN 0104 - 4869

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

INSTITUTO DE INFORMÁTICA

REVISTA DO INSTITUTO DE INFORMÁTICA DA PUCCAMP

Rev. Inst. Inform. PUCCAMP	Campinas	V. 2	n. 1	p. 1-60	março/set 1994

EDITORIAL

As etapas a serem vencidas para o lançamento de uma revista são inúmeras, mas vão sendo executadas gradativamente e a motivação para cumprí-las é reforçada pelas boas notícias que recebemos. Para esse número que está sendo lançado, a Revista recebeu o ISSN - Número Internacional Normalizado para Publicações Seriadas, fornecido pelo Instituto Brasileiro de Informação em Ciência e Tecnologia - IBICT. Esse número irá identificar e individualizar a Revista do Instituto de Informática da PUCCAMP além de garantir a inclusão da Revista em um banco de dados de publicações seriadas editadas em diversos países.

A publicação deste semestre inclui sete artigos técnico-científicos que apresentam diversos assuntos.

Na área de Gerenciamento de Sistemas de Informação, o artigo "Conceituação de Sistemas de Informação do Ponto de Vista do Gerenciamento" mostra a combinação de diversos aspectos relacionados aos sistemas de informação, que auxiliam na obtenção dos objetivos organizacionais. No artigo "Impacto da Tecnologia no Estilo Gerencial de Gerentes de Sistemas de Informação", o autor enfatiza um tipo de estrutura organizacional que auxilia a iniciativa e a criatividade dos profissionais da área.

Na área de desenvolvimento de software, o autor do artigo "Sistemas Gerenciadores de Bancos de Dados Orientados a Objetos para Ambientes de Desenvolvimento de Software" faz considerações sobre ambientes de desenvolvimento de software e o apoio que sistemas gerenciadores de bancos de dados orientados a objetos podem oferecer a esses ambientes.

No artigo "POKE-TOOL - Uma Ferramenta para Suporte à Aplicação dos Critérios Potenciais Usos para Teste de Programas", os autores descrevem uma ferramenta que auxilia as atividades de teste de software comentando a sua operação.

Em "Um Modelo de Interface de Aplicação para Ambientes de Automação Industrial", os autores enfatizam aspectos ligados a redes industriais, mostram uma implementação desenvolvida na UNICAMP e discutem alguns aspectos associados a essa implementação.

Para mostrar como determinar um mix de produção de maior lucratividade e as variáveis envolvidas, o artigo "OPTMIX - Um Modelo para Obtenção do MIX Ótimo de Produção" apresenta um modelo, desenvolvido por professores da PUCCAMP, usando programação linear.

O artigo "Processamento de Linguagem Natural: Atribuição de Caso aos Constituintes das Sentenças", é um trabalho desenvolvido com a participação de alunos do Instituto de Informática, mostrando um sistema que faz a análise semântica de uma frase declarativa, discutindo os aspectos da abordagem utilizada.

Agradecemos as sugestões recebidas que irão auxiliar na publicação dos próximos números. A nossa expectativa é a de receber, cada vez mais, a colaboração das pessoas que querem contribuir para a melhoria da qualidade de nosso trabalho.

Profª M. Cristina L. F. M. Aranha
Editora Executiva

CONTEÚDO/CONTENTS

Artigos/Articles

CONCEITUAÇÃO DE SISTEMAS DE INFORMAÇÃO (S.I.) DO PONTO DE VISTA DO GERENCIAMENTO

UNDERSTANDING INFORMATION SYSTEMS (IS) FROM A MANEGEMENT POINT OF VIEW

Prof. Dr. Mauricio PRATES*

ABSTRACT

An Information System (IS) can be seen by a manager as a structured combination of Information Technologies, Work Practices, People, and Information, organized as connected components in such a way to accomplish Goals in an Organization. Work Practices can be considered as the methods used by People to perform work. Information can include formatted data, text, pictures and sound. Except in the cases where they totally automate a task, IS include People who enter, process and/or use data. Information Technology includes hardware and software that perform data processing tasks such as capturing, transmitting, storing, retrieving, manipulating, or displaying data.

KEY WORDS: Information Systems (IS), Work Practices, Information Technology, Organization Goals, and Management.

RESUMO

Um Sistema de Informação (S.I.) pode ser visto por uma ótica gerencial como uma combinação estruturada de Tecnologias de Informação, Práticas de Trabalho, Recursos Humanos, e a Informação propriamente dita, organizados como componentes conectados de tal forma a permitir que sejam alcançados os Objetivos Organizacionais em questão. As Práticas de Trabalho podem ser consideradas como os métodos utilizados pelos Recursos Humanos para desempenharem suas tarefas. A Informação pode consistir de dados formatados, textos, imagens e som. A menos que sejam totalmente automatizados, os S.I. incluem os Recursos Humanos que coletam, processam e utilizam os dados. As Tecnologias de Informação consistem do hardware e do software que executam as tarefas de processamento dos dados, tais como capturar, transmitir, estocar, recuperar, manipular ou exibir dados concernentes.

PALAVRAS-CHAVE: Sistemas de Informação (S.I.); Práticas de Trabalho; Tecnologia de Informação; Recursos Humanos; Objetivos Organizacionais; Informação; Gerenciamento.

1. INTRODUÇÃO

Os S.I. afetam de variadíssimas formas o desempenho das modernas organizações empresariais ou públicas, abrindo-lhes, inclusive, novos espaços e oportunidades de atuação, motivo pelo qual os gerentes e administradores não podem se furtar a compreender sua natureza e a utilizar seus recursos com eficácia. O leque de novas oportunidades que os S.I. trazem começa com a melhoria e otimização das operações internas da organização, indo até suas operações externas, auxiliando na competitividade vantajosa através de benefícios diretos aos clientes ou usuários. Por outro lado, os S.I. também podem acrescentar novos riscos ao desempenho organizacional, tais como vulnerabilidade a acidentes e vandalismo, e ao uso estratégico de S.I. pelos competidores.

Assim, os S.I. podem ser conceituados, do ponto de vista do seu gerenciamento, como uma combinação estruturada de Informação, Recursos Humanos, Tecnologias de Informação e Práticas de Trabalho, organizados de tal forma a permitir o melhor atendimento dos

(*) Professor Titular da F. E. M/UNICAMP e do I.I./PUCCAMP

Objetivos da Organização. Esta conceituação é bastante abrangente, mas tem a vantagem de levar a uma compreensão maior sobre os usos e limitações dos sistemas convencionais e tradicionais de negócios ou serviços, assim como sobre os possíveis impactos de futuras inovações no comportamento organizacional. De acordo com esta conceituação, os seguintes exemplos podem ser considerados como S.I. consistentes(1):

. Um Sistema Interativo usado por gerentes e executivos de alto escalão para o monitoramento das operações de sua empresa;

. Um Sistema Biométrico que garante acesso seletivo a setores de segurança da organização, através da leitura e análise de impressões digitais ou de vasos sangüíneos da retina ocular;

. Um Sistema de Reservas de Companhia Aérea usado por agentes de viagem para marcação de vôos dos seus clientes;

. Um Sistema de Videoconferência usado para ajudar a manter coordenados os gerentes de marketing geograficamente dispersos de uma mesma empresa.

2. OS COMPONENTES DO S.I.

A Figura-1 procura expandir o conceito de S.I., mostrando esquematicamente as correlações existentes entre os objetivos organizacionais e os componentes informação, práticas de trabalho, recursos humanos e tecnologias de informação. A figura enfatiza que o ponto focal para o entendimento da natureza do S.I. são as práticas de trabalho, e não as tecnologias de informação como pode parecer à primeira vista. A chave para a conceituação está na forma através da qual os componentes interagem para criar práticas de trabalho que atendam convenientemente aos objetivos da organização. A figura também fornece um formato básico para desenvolver a configuração de qualquer tipo ou exemplo de S.I.. Vejamos em algum detalhe os elementos dessa configuração:

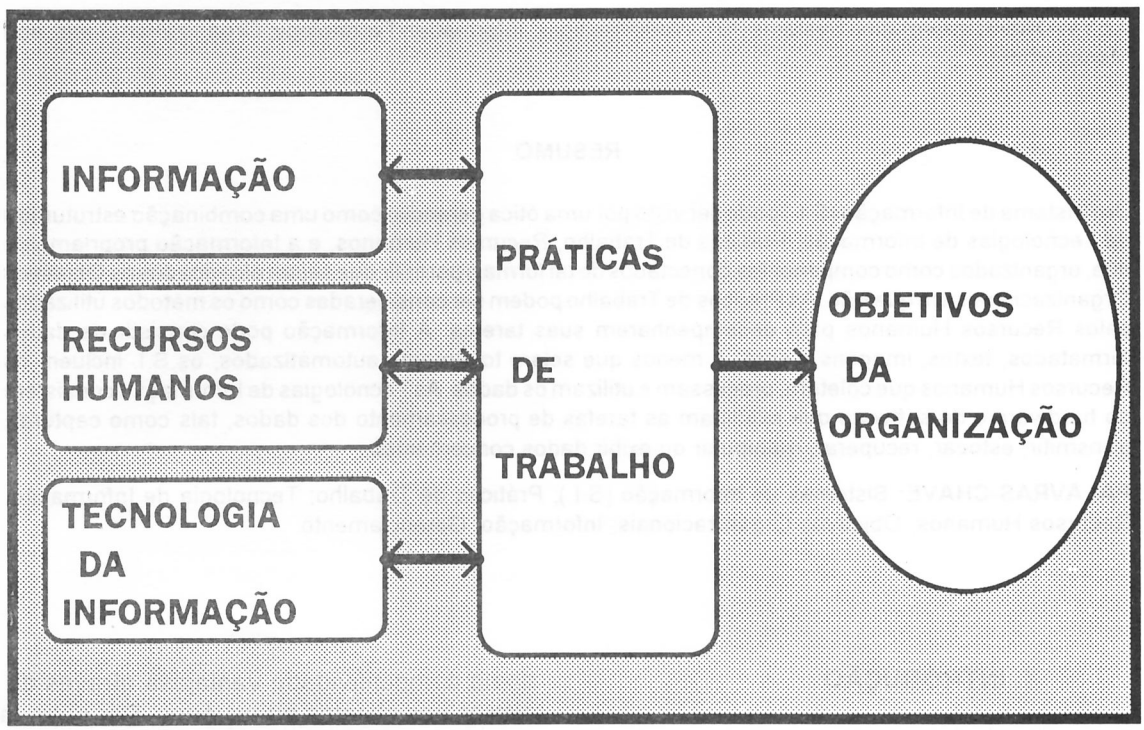


Figura 1 - Representação dos Elementos e Componentes do S.I.

2.1 - Os Objetivos Organizacionais

Os objetivos organizacionais não são exatamente um componente do S.I., mas exercem um papel vital na determinação das práticas de trabalho. São os objetivos que provêem os critérios básicos para se decidir como e quando as práticas de trabalho da organização devem ser alteradas e adaptadas. Outros determinantes de importância são as características dos recursos humanos, a informação disponível e as tecnologias de informação em uso, além de outros fatores externos ao sistema. O lançamento de um produto novo no mercado, por exemplo, pode motivar rapidamente uma variedade de práticas de trabalho, combinando sistemicamente os

recursos humanos, as informações e as tecnologias de informação.

2.2 - As Práticas de Trabalho

As práticas de trabalho consistem nos métodos utilizados pelos recursos humanos para desempenharem suas atividades no S.I., incluindo, não somente os procedimentos descritos pelos manuais de operação, mas também as linhas de ação nas quais os recursos humanos se coordenam, se comunicam e tomam decisões, realizam negócios ou serviços e desempenham outras tarefas. Note-se que se usa o termo "práticas de trabalho" e não "procedimentos", implicando que os S.I. incluem, tanto as atividades orientadas aos procedimentos, bem como as atividades orientadas aos recursos. As atividades orientadas ao procedimento são aquelas utilizadas para tarefas repetitivas e rotineiras, tais como a emissão de ordens de compra e de faturas, ao passo que as atividades orientadas ao recurso, tais como planilhas de cálculo e pacotes gráficos, servem para dar suporte aos recursos humanos na comunicação e na tomada de decisões.

A Figura-1 revela o fato de que as práticas de trabalho centralizam todas as conexões com os outros elementos, ilustrando que o S.I. só existe no contexto das atividades que os recursos humanos desempenham na organização. A centralização das práticas de trabalho nos S.I. explica como elas operam e quais as razões de seus sucessos ou falhas. Por exemplo, a diferença principal entre esforços bem ou mal sucedidos para impor o uso de microcomputadores nas atividades empresariais, reside se eles são ou não incorporados às práticas de trabalho da organização. Em práticas de trabalho tais como o monitoramento de progressos em projetos programados, os microcomputadores são parte ativa do S.I., caso contrário, eles irão apenas ocupar espaço ocioso nas mesas de trabalho(2).

2.3 - A Informação

Dados formatados e textos fazem parte integrante dos S.I.. Os dados são fatos, imagens ou sons que podem ou não ser pertinentes e utilizáveis para uma tarefa em particular. A informação pode ser considerada como um conjunto de dados cuja forma e conteúdo são apropriados para uma utilização em particular. A Figura-1 deixa claro que a vinculação entre as práticas de trabalho e a informação funciona biunivocamente, ou seja, nos dois sentidos (como um flecha dupla), indicando que as práticas determinam as necessidades de informação, ao passo que a disponibilidade de dados é que vai determinar quais as práticas viáveis para uma determinada aplicação.

Cabe lembrar que os primeiros S.I. informatizados continham apenas textos e dados numéricos. Nos anos 1980, entretanto, ocorreram progressos extraordinários na informática, surgindo as modernas tecnologias de informação, e hoje os S.I. podem utilizar também imagens e sons de forma então insuspeitada.

2.4 - Os Recursos Humanos

Com exceção dos casos em que as tarefas são totalmente automatizadas, os S.I. deverão incluir, necessariamente, recursos humanos que coletam, processam, recuperam e utilizam dados. Também aqui, a Figura-1 mostra que a vinculação entre os recursos humanos e as práticas de trabalho são biunívocas, ou seja, as práticas afetam os recursos humanos, ao passo que as características dos recursos humanos no sistema vão determinar quais práticas serão viáveis e adequadas.

Em geral, o desenvolvimento e a implementação do S.I. provocam impactos sobre os recursos humanos envolvidos, e essa é uma situação de difícil gerenciamento. As oportunidades pessoais podem ser provocadas, por exemplo, através da aplicação de tecnologias de informação às práticas de trabalho, tornando as atividades mais motivadoras e desafiadoras. Por outro lado, esse tipo de iniciativa deve ser tomada de forma dosada e otimizada, para que não venha a desvalorizar a experiência profissional ou tornar as tarefas tediosas ou obsoletas.

2.5 - As Tecnologias de Informação

Entende-se por tecnologias de informação o conjunto de hardware e software que desempenha uma ou mais tarefas de processamento das informações do S.I., tal como coletar, transmitir, estocar, recuperar, manipular, e exibir dados. Aí podem estar incluídos microcomputadores (em rede ou não), mainframes, scanners de código de barra, estações de trabalho, software de execução, software de planilhas eletrônicas ou de banco de dados, etc. É preciso ficar claro que a tecnologia de informação só é importante na medida que seja considerada apenas como um dos componentes do S.I., pois entender as tecnologias de informação não é o mesmo que entender o S.I. como um todo.

Um problema freqüente no desenvolvimento, implantação e gerenciamento de S.I. empresariais reside na tendência distorcida dos staff técnicos em supervalorizar as tecnologias, o que colide frontalmente com a visão dos usuários e clientes, bastante focalizada

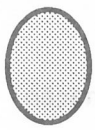
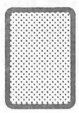
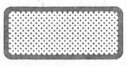
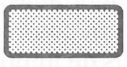
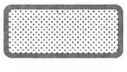
nas práticas de trabalho. Esta dicotomia pode estorvar significativamente a saudável e necessária comunicação entre os implantadores de S.I. e seus usuários, sendo uma das mais ocorrentes causas de insucessos e de falhas do S.I.

3. UMA HISTÓRIADE DE CASO EXEMPLAR

É bastante ilustrativo o caso relatado por Alter(1) a respeito da Hitachi Ltd., uma empresa multinacional fornecedora de produtos e serviços de alta tecnologia. Recentemente, a empresa adotou tecnologias de CAD (Computer Aided Design) de última geração, com o objetivo de melhorar seus processos de projeto e construção de usinas de força. Os sistemas de CAD anteriormente utilizados permitiam que os engenheiros e projetistas criassem desenhos dos componentes e que localizassem um determinado componente em relação aos outros componentes do projeto. Entretanto, dada a alta complexidade desse tipo de projeto, tornava-se necessária a geração de quantidades enormes de desenhos impressos em papel, assim como a construção de gigantescas maquetes em escala, que ocupavam áreas equivalentes a quadras de basquetebol ou mais. Tais modelos em escala tornavam-se necessários para a visualização do projeto como um todo e para a discussão com os clientes a respeito de eventuais modificações específicas. Apesar disso, os modelos eram considerados insuficientes para simular problemas rotineiros como, por exemplo, as dificuldades de inspeção em válvulas de pressão. Dessa forma, pode-se estabelecer um S.I. para este caso, conforme explicitado na Tabela-1.

Os novos equipamentos de CAD permitiam que os engenheiros projetassem qualquer componente específico e então visualizassem, através de imagens fotorealísticas, uma representação dinâmica do componente e de suas correlações com os outros variadíssimos componentes. O sistema permitia também a simulação de componentes móveis através de animação de imagens, o que permitia aos projetistas a detecção preventiva de falhas, evitando assim as crises de atrasos no cronograma acertado com os clientes. Era possível, por exemplo, simular a entrada de um técnico de manutenção no modelo da usina de força e testar as dificuldades de detecção de uma válvula específica a ser inspecionada. Em suma, com a utilização deste sistema, a empresa economizou 250.000 horas de engenharia, cortou significativamente os custos, diminuiu o tempo de teste de 18 para 9 meses, e tornou desnecessária a confecção de modelos em escala e a excessiva produção de desenhos em papel(3).

Tabela 1 - Componentes do S.I. no Caso Hitachi

	OBJETIVOS DA ORGANIZAÇÃO	> Redução de Custos > Melhoria da Qualidade > Eliminação dos Atrazos
	PRÁTICAS DE TRABALHO	> Utilização de Imagens Fotorealísticas em 3D por Dados Gerados no CAD
	TECNOLOGIA DA INFORMAÇÃO	> Estações de Trabalho CAD > Software Avançado de CAD
	RECURSOS HUMANOS	> Engenheiros Operando CAD > Clientes Avaliando Outputs
	INFORMAÇÃO	> Dados Sobre Correlações de Componentes do Projeto

4. MOTIVAÇÕES PARA A MUDANÇA CONTINUADA

Os S.I., em geral, não são sistemas estáticos, o que é reconhecido nas modernas organizações onde as práticas de trabalho estão em constante mudança para manter a necessária competitividade. A Figura-2 procura ilustrar como a dinâmica do S.I., feita principalmente através das mudanças das práticas de trabalho, é dirigida por novas idéias, novos problemas e de forças competitivas, ambientais e tecnológicas. Tais forças criam novos objetivos organizacionais, os quais vão requerer novas práticas de trabalho que, por sua vez, requerem adaptações por parte dos recursos humanos além de novas informações disponíveis e do desenvolvimento de novas tecnologias de informação(4).

Trata-se, na verdade, de um processo cíclico de mudança continuada. As mudanças e adaptações que envolvem os recursos humanos, a informação e as tecnologias de informação, afetam diretamente as práticas de trabalho, e então reverberam para os outros componentes do S.I., transformando-o(5). A natureza dinâmica dessas intrincadas relações é uma das razões que tornam complexo e arriscado o desenvolvimento e a implementação dos S.I.. Isso deixa evidenciado que o S.I. envolve muito mais que meramente a tecnologia.

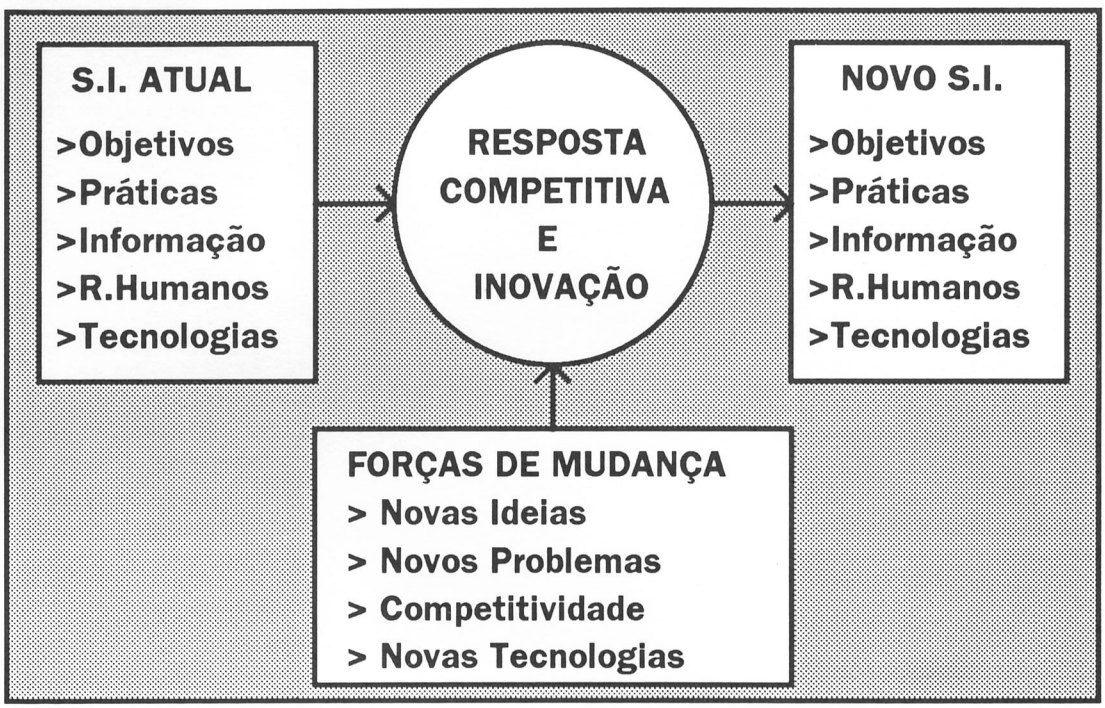


Figura 2 - Fatores de Resposta Competitiva para Inovação e Mudança

5. CONCLUSÕES

O S.I. é atualmente parte indispensável das modernas organizações, de tal forma que os gerentes de hoje precisam aprender muito sobre os S.I., cujo gerenciamento deve ser visto como a motivação central de suas carreiras profissionais. Os gerentes atuais devem estar preparados para identificar e configurar o S.I. na organização, pensar sobre como o S.I. pode afetar a organização, e decidir como tirar o melhor proveito do S.I. em benefício da organização. Alguns gerentes encontrarão oportunidades profissionais e pessoais neste campo, outros lidarão confortavelmente com o S.I. como um fato inevitável que carreia benefícios e custos, e ainda outros gerentes terão a visão imobilista de que os S.I. são arriscados e restritivos.

REFERÊNCIAS

(1) ALTER, S. - "Information Systems-A Management Perspective". Addison Wesley Pub. Co., New York, 1992, pp. 117-156.

(2) IVES, B. e Learmonth, G. - "The Information System as a Competitive Weapon". Communications of the ACM, vol. 27, nº 12, dez. 1984, pp. 1193-1201.

(3) JOHNSTON, H. R. e Vitale, M. R. - "Creating Competitive Advantage with Interorganizational Information Systems". MIS Quarterly, jun. 1988, pp. 153-165.

(4) RPCKART, J. e Morton, M. - "Implications of Changes in Information Technology for Corporate Strategy". Interfaces, vol. 14, nº 1, jan./fev. 1984, pp. 84-95.

(5) BENJAMIN, R. e Morton, M. - "Information Technology, Integration and Organizational Changes". Interfaces, vol. 18, nº 3, Mai./J un. 1988, pp. 86-98.

IMPACTO DA TECNOLOGIA NO ESTILO GERENCIAL DE GERENTES DE SISTEMAS DE INFORMAÇÃO

IMPACT OF TECHNOLOGY ON THE MANAGEMENT STYLE OF INFORMATION MANAGEMENT SYSTEMS

Prof. Dr. Silas Marques de OLIVEIRA*

ABSTRACT

Approaches the impact of technology on the management style of information systems directors. Argues that an open organizational structure is more compatible with the requirements and changes caused by the introduction of new informational technology. Information professionals need to work in an environment where iniciative and criativity are encouraged. Identifies trates that can promote criativity and iniciative in people and presents management concepts that managers need to adopt in order to stimulate a criative environment at the information unit.

KEY-WORDS: Thechnological Impact; Information Management Systems; Creativity; Management Style.

RESUMO

Aborda o impacto da tecnologia no estilo gerencial de diretores de sistemas de informação. Sugere que a estrutura organizacional aberta é mais compatível com as exigências e mudanças provocadas pela introdução de novas tecnologias informacionais. Os profissionais da informação precisam trabalhar em um ambiente que estimule e incentive a iniciativa e criatividade. Identifica traços que promovem a criatividade e inovação nas pessoas, e apresenta conceitos administrativos que os gerentes de sistemas de informação precisam adotar para estimular um ambiente criativo e adaptável às mudanças.

PALAVRAS-CHAVE: Impacto Tecnológico; Gerência de Sistemas de Informação; Criatividade; Estilo Gerencial.

Uma nova consciência social está surgindo no ambiente organizacional. Trabalhadores estão se tornando mais ecléticos, independentes e treinados. Mais do que nunca, se reconhece o direito de cada ser humano à liberdade de pensamento e expressão, a participar do processo democrático e a atingir todo o seu potencial. Mesmo assim, a existência do homem continua sendo afetada por racionamento de energia, instabilidade econômica e social, dilemas com respeito ao aproveitamento e uso do meio ambiente, e mudanças dramáticas no cenário político e comércio internacional. Estas condições complexas dificultam a percepção que um cidadão desenvolve do mundo, da vida, e ofusca sua fé na tecnologia como um instrumento de melhoria da condição humana.

A maioria das pessoas, de acordo com MARCHANT & ENGLAND (1989: 477) sente que tecnologia afeta suas vidas de uma das duas formas:

(1) Ou "tecnologia é prejudicial, pois sugere (a) um gasto de recursos, (b) organização centralizada, (c) perda de liberdade pessoal e dignidade, (d) desigualdade e consumismo, (e) trabalhos desqualificados, e (f) desemprego;"

(2) Ou "tecnologia é benéfico pois sugere (a) maior liberdade pessoal, (b) democracia participativa, (c) mais tempo para recreação, (d) maior conhecimento, e (e) melhoria na qualidade de vida."

(*) Professor de Mestrado em Informática - II - PUCCAMP e Professor do Mestrado em Biblioteconomia - PUCCAMP

ATTEWEL & RULE (1984) revisaram a literatura sobre o efeito ou impacto da tecnologia nas organizações e sugerem que ambos os pontos de vista são justificados. A literatura mostra que tecnologia pode:

(1) centralizar e descentralizar a autoridade dentro da organização;

(2) aumentar e diminuir as oportunidades para participação e envolvimento dos funcionários;

(3) permitir aos trabalhadores maior acesso à informação administrativa ou reforçar o controle administrativo sobre o fluxo de informação;

(4) mudar ou congelar a estrutura organizacional;

(5) limitar ou aumentar a satisfação no trabalho.

Em resumo, a tecnologia pode humanizar ou desumanizar o ambiente de trabalho - o fator determinante é a filosofia gerencial adotada levando em consideração as implicações de um ambiente em mudança.

O gerente precisa manter um estado de espírito próprio para se adaptar às inovações e mudanças, pois elas são uma realidade. DRUCKER (1980:89) já afirmava que "a única certeza sobre o futuro... é que ele será turbulento. Em época de turbulência, a primeira tarefa do administrador é tomar medidas para garantir a capacidade de sobrevivência da instituição... adaptar a mudanças bruscas e se beneficiar das novas oportunidade."

ADAMS (1986:102) corrobora com este pensamento, em trabalho mais recente, argumentando que: "Se o futuro trouxer uma constante, esta será a mudança... O ritmo com que a mudança ocorrerá será acelerado, com mudança sendo sentida em nenhum outro lugar mais fortemente do que no processo envolvendo o manuseio de informação... "os sistemas de informação precisam desenvolver novos papéis para poder satisfazer as necessidades da organização e dos clientes.

As conseqüências da informatização para o trabalho em sistemas de informação e seus funcionários são óbvias. A tecnologia tem afetado não só o conteúdo de distribuição dos serviços, mas tem também criado tipos totalmente novos de trabalho, forçado a redefinição de algumas funções, influenciado relacionamentos interpessoais, repercutindo seriamente na estrutura organizacional.

A introdução de tecnologia informacional tem forçado os gerentes a confrontarem várias questões. A solução destas sugere que um novo tipo de administrador/gerente está surgindo neste meio. CARTEE JR. (1990) aponta várias questões a que este gerente terá que responder:

(1) Como que as decisões gerenciais com relação à implantação da automação podem ser tomadas no sentido de diminuir custos e maximizar benefícios?;

(2) Como a gerência deve responder às demandas crescentes com respeito à melhoria de qualidade de vida no trabalho vis a vis às preocupações tradicionais com respeito à produtividade?;

(3) Como que os gerentes podem melhor compreender, moldar e controlar as mudanças associadas à automação e não simplesmente encarar suas conseqüências como algo inevitável?

Para responder a estas questões, o gerente precisa estar imbuído de um estado de espírito adequado para se adaptar às inovações tecnológicas e mudanças causadas por estas. Para isso, a estrutura de sua organização precisa ser aberta. De acordo com WEBB e JENSEN (1989), uma estrutura aberta enfatiza;

- maior desejo e estímulo à cooperação;

- aumento do fluxo de informação;

- maior flexibilidade nas tomadas de decisões;

- recompensa e estímulo a novas idéias;

- um aumento da capacidade de ver além do óbvio;

- uma preocupação com o indivíduo e suas necessidades;

- desenvolvimento de uma atmosfera de escolha e liberdade com poucos regulamentos e políticas.

O gerente precisa ter certas qualidades específicas para atuar nesta estrutura. Além do mais, mudanças sociais, avanços tecnológicos e outras mudanças do ambiente externo e interno afetam diretamente a estrutura de sistemas de informação.

É evidente que o estilo gerencial tradicional não é apropriado para levar essas organizações até o século XXI. (RIGGS, 1982) A literatura sugere algumas qualidades que esses gerentes precisam ter para agir nesse contexto de desenvolvimento tecnológico:

(1) Demonstrar flexibilidade em seu estilo e explorar estruturas organizacionais participativas. Estimular um ambiente de colaboração e inovação. Trabalhar embasado numa abordagem que tem como enfoque a contribuição de cada indivíduo. (WEBSTER, 1987);

(2) Disposição para tomar riscos calculados. Deve acessar a atitude da administração superior, para verificar se esta apóia as mudanças propostas. Examinar as conseqüências de um eventual fracasso e determinar se a organização possui os recursos e está disposta a investir. (CARGILL & WEBB, 1988);

(3) Planejamento estratégico para lidar com as ameaças e oportunidades do meio ambiente. Consolidar recursos nas áreas que afetarão significativamente o papel futuro da unidade,

seu desempenho e suas capacidades. (LOWRY, 1988);

(4) Mostrar a seu corpo de funcionários para onde a organização está indo, avaliar os caminhos alternativos para se chegar lá, implementar planos que contribuirão para se alcançar as metas propostas, e analisar o grau de sucesso que foi alcançado. (McCLURE, 1980).

A estes, podemos acrescentar;

(5) Agir com respeito e confiança. Assumir um compromisso com o bem-estar de cada participante do processo. Reconhecer que o pessoal é o recurso mais valioso da instituição. Por seu exemplo, o gerente deve estimulá-lo a alcançar níveis cada vez mais altos de realização;

(6) Articular e manter uma visão de futuro que possa ser assimilada pelo pessoal, capacitando-o a relacionar sua responsabilidade com o atingimento das metas da organização. Inspirar confiança no futuro e estimular compromisso e lealdade à missão da instituição;

(7) Manter uma força de trabalho de especialistas treinados e capazes de responder às rápidas mudanças provocadas pelo desenvolvimento tecnológico.

DACSHINAMURTI (1985) pesquisou a opinião mais de 100 profissionais norte-americanos quanto às qualidades gerenciais desejáveis para uma implementação suave de novas tecnologias neste contexto de trabalho. As respostas incluíram: administração participativa; paciência para permitir aprendizagem; aceitar sugestões; permitir envolvimento e delegar responsabilidades; comunicação com o pessoal; conhecimento de todo o processo de automação; ter expectativas realísticas; possuir uma atitude positiva e entusiástica; e ser sensível às necessidades pessoais dos funcionários.

Nota-se que a maioria destes fatores ou qualidades diz respeito aos aspectos de relacionamento entre gerentes e subordinados.

IMPACTO NO PESSOAL

Ao administrar uma organização no contexto tecnológico, o gerente precisa, portanto, estar atento aos impactos da tecnologia no comportamento e atitudes do pessoal profissional. JOGODZINSKI (1985) alerta que a automação possui um impacto em vários aspectos centrais à individualidade dos funcionários, incluindo o "status profissional", segurança no trabalho, e auto-estima. GANUS (1985), em publicação do mesmo ano, também adverte que não existe apenas o temor de que com a introdução de computadores o ambiente será desumanizado, mas também um temor de que seu próprio espaço territorial de trabalho será modificado para algo desconfortável.

Embora OLSGAARD (1989) afirme que se reconhece que a interface homem-máquina cause alguns efeitos negativos, não tem havido ainda, um volume de pesquisa que tenha fornecido dados suficientes a ser analisados, i.e., a necessidade de se saber melhor sobre o stress que a nova tecnologia provoca; aspectos ambientais e ergonométricos que contribuem para a eficácia; tipo de treinamento necessário para uma implementação adequada; questões relativas à aceitação, e outras enfocando a interação homem-máquina.

AMBIENTE CRIATIVO

As organizações que estão inseridas no contexto tecnológico requerem habilidades de pensamento criativo dentre seu pessoal para lidar com este ambiente inovador, em mudança, de constante conflito, impregnado de limitações e oportunidades causadas por esta interação. O pessoal precisa aprender a desempenhar funções rotineiras de formas diferentes. A estrutura organizacional pode estimular ou abafar o pensamento criativo e iniciativa. O gerente é a pessoa-chave responsável por esta estrutura onde a criatividade e inovação devem ser estimuladas.

CURTISS (1987) identificou alguns traços que promovem a criatividade e inovação nas pessoas. Ter consciência destas características pode ajudar o gerente a desenvolver e estimular criatividade entre seus subordinados, principalmente entre os profissionais:

Curiosidade - Mente aberta, interesse pelo desconhecido;

Flexibilidade - Espontaneidade, liberdade para mudar;

Imaginação - Visualização, fantasia;

Apreciação - Pelo complexo e ambíguo;

Perspectiva - Habilidade de ver por diferentes prismas.

Persistência - Muito trabalho, mesmo sem certeza dos resultados;

Coragem - Para ser você mesmo, reconhecer a validade e importância do que você faz-arriscar;

Desafio - Desafio de questionar, analisar e criticar;

Paixão - Confiança de que as frustrações levarão ao crescimento, força e descobertas significativas;

Celebração - Tomar tempo para saborear as realizações e vitórias.

O gerente que lida com tecnologia no contexto informacional precisa possuir e promover estas características para desenvolver na organização um clima de aceitação e inovação. Precisa criar um ambiente onde os indivíduos participem e cheguem às suas próprias conclusões com confiança.

Existem alguns conceitos administrativos que estimulam um ambiente criativo. NAYAK & KETTERINGHAM (1986) no bestseller "Breakthroughs" apontam vários:

1. não desestimular a criatividade e inovação sempre que alguém estiver desenvolvendo sua tarefa;

2. fracassos e erros ajudam na aquisição e maiores informações sobre a tarefa e sobre você mesmo;

3. os "experts" irão com freqüência rejeitar uma boa idéia devido à sua incapacidade de ver além das metas e necessidades do sistema;

4. sempre demonstrar respeito aos subordinados;

5. quando as pessoas normais são apresentadas com um desafio, elas se elevam ao nível deste;

6. a pessoa certa é o recurso mais importante para desenvolver qualquer tarefa;

7. quando os outros sabem que você depende deles, eles assumem o compromisso de desempenhar a contento;

8. tenacidade e energia neutralizam a resistência e ceticismo provindos do sistema.

CONCLUSÃO

De acordo com FRICK (1985:265), "Os mais de 100 estudos realizados nos últimos anos indicam que o que as pessoas mais querem é se tornar mestres de seu ambiente imediato e sentir que seu trabalho e si próprios são importantes. "A meta dos gerentes de informação deve ser maximizar os recursos para eficazmente servir os clientes.

Em 1989 MARCHANT & ENGLAND (1989) afirmaram que uma revolução de tecnologia informacional estava ocorrendo e que iria abalar a eficácia dos métodos administrativos tradicionais, científicos e autoritários. Esta revolução provocaria mudanças profundas nas expectativas dos clientes e nas funções dos gerentes.

Essa revolução contínua, e o futuro depende em grande parte da atuação dos gerentes da informação, do papel que desejam desempenhar e do estilo gerencial que irão adotar. Eles devem fazer proveito desse ambiente de iniciativa e liderar essas mudanças e não permitir que sejam arrastados por elas, pois a tecnologia deve ser serva e não mestra da conduta gerencial.

O gerente que deseja obter sucesso na organização, que é afetada pelos desenvolvimentos tecnológicos, precisa adaptar seu estilo gerencial às exigências de um ambiente em constantes mudanças, utilizando-se de inovação e criatividade bem como possuir um elevado grau de relacionamento com todos aqueles envolvidos neste e por este ambiente em constante transformação.

BIBLIOGRAFIA

(1) ADAMS, R. J. **Information technology & libraries**: a future for academic libraries. London: Croom, 1986.

(2) ATTWELL, P. & RULE, J. Computing and organizations: what we know & what we don't know. **Communications of the Association for Computing Machinery**. v. 27, p. 1184-1192, dec., 1984.

(3) BURTON, P. F. Information technology and organizational structure. **ASLIB Proceedings**, v. 40, p. 57-68, mar., 1988.

(4) CARGILL, J. & WEBB, G. M. **Managing libraries in transition**. Phoenix: Orys Press, 1988, p. 68-69.

(5) CARTEE JR., L. D. Is library automation producing a new kind of manager? **Journal of Library Administration**, v. 13, n. 1/2. p. 99-115, 1990.

(6) CURTIS, D. **Introduction to visual literacy**. Englewood Cliffs: Prentice Hall, 1987. p. 221-224.

(7) DAKSHINAMURTI, G. Automation's effect on library personnel. **Canadian Library Journal**, v. 42, n. 6, p. 343-351, dec., 1985.

(8) DRUCKER, P. **Managing in turbulent times**. New York: Harper, 1980.

(9) FRICK, E. Humanizing technolgy through instruction. **Canadian Library Journal**, v. 41, n. 5, p. 265, oct., 1984.

(10) GANUS, S. S. Office computers: managing the human impact. **Journal of Information & Image Management**, v. 18, n. 2/3, p. 26-31, 1985.

(11) JAGODZINSKI, A. P. The interaction between electronic storage systems and their users. In: N. Fjallbrant, (ed.) **The future of information resources for science & technology and the role of libraries**. (Proceedings of the lith meeting of IATUL). Goteburg: Chalmers University of Technology, 1985. p. 133-137.

(12) LOWRY, C. Convergence of technologies: how will libraries adapt? **Library Administration & Management**, v. 2, p. 82, mar., 1988.

(13) MARCHANT, M. P. & ENGLAND, M. M. Changing management techniques as libraries automate. **Library Trends**, v. 37, n. 4, p. 469-483, spring, 1989.

(14) McCLURE, C. R. Library managers: can they manage? Will they lead? **Library Journal**, v. 105, p. 2389, nov. 15., 1980.

(15) NAYAK, P. R. & KETTERINGHAM, J. M. **Breakthroughs**. New York: Dawson, 1986.

(16) OLSGAARD, J. N. The physiological & managerial impact of automation on libraries. **Library Trends**, v. 37, n. 4, p. 484-494, spring, 1989.

(17) RIGGS, D. E. Transformational leadership and the electronic academic library **Conference on Integrated Online Library Systems**, Sept. 23-24, 1986. St. Louis, Proceedings. ed. David C. Genaway. Canfield: Genaway & Associates, 1987. p. 366.

(18) WEBB, T. D. & JENSEN, E. A. Managing innovative information technology. **Journal of Library Administration**, v. 10, n. 2/3. p. 131-142, 1989.

(19) WEBSTER, D. E. Impact of library technology on management. **Conference on Integrated Online Library Systems**, Sept. 23-24, 1986. St. Louis, Proceedings, ed. David C. Genaway. Canfield: Genaway & Associates, 1987. p. 175.

SISTEMAS GERENCIADORES DE BANCOS DE DADOS ORIENTADOS A OBJETOS PARA AMBIENTES DE DESENVOLVIMENTO DE SOFTWARE

OBJECT ORIENTED DATA BASE MANEGEMENT SYSTEMS FOR THE SUPPORT OF SOFTWARE DEVELOPMENT ENVIRONMENTS

Carlos Miguel Tobar TOLEDO*

ABSTRACT

Data Base Management Systems (DBMSs) for the support of Software Development Environments (SDEs) constitute a new generation of DBMSs; some of those DBMSs, called Object Oriented Data Base Management Systems (OODBMSs), present characteristics of the object oriented approach and are considered to be an answer for the support of SDEs. This paper presents a review of concepts, characteristics, and established requirements for an SDE in order to characterisize the functionalities of an OODBMS for the support of SDEs.

KEY-WORDS: Data Base Management Systems, DBMS, Object Oriented Data Base Management Systems, OODBMS, Software Development Environments, SDE, Software Engineering, SE

RESUMO

Sistemas Gerenciadores de Bancos de Dados (SGBDs) para ambientes de desenvolvimento de software (ADSs) compõem uma nova geração de SGBDs, alguns dos quais apresentam características do paradigma da orientação a objetos, os Sistemas Gerenciadores de Bancos de Dados Orientados a Objetos (SGBDOOs), que surgem como uma resposta ao suporte de ADSs. Este trabalho apresenta uma revisão de conceitos, características e requisitos estabelecidos para ADSs, com o objetivo de caracterizar as funcionalidades de um SGBDOO suporte para ADSs.

PALAVRAS-CHAVE: Sistemas Gerenciadores de Bancos e Dados, SGBDs, Sistemas Gerenciadores de Bancos de Dados Orientados a Objetos, SGBDOOs, Ambientes de Desenvolvimento de Software, ADSs, Engenharia de Software, ES

1. INTRODUÇÃO

Sistemas Gerenciadores de Bancos de Dados (SGBDs) para ambientes de desenvolvimento de software (ADSs) compõem uma nova geração de SGBDs, destinada a suportar o que se convencionou chamar novas aplicações. Como exemplo das novas aplicações, além daquelas abrangidas pela engenharia de software, podem ser citadas:

- CAD/CAM, projeto auxiliado por computador,
- AI, bases de conhecimento/sistemas especialistas,
- GIS, sistemas de informação cartográficos/geográficos,
- multimidia e hipertexto,
- redes de telefonia inteligentes e
- sistemas de entretenimento.

Para atender às novas aplicações, o papel do SGBD deixa de ser de um repositório central contendo dados estáticos, para transformar-se em um sistema distribuído e inteligente para armazenamento de objetos ativos.

O suporte aos ADSs continua sendo um desafio para os pesquisadores da área, apesar de já existirem dezenas de SGBDs comercializados e centenas de pro-

(*) Mestre em Ciência da Computação pelo IMECC-UNICAMP, atualmente trabalha como pesquisador junto à Fundação Centro Tecnológico para Informática e como professor junto ao Instituto de Informática da PUCCAMP.

jetos e protótipos ditos orientados às novas aplicações.

Considera-se que um SGBD para engenharia de software, além de ter que suportar as atividades de desenvolvimento de software e respectiva gerência, deve considerar uma miríade de ferramentas e necessidades advindas dos diversos domínios da aplicação que possam existir [Zicari92]. Precisa, também, ser adptável, flexível e dinâmico na mesma proporção do empreendimento, projetos e pessoas que deve suportar [Chen92].

O restante deste trabalho está dividido como se segue. A seção 2 apresenta as necessidades e restrições que um ADS pode impor a um SGBD. A seção 3 define o que vem a ser um SGBD Orientado a Objetos (SGBDOO). Na seção 4 são tecidas considerações com relação às características desejáveis em um SGBDOO para suporte à engenharia de software e, finalmente, são tecidas conclusões.

2. ADS COM SGBD

O termo ADS se refere a um conjunto de ferramentas de hardware e software necessário ao desenvolvedor de sistemas de software (aplicações).

A complexidade advinda da aplicação repercute na quantidade de tempo e de recursos (hardware, software e pessoal) necessária ao seu desenvolvimento, além, é claro, nos requisitos de gerência impostos por eles.

A integração dos recursos de hardware e de software a ferramentas que auxiliem o desenvolvimento e a sua gerência é fator fundamental para a obtenção de qualidade nos resultados e diminuição dos custos a eles associados.

Existem cinco tipos de integração de ferramentas: dados (das ferramentas e seus inter-relacionamentos), controle (notificação de eventos entre ferramentas), apresentação (interface como o usuário), processos (controle das atividades de desenvolvimento e seus encadeamentos) e plataforma (inter-operabilidade das ferramentas) [Clow89] [Wasserman89].

A combinação dos tipos de integração e mesmo a sua existência varia de ADS para ADS. Para efeitos deste trabalho, considera-se a existência da integração de dados nos ADSs.

O papel de um SGBD em um ADS é o de integrador de dados. Como tal, o SGBD deve prover serviços básicos que incluem: armazenamento e gerência de objetos, entidades, relações e ligações; controle de versões e configuração; serviço e identificação ("naming"); segurança e controle de transações [Chen92].

Podemos classificar ASDs de diversas maneiras: de acordo com a cobertura de um ciclo de vida, de acordo com os tipos de ferramentas por eles providos, de acordo com a dimensão da instalação e complexidade da aplicação, etc.

A abordagem adotada para análise dos ADSs foi a da escolha de uma perspectiva centrada na existência do SGBD e, a partir daí, explorar os aspectos decorrentes de seu estudo.

Na análise promovida em [Lockeman90], a respeito do futuro da tecnologia de SGBDs, são estabelecidas três forças que direcionam os esforços atuais de pesquisa na área:

- aplicações da tecnologia de SGBDs,
- tecnologia básica de hardware e software existente, sobre a qual a tecnologia de SGBDs se apoia ou pode vir a se apoiar, e
- padrões que, por um lado, estabelecem restrições sobre os desenvolvimentos, mas, por outro, promovem a integração de produtos e tecnologias heterogêneos.

Tendo estabelecido que a aplicação da tecnologia de SGBDs é a engenharia de software, obtêm-se três dimensões para a análise dos requisitos que um ADS impõe ao seu componente integrador de dados:

a) das aplicações a serem desenvolvidas no ADS, resultante das necessidades da organização, denominada visão da organização;

b) das políticas para desenvolvimento, advindas do exercício da engenharia de software, denominada visão da engenharia de software; e

c) das tecnologias e padrões existentes no ADS e requeridos nas aplicações a serem desenvolvidas, denominada visão tecnológica.

Nota-se que as duas primeiras dimensões resultam da primeira força direcionadora de [Lockeman90]. Esta mudança de perspectiva se origina na existência de duas dimensões quando se estabelece a engenharia de software como aplicação da tecnologia de SGBDs (a própria engenharia de software e a aplicação desta).

A terceira dimensão é uma junção das duas últimas forças direcionadoras, por considerarmos que os padrões se aplicam aos componentes da tecnologia, podendo dessa maneira, serem analisados em conjunto.

2.1. A Visão da Organização

Esta dimensão representa o porquê do uso do ADS. Nela podem-se distinguir três aspectos relativos à organização onde o ADS se encontra instalado: o perfil da instalação responsável pelos desenvolvimentos, o domínio das aplicações e os tipos de soluções desenvolvidas.

Termos como "programming in small" e "programming in the large" foram cunhados para sintetizar aspectos desta dimensão.

O perfil da instalação representa todas as componentes encontradas no ambiente de desenvolvimento, principalmente com relação aos recursos humanos: tamanho da equipe, funções, relacionamentos, etc.

O domínio das aplicações representa a área de atuação da organização e, portanto, o tipo de problemas que requerem solução.

Os tipos de solução representam os modelos tecnológicos selecionados como base para a solução dos problemas da organização: tempo real, tempo compartilhado ou "batch"; centralizado ou distribuído; multi usuário ou mono usuário; etc. (este aspecto é discutido mais adiante na dimensão tecnológica, porém, nesta dimensão se aplica aos produtos do ADS e lá ao próprio ADS).

Uma classificação de ADSs considerando esta dimensão, seria a encontrada em [Perry88]:

- modelo individual - também conhecido como ambiente de programação, representa os ambientes que provêm o conjunto mínimo de ferramentas de implementação necessárias para a construção de software;
- modelo familiar - representa os ambientes que, além de oferecerem o que o modelo individual supre, oferece facilidades para o suporte da interação de um pequeno grupo de programadores. Basicamente, estende o modelo anterior com uma combinação de gerência de configuração, integração, distribuição (lan) ou gerência de projeto;
- modelo metropolitano - representa os ambientes que oferecem o mesmo que o modelo familiar e se destinam a um grupo de mais de 20 pessoas, orientado por uma metodologia (métodos e técnicas de desenvolvimento e gerência);
- modelo estadual - que apresenta um modelo genérico para desenvolvimento de software, o qual é instanciado para cada projeto e que permite a gerência das diferenças entre as várias instâncias.

2.2. A visão da engenharia de software

Esta dimensão representa o que o ADS oferece, em termos de políticas e estratégias, para facilitar o desenvolvimento. Nela podem-se distinguir três aspectos existentes no desenvolvimento e que, de alguma maneira, receberam atenção da engenharia de software. Esta perspectiva é encontrada em [Pocock91]. São eles:

- o processo, onde se concentram as atividades de planejamento, definição e gerência do desenvolvimento e dos recursos necessários; as ferramentas de suporte a estas atividades são as de gerência de projeto;
- o produto, onde se concentram as atividades de geração e validação de produtos e sub produtos; as ferramentas de suporte são as denominadas CASE ou, mais precisamente, "lower case" (codificação e teste) ou "upper case" (especificação e projeto);
- a construção, onde se concentram as atividades de coordenação dos produtos e sub produtos gerados em paralelo por atividades inter dependentes; as ferramentas são as denominadas gerência de configuração ou controle de mudanças e integração.

Uma classificação de ADSs considerando esta dimensão seria a encontrada em [Wasserman89]:

- ferramentas horizontais, usadas durante todo o processo de desenvolvimento de software,
- ferramentas verticais, usadas para a especificação de uma determinada fase do processo.

[Gibson91] estende a classificação das ferramentas verticais em:

- "upper", utilizadas para a definição da perspectiva estratégica da organização (missão, objetivos, planos estratégicos, planos operacionais, recursos, etc.);
- "middle", utilizadas para o planejamento (análise e projeto; definição do nível lógico do modelo estratégico);
- "lower", utilizado para a construção (nível físico do modelo estratégico).

2.3. A visão tecnológica

Esta dimensão representa o como a ADS suporta o ferramental oferecido aos seus usuários. Os aspectos relevantes nesta dimensão dizem respeito ao conjunto de soluções utilizadas para a implementação e funcionamento do próprio ADS, são elas:

- a arquitetura, diz respeito à tecnologia usada para tratar os problemas de capacidade de processamento, capacidade de armazenamento e tempo de resposta; basicamente, indica se a solução é centralizada ou distribuída, podendo apresentar uma combinação das duas;
- a plataforma, diz respeito à tecnologia existente nos produtos escolhidos para comporem o hardware e o software básico; aquí se encontra uma gama enorme de fornecedores e opções; por exemplo, para o hardware, "main frames", estações de trabalho, etc., para o software, basicamente, o sistema operacional (pode-se pensar nos componentes de comunicação, armazenamento primário e secundário, interface, etc.);
- o modelo computacional, diz respeito às tecnologias escolhidas para comporem as facilidades oferecidas, i.e., a variedade de suporte às possíveis soluções ou necessidades dos usuários; basicamente incorporam as características funcionais do ambiente (concorrência, multi-usuário, multi-linguagens, paradigmas de programação, etc.);
- filosofia de integração, diz respeito à preocupação para que os componentes do ambiente (presentes e futuros) sejam portáveis, possam interoperar entre si e interajam com o usuário de maneira similar, basicamente indica o grau de "abertura" do ambiente.

Estes aspectos em conjunto indicam a existência ou não de: facilidades para a inclusão de novas tecnologias, transparência de localização, trabalho em grupo, portabilidade, evolução, etc.

Uma classificação de ADSs considerando esta dimensão seria a encontrada em [Dart87] e estendida por [Strelich89]:

- centrados em linguagens (Interlisp, Smalltalk ou Cedar);
- orientados por estrutura (Gandalf ou Mentor);
- "tool-kits" (sistemas Unix);
- baseados em métodos (Excelerator ou SREM); e
- "environment framework" (Arcádia ou ISM).

3. SGBDOO

O paradigma de programação orientada a objetos, incorporado ao mundo dos SGBDs, resultou no aparecimento de bancos de dados não convencionais, denominados SGBDs Orientados a Objetos (SGBDOOs).

O termo "orientado a objetos" é melhor interpretado quando se refere a qualquer abordagem que explore encapsulamento, no processo de projeto e desenvolvimento de software [Nierstrasz88].

O termo SGBDOO é utilizado da mesma maneira que [Cattell91], para quem as abordagens mais populares para SGBDs de próxima geração são chamadas de abordagens para SGBDOOs ou SGBDs relacionais estendidos.

Existem, basicamente, duas escolas de pesquisa trabalhando com SGBDOO: a primeira advoga a abordagem de que um SGBDOO é o resultado da tecnologia dos SGBDs relacionais e estendidos semanticamente com os aspectos da orientação a objetos [CADBMSF90]; a segunda advoga os princípios existentes nas linguagens orientadas a objetos como base para o desenvolvimento de uma nova tecnologia de SGBDs [Atkinson89].

Exemplos de SGBDOOs da primeira escola são Postgres [Stonebraker91] e Starburst [Lohman91], enquanto que, da segunda escola são Object Store [Lamb91], Gemstone [Butterworth91] e O [Deux91].

As características essenciais para um SGBDOO defendidas por cada escola são apresentadas nas tabelas 1 e 2. Na tabela 3 as características de cada escola são apresentadas lado a lado. Note-se que, além da coincidência no número (treze), algumas características são iguais ou podem ser vistas como uma composição das defendidas pela outra escola. É o caso da característica número 1 em [CADBMSF90] que é uma combinação das características 1, 4 e 8 em [Atkinson89].

Tabela 1 - Características de um SGBDOO em [Atkinson89]

1. Complex Objects
   A capacidade de permitir definição de objetos (mais complexos) a partir de objetos já definidos (mais simples), através de operadores estruturais utilizados arbitrariamente. Além disso, a capacidade de gerenciar as hierarquias de composição resultantes.
2. Object Identity
   O mecanismo de se poderem distinguir dois objetos, independentemente dos valores de seus atributos.
3. Encapsulation
   A separação clara entre a semântica visível e a implementação dos objetos.
4. Types and classes
   A capacidade de organização de objetos similares e suas implementações em uma abstração.
5. Inheritance
   A capacidade de criação de novas classes a partir das existentes.
6. Polymorphism and Dynamic Binding
   A capacidade de associação de mensagens similares a diferentes métodos, dependendo do tipo do objeto passado como parâmetro; os métodos podem ser associados a tempo de execução (dynamic) ou de compilação.
7. Computation completness
   A capacidade de permitir ao usuário expressar qualquer tipo de função computável.
8. Extensibility
   A capacidade de permitir a extensão do conjunto de tipos de um SGBD; não permitindo a ocorrência de distinção de uso entre os tipos do sistema e os definidos pelo usuário.
9. Persistence
   A capacidade de permitir a existência de objetos além do tempo de vida dos processos que os criaram.
10. 2ndary Storage Management
    Mecanismos de acesso às informações que sejam eficientes; por exemplo "clustering", índices, "buffers" e otimizações de busca.
11. Concurrency
    A capacidade de permitir o acesso concorrente às informações através de atomicidade, compartilhamento controlado por "locks" e "serializability".
12. Recovery
    A capacidade de permitir a recuperação das informações do SGBD a um estado anterior íntegro, em caso de falha de software ou hardware.
13. Ad hoc query facility
    Mecanismos que permitam o acesso eficiente aos objetos, através de declarações de alto nível, além de navegação programável.

Tabela 2 - Características de um SGBDOO em [CADBMSF90]

1. Rich Type System
   Existência de um sistema de tipos semanticamente rico e extensível.
2. Inheritance
   Capacidade de herança entre tipos.
3. Encapsulation
   Capacidade de isolar, através de encapsulamento, a chamada das funções (incluindo os procedimentos do SGBD) de suas implementações e das estruturas de dados.
4. Unique Identifiers
   Capacidade de associação de identificadores únicos aos registros em caso de não existir uma chave primária.
5. Rules (triggers, constraints)
   Capacidade de associação de regras (restrições ou gatilhos) a uma função ou conjunto de informações específico (coleção).
6. Non-procedural HL Access Lang.
   Mecanismo programável (único) de acesso às informações do SGBD, cuja "interface" é uma linguagem de acesso de alto nível não "procedural".
7. 2Ways to specify collections
   Existência de ao menos duas formas para a especificação de coleções: uma usando a enumeração dos membros e a outra usando a linguagem de recuperação para a especificação de pertinência.
8. Updatable Views
   Mecanismos atualizáveis que permitam diferentes visões das informações no SGBD.
9. No performance indicators
   Indicadores de "performance" pouco têm a ver com modelos de dados e não devem aparecer neles.
10. Multiple HLLs
    Existência de múltiplas linguagens de alto nível para o acesso às informações do SGBD.
11. Persistence
    Capacidade de permitir que um entre vários objetos permaneça armazenado após o término do processo que o gerou. Essa capacidade deve ser suportada através de extensões em compiladores e sistemas de execução (mais ou menos) complexos.
12. SQL
    Suporte à linguagem SQL, visto que a mesma se tornou um padrão.
13. Queries for communication
    Mecanismo onde consultas e suas respostas resultem no nível mais baixo de comunicação entre clientes e servidor.

Tabela 3 - Comparação de Características para SGBDOOs

[Atkinson89]	[CADBMSF90]
01. Complex Objects	01. Rich Type System
02. Object Identity	02. Inheritance
03. Encapsulation	03. Encapsulation
04. Types and classes	04. Unique Identifiers
05. Inheritance	05. Rules (triggers, constraints)
06. Dynamic Binding and Polymorphism	06. Non-procedural HL Access Lang.
07. Computation completness	07. 2 ways to specify collections
08. Extensibility	08. Updatable Views
09. Persistence	09. No performance indicators
10. 2ndary Storage Management	10. Multiple HLLs
11. Concurrency	11. Persistence
12. Recovery	12. SQL
13. Ad hoc query facility	13. Queries for communication

Pode-se verificar que existe concordância entre as escolas quanto a determinado subconjunto de características que um SGBDOO deve oferecer:

a. sistema de tipos e classes extensível que permita a definição de objetos complexos;

b. identificação de objetos;

c. encapsulamento e a existência de polimorfismo;

d. hereditariedade e a existência de "overriding"; e

e. persistência.

Dessas apenas a identificação de objetos e a persistência não são encontradas, normalmente, nas linguagens orientadas a objetos.

Estas cinco características correspondem a oito das características essenciais em [Atkinson89] e a cinco em [CADBMSF90].

De qualquer forma, pode-se definir um SGBDOO como o resultado da conjunção de idéias provenientes das linguagens de programação orientadas a objetos e dos SGBDs convencionais. Em [Schlageter89] encontramos a seguinte definição para SGBDOO: um SGBDOO deve ser um SGBD, deve ser um sistema orientado a objetos e, além disso, suportar as necessidades advindas das novas aplicações.

4. SGBDOO PARA ENGENHARIA DE SOFTWARE

No restante deste trabalho assume-se a existência de um SGBDOO como integrador de dados nos ADSs, isto porque os SGBDOOs são considerados ideais para o suporte de ADSs [Zicari92] [Cattell91] [Blair90] [Atkinson89] [Peterson87], embora ainda não exista uma concordância quanto às características e funcionalidades necessárias para atendimento dessa nova aplicação.

[Zicari92] propõe dezessete características essenciais em um SGBD para suporte de ADSs. Estas são apresentadas na Tabela 4, relacionadas aos seguintes macro aspectos:

- suporte multi-usuário e perfis de usuários,
- tipo de informação e sua transformação/evolução,
- armazenamento distribuído, íntegro, eficiente e confiável,
- incorporação de tecnologia proveniente de outras áreas.

O aspecto suporte multi-usuário e perfis de usuários compreende características advindas de um ambiente compartilhado por diversos usuários com responsabilidades e interesses distintos. São elas: compartilhamento de informações, concorrência, trabalho cooperativo e gerência de transações não convencionais, além de visões e mecanismos de segurança.

O aspecto tipo de informação e sua transformação/evolução compreende características relacionadas com os diferentes objetos criados e manipulados e seus inter-relacionamentos. São elas: complexidade dos objetos, atualização da estrutura da informação, relacionamento (esquema) e constituição dos objetos, extensibilidade para novos tipos de objetos e manipulação de meta-informação.

O aspecto armazenamento distribuído, íntegro, eficiente e confiável compreende características advindas das facilidades encontradas em SGBDs convencionais, quais sejam: persistência, restrições de integridade, facilidades para consultas (queries) específicas, recuperação de falhas, distribuição em ambientes (hardware e software) não homogêneos e gerência eficiente do armazenamento secundário.

O aspecto incorporação de tecnologia proveniente de outras áreas compreende características existentes em outras áreas de pesquisa e que trazem benefícios. São elas: triggers (gatilhos, responsáveis por permitir a ativação de ações a partir de modificações no estado do SGBD), capacidade dedutiva (encontrada em sistemas especialistas), além de mecanismos gráficos avançados.

Além desses macro-aspectos, existe a necessidade de facilidades para a gerência de configuração, representada pelo tratamento de versões.

A figura 4 apresenta uma síntese da comparação da Tabela 1 [Atkinson89] e da Tabela 2 [CADBMSF90] com as características estabelecidas em [Zicari92].

Nota-se que a abordagem orientada a objetos baseada em linguagens orientadas a objetos, essencialmente, atende a apenas sete dos dezessete requisitos para um SGBD suporte de ADSs. Outros quatro requisitos, considerados opcionais em [Atkinson89], são encontrados em [Zicari92] como essenciais.

A abordagem orientada a objetos baseada na extensão do modelo relacional, essencialmente, atende a apenas cinco dos dezessete requisitos.

Apenas três requisitos recebem concordância de ambas as escolas.

Cada um dos aspectos encontrados nas três dimensões, introduzidas na seção 2, requer algum tipo de suporte do SGBDOO. Por exemplo, o processo na visão da engenharia de software necessita que seja possível: modelar o ciclo de vida do desenvolvimento, representar a estrutura de divisão de trabalho ("work breakdown structure"), monitorar eventos, registrar a história do projeto, representar o estado de cada recurso, de cada atividade e de cada produto ou sub produto, etc. Para isso [Liu88] considera necessário dispor de uma notação temporal, modificação dinâmica de meta-informação e hereditariedade múltipla.

Ao mesmo tempo, a figura 4 apresenta uma distribuição das características em [Zicari92], considerando as três dimensões. Essa distribuição considerou que determinada característica atendia mais fortemente um determinado aspecto, podendo, no entanto, atender outros aspectos das três dimensões.

Tabela 4 - Características essenciais a SGBDs para ADSs em [Zicari92]

	suporte multi-usuário	transf. evolução inform.	tecnol. SGBDs convenc.	novas tecnol.	
1. Complex information modeling		+			(1) (2)
2. Versioning					(3)
3. Integrity constraints and triggers			+	+	(2)
4. Advanced transaction management	+				(3)
5. Schema and object updates		+			
6. Views and authorization mechanisms	+				(2)+
7. Deductive capabilities				+	
8. Advanced graphical facilities				+	
9. Meta-data handling		+			(3)+
10. Data sharing	+				
11. Secondary storage management			+		(1)
12. Ad hoc query facilities			+		(1)
13. Distribution and cooperative work	+		+		(3)
14. Extensibility		+			(1) (2)
15. Persistence			+		(1) (2)
16. Concurrency	+				(1)
17. Recovery			+		(1)

(1) essencial em [Atkinson89]
(2) essencial em [CADBMSF90]
(3) opcional em [Atkinson89]
+ parcialmente

Nota-se uma preocupação maior em considerar os aspectos tecnológicos e da engenharia de software em detrimento dos organizacionais.

Além das 17 características definidas por [Zicari92], considera-se que as seguintes cinco primeiras características essenciais e seis últimas opcionais deveriam ser consideradas:

- facilidades de recuperação semântica, para suporte dos aspectos domínio da aplicação, o produto, o processo e tipos de solução;
- multi-linguagem (completude computacional), para suporte ao aspecto o produto;
- interface com outros SGBDs, para suporte ao aspecto filosofia de integração;
- disponibilidade, para suporte ao aspecto arquitetura;
- gerência de processamento, para suporte ao aspecto plataforma;
- "traceability', para suporte ao aspecto a construção;
- "branching squeme", para suporte ao aspecto a construção;
- multi-mídia, para suporte ao aspecto modelo computacional;
- adaptabilidade (preservação do hábito, níveis dos usuários), para suporte dos aspectos filosofia de integração e perfil da instalação;
- "customizability", para suporte ao aspecto perfil da instalação;
- facilidades de representação temporal (antes, depois, durante etc.), para suporte dos aspectos domínio da aplicação e o processo.

6. CONCLUSÕES

Neste trabalho, inicialmente, caracteriza-se SGBD suporte de ADS e SGBDOO. Em seguida são tecidas considerações com relação à caracterização de um SGBDOO suporte de ADS.

Existe uma clara necessidade de se estabelecerem características básicas que um SGBDOO deve ter para atender a engenharia de software.

Essa necessidade decorre de dois fatos: primeiro, não existe consenso dos requisitos mínimos para SGBDOOs que atendam outras aplicações, incluindo o suporte a ADSs; segundo, apesar de alguns esforços para padronização, os interesses que estão por trás da comunidade que pesquisa SGBDOOs permitirão que apenas alguns tópicos possam ser padronizados.

Exemplos de esforços para padronização: X3/SPARC/DBSSG/OOBTG, o Object Management Group [Joseph91], o ANSI Resource Dictionary System, o trabalho do IEEE em interconexão de ferramentas (P1175) e o CASE Data Interchange Format da Eletronic Industries Association.

Considerando a caracterização de [Zicari92] como a correta para SGBDOOs, suporte para engenharia de software, e de sua inadequada compatibilidade com as diferentes propostas para SGBDOOs genéricos (ver tabela 4), verificamos que a conceituação de SGBDOOs poderia ser revista.

A revisão poderia dar origem a sistemas especializados e específicos para atender cada nova aplicação, cujas estruturas multi-níveis apresentassem funcionalidades comuns, similares à estrutura do sistema operacional UNIX.

A estrutura multi-níveis associada ao conceito de encapsulamento e especialização, presentes no paradigma de objetos, permitiriam, por exemplo, que um servidor de arquivos servisse como base para um servidor de objetos (que oferecesse a possibilidade de modelar objetos complexos, extensibilidade e persistência, por exemplo). Sobre esse servidor de objetos seriam adicionadas outras características dando origem a um SGBDOO genérico.

Utilizando o SGBDOO genérico e, talvez, outros servidores (comunicação remota, por exemplo) seria possível desenvolver sistemas de gerência de dados específicos para cada nova aplicação. Esse sistema poderia ser visto como um utilitário-aplicação, como uma interface ou como uma versão especializada do SGBDOO.

Não podemos esquecer que não estão contempladas, em qualquer das caracterizações estudadas, as necessidades inerentes ao processo de desenvolvimento de software (ciclos de vida ou metodologias, por exemplo), sua padronização e estruturação nos diversos componentes de software.

Um claro apelo mercantilista é notado nos artigos que descrevem os SGBDOOs existentes, quer porque apresentam um produto comercializado, quer porque defendem um projeto cujas características são contestadas por outros autores (é o caso de Stonebraker e seu Postgres). Este fato pode ser constatado ao compararmos artigos de diferentes datas a respeito do mesmo SGBDOO: ou se dizia que o objetivo não era atender ferramentas CASE e hoje é vendido como um SGBD de última geração, que suporta também ferramentas CASE; ou, inicialmente, se apresentava, sem meias palavras, como extensão ao modelo relacional e hoje tenta estabelecer uma visão tendenciosa do que vem a ser orientação de objetos para um SGBD.

Concluímos que há muito a ser feito, antes de podermos considerar a existência de um modelo apropriado para a engenharia de software, análogo ao existente modelo relacional para as aplicações comerciais e de gerência da informação.

BIBLIOGRAFIA

[Atkinson89] ATKINSON, M. et alli The Object-Oriented Database System Manifesto Proc. DOOD 89, Dec/89

[Blair90] BLAIR, G. S. et alli A synthesis of object-oriented and functional ideas in the design of a distributed software engineering environment Software Engineering Journal, May/90, p. 193-204

[Butterworth91] BUTTERWORTH, P., Otis, A. & Stein, J. The Gemstone Object Database Management System CACM, 34(10), Oct/91

[CADBMSF90] The Committee for Advanced DBMS Function Third-Generation Database System Manifesto SIGMOD Record, 19(3), Sept/90

[Cattell91] CATTELL, R. G. G. What are next-generation Database Systems? CACM, 34(10), Oct/91

[Chen92] CHEN, M. & Norman, R. J. A Framework for Integrated CASE IEEE Software, 9(2), Mar/92

[Dart87] DART, S. A. et alli Software Development Environments IEEE Computer, Nov/87, p. 18-28

[Deux91] DEUX, O. et alli The 02 System CACM, 34(10), Oct/91

[Gibson91] GIBSON, M. & Snyder, C. Computer Aided Software Engineering: Facilitating the Path for True Software and Knowledge Engineering International Journal of Soft. Eng'g and Knowledge Eng'g, 1(1), 1991, p. 99-114

[Joseph91] JOSEPH, J. V. et alli Object-Oriented Databases: Design and Implementation Proc. of the IEEE, 79(1), Jan/91

[Lamb91] LAMB, C. et alli The Objectstore Database System CACM, 34(10), Oct/91

[Liu88] LIU, L. & Horowitz, E. Object Database Support for a Software Project Management Environment Sigsoft Soft. Eng'g Notes. 13(5), Nov/88

[Lockeman90] LOCKEMAN, P., Kemper, A. & Moerkotte, G. Future Database Technology: Driving Forces and Directions Data Base Systems of the 90's Proc. Int. Symposium. Nov/90 Lecture Notes in Computer Science

[Lohman91] LOHMAN, G. M. et alli Extensions to Starburst: Objects, Types, Functions and Rules CACM, 34(10), Oct/91

[Nierstrasz88] NIERSTRASZ, O. M. A Survey of Object-Oriented Concepts in Object-Oriented Concepts and Databases, ed. Kim and Lochovsky, Addison-Wesley, 1988

[Perry88] PERRY, D. E. & Kaiser, G. E. Models of Software Development Environments Proc, loth Int. Conf. on Soft. Eng'g 1988, p. 60-68

[Peterson87] PETERSON, R. Object Oriented Data Base Design AI Expert, 2(3), March/87

[Pocock91] POCOCK, J. N. VSF and its Relationship to Open Systems and Standard Repositories Soft. Devel. Env. and CASE Technology Proc. European Symposium, Jul/91, p. 55-64 Lect. Notes in Computer Science

[Schilageter89] SCHLAGETER, R. Object-Oriented Database Systems: Concepts and Perspectives Data Base Systems Proc. Int. Symposium, Oct/89, p. 154-197 Lecture Notes in Computer Science

[Stonebraker91] STONEBRAKER, M. & Kemnitz, G.The Postgres Next-Gerenation Database Management System CACM, 34(10), Oct/91

[Strelich89] STRELICH, T. The Software Life Cycle Support Environment: A computer based framework for developing software systems Proc. ACM SigSoft/ Siggplan Soft. Eng'g Symposium on Practical Soft. Develp. Environ. Sig Plan Notices, 24(2), a989, p. 35-44

[Wasserman89] WASSERMAN, A. I.Tool Integration in Soft. Eng'g Environments Soft. Eng'g Environments Proc. Int. Workshop on Environments, Sept/89, p. 137-149 Lect. Notes in Computer Science

[Zicari92] ZICARI, R. & Medeiros, C. B. Databases for Software Engineering

POKE-TOOL - UMA FERRAMENTA PARA SUPORTE À APLICAÇÃO DOS CRITÉRIOS POTENCIAIS USOS PARA TESTE DE PROGRAMAS

POKE-TOOL - A TOOL TO SUPPORT POTENTIAL USES CRITERIA APPLICATION TO PROGRAM TESTING

Prof. Dr. José Carlos MALDONADO*
Marcos Lordello CHAIM**
Prof. Dr. Mario JINO***

ABSTRACT

This work describes implementation and operation of a multilanguage tool to support Potential Uses Criteria application, named POKE-TOOL (**PO**tencial Uses **CRI**teria **Tool** for Program Testing). Presently, this tool assists application of structural testing criteria to programs written in C, automating unit testing and permittinf more complex programs to be tested. POKE-TOOL also organizes generated data through a data base of testinf data.

KEY WORDS: Structural Software Testing, Automated Tool, Data Flow Analysis.

SUMÁRIO

Este trabalho descreve a implementação e a operação de uma ferramenta multilinguagem de suporte à aplicação dos critérios Potenciais Usos, denominada POKE-TOOL (**PO**tencial Uses **CRI**teria **Tool** for Program Testing). Atualmente, esta ferramenta auxilia a aplicação de critérios de teste estruturais em programas escritos na linguagem C, automatizando o teste de unidades e permitindo que programas mais complexos possam ser testados. A POKE-TOOL também organiza os dados gerados com a criação de uma base de dados da atividade de teste.

PALAVRAS-CHAVE: Teste Estrutural de Software, Ferramenta Automatizada, Análise de Fluxo de Dados.

1. INTRODUÇÃO

1.1 Motivação

À medida que os sistemas de software cresceram em tamanho e complexidade, o esforço requerido para testar esses sistemas tem crescido muito além das expectativas, implicando altos custos e produtos com baixa confiabilidade. Tem-se verificado que embora se gaste, em geral, até 50% do orçamento para desenvolvimento do software em atividades de teste, um número significativo de defeitos permanece sem ser detectado nos softwares liberados; esses defeitos normalmente têm um impacto grande na operação normal do sistema. Este alto custo das atividades de teste ensejou o desenvolvimento de métodos mais eficientes na detecção de defeitos no software e de ferramentas automáticas para auxiliar na produção de testes efetivos e na análise dos resultados dos testes.

Basicamente, os métodos utilizados para testar as unidades que compõem um software são baseados em duas técnicas: funcional e estrutural. Com respeito ao teste estrutural vários critérios para a seleção de casos de teste foram estabelecidos. Os primeiros critérios a surgirem foram baseados unicamente no fluxo de controle dos programas. Assim, têm-se critérios que requerem que todo os blocos de comandos seqüênciais de um programa sejam executados pelos menos uma vez (chamado critério todos-nós), ou todos os comandos de

(*) Professor do Departamento de Ciências da Computação e Estística Instituto de Ciências Matemáticas de São Carlos - ICMSC-USP.
(**) Pesquisador do Centro Nacional de Pesquisa Tecnológica em Informática para Agricultura Empresa Brasileira de Pesquisa Agropecuária - EMBRAPA.
(***) Professor de Engenharia de Computação e Automação Industrial Faculdade de Engenharia Elétrica - FEE - UNICAMP

transferência (critério todos-ramos), ou mesmo todos os caminhos do programa (critério todos-caminhos). Note-se, entretanto, que este último critério pode levar a um número infinito de casos de teste.

Com o intuito de tornar o teste de unidades mais eficiente, novos critérios de teste foram introduzidos (HER76, LAS83, NTA84, RAP85, MAL88a, MAL88a, MAL88b, URA88). Estes novos critérios de teste baseiam seus requisitos de teste na análise de fluxo de dados da unidade. A análise de fluxo de dados foi inicialmente utilizada para otimização de código por compiladores e, em geral, estabelece que a ocorrência de uma variável pode ser de dois tipos: definição e uso. Uma ocorrência é entendida como definição quando a variável recebe um valor através, por exemplo, de um comando de atribuição ou um comando de entrada de dados. Um uso ocorre quando a variável não está sendo definida. No contexto de teste de software, critérios de teste baseados em análise de fluxo de dados requerem que as interações que envolvem definições de variáveis de programa e subseqüentes referências a essas definições sejam testadas. Portanto, esses critérios baseiam-se nas associações entre uma definição de uma variável e os seus possíveis subseqüentes usos para a derivação de casos de teste.

Especificamente, Maldonado, Chaim e Jino (MAL88a, MAL88b), introduziram os Critérios Potenciais Usos baseados no conceito potencial uso; os Critérios Potenciais Usos são critérios de teste estrutural, baseados na análise de fluxo de dados e consistem, fundamentalmente, em variações da família de critérios apresentada por Rapps e Weyuker (RAP82, RAP85); são denominados: critérios todos-potenciais-du-caminhos, todos-potenciais-usos e todos-potenciais-usos/du. As associações são requeridas independentemente da ocorrência explícita de uma referência a uma determinada definição; se um uso pode existir - um potencial uso - a potencial associação é requerida. Uma associação é representada por (i, J, x) onde i é um nó onde existe uma definição de variável x e j é um nó onde existe um potencial uso de x. Um uso pode existir em j se for possível alcançar este nó através de pelo menos um caminho onde não ocorra redefinição de x.

O uso de uma ferramenta de software para auxílio ao teste de programas pode ser vinculado a um critério de teste de duas maneiras. A ferramenta de software pode utilizar o critério de teste como um guia para a geração de casos de teste que satisfaçam o critério; outra possibilidade é a utilização critério para a análise de cobertura de um conjunto de casos de teste, isto é, verificar se os casos de teste aplicados preencheram os requisitos de teste do critério.

Dessa maneira, pode-se concluir que as ferramentas automatizadas de suporte à aplicação dos diversos critérios de teste de programas são a chave para o aumento na produtividade e da eficiência da atividade de teste, pois permitem que programas maiores e mais complexos sejam testados e também automatizam a aplicação dos mais variados critérios, em especial, daqueles baseados em análise de fluxo de dados.

1.2 Organização do Trabalho

Este trabalho descreve a implementação e a operação de uma ferramenta de suporte à aplicação dos critérios Potenciais Usos, denominada POKE-TOOL (**PO**tential Uses **CRI**teria **Tool** for Program Testing) (MAL89)1. Essa ferramenta auxilia a aplicação de critérios de teste estruturais em programas escritos na linguagem C, automatizando o teste de unidades e permitindo que programas mais complexos possam ser testados. A POKE-TOOL também organiza os dados gerados com a criação de uma base de dados da atividade de teste. Dessa maneira, este tipo de ferramenta procura aumentar a produtividade na fase de teste e aumentar a qualidade dos softwares liberados.

Ela fornece ao usuário os caminhos necessários para satisfazer os Critérios Potenciais Usos e é capaz de verificar se o conjunto de casos de teste fornecido pelo usuário executou todos os caminhos requeridos. A POKE-TOOL é uma ferramenta flexível, isto é, não atrelada a nenhuma linguagem de programação específica, e permite que o usuário a configure para a linguagem de programação de seu interesse.

Na Seção seguinte são apresentadas a arquitetura estabelecida para a ferramenta e a implementação deste trabalho; na Seção 3 os aspectos de configuração da POKE-TOOL; na Seção 4, os aspectos operacionais da ferramenta são apresentados através de um exemplo de utilização; e, na Seção 5, as conclusões são apresentadas.

2. ARQUITETURA E IMPLEMENTAÇÃO DA POKE-TOOL

Nesta seção são apresentadas a arquitetura e a implementação da ferramenta POKE-TOOL.

Com a implementação da ferramenta POKE-TOOL pretende-se, além de auxiliar o uso prático dos critérios Potenciais Usos, viabilizar a realização de comparações entre estes critérios e os demais critérios de teste estrutural, bem como a avaliação da adequação destes critérios a classes de erros.

2.1 Arquitetura Funcional da Ferramenta POKE-TOOL

A arquitetura funcional da ferramenta POKE-TOOL, ilustrada na Figura 1, foi inicialmente proposta por Maldonado, Chaim e Jino (MAL89).

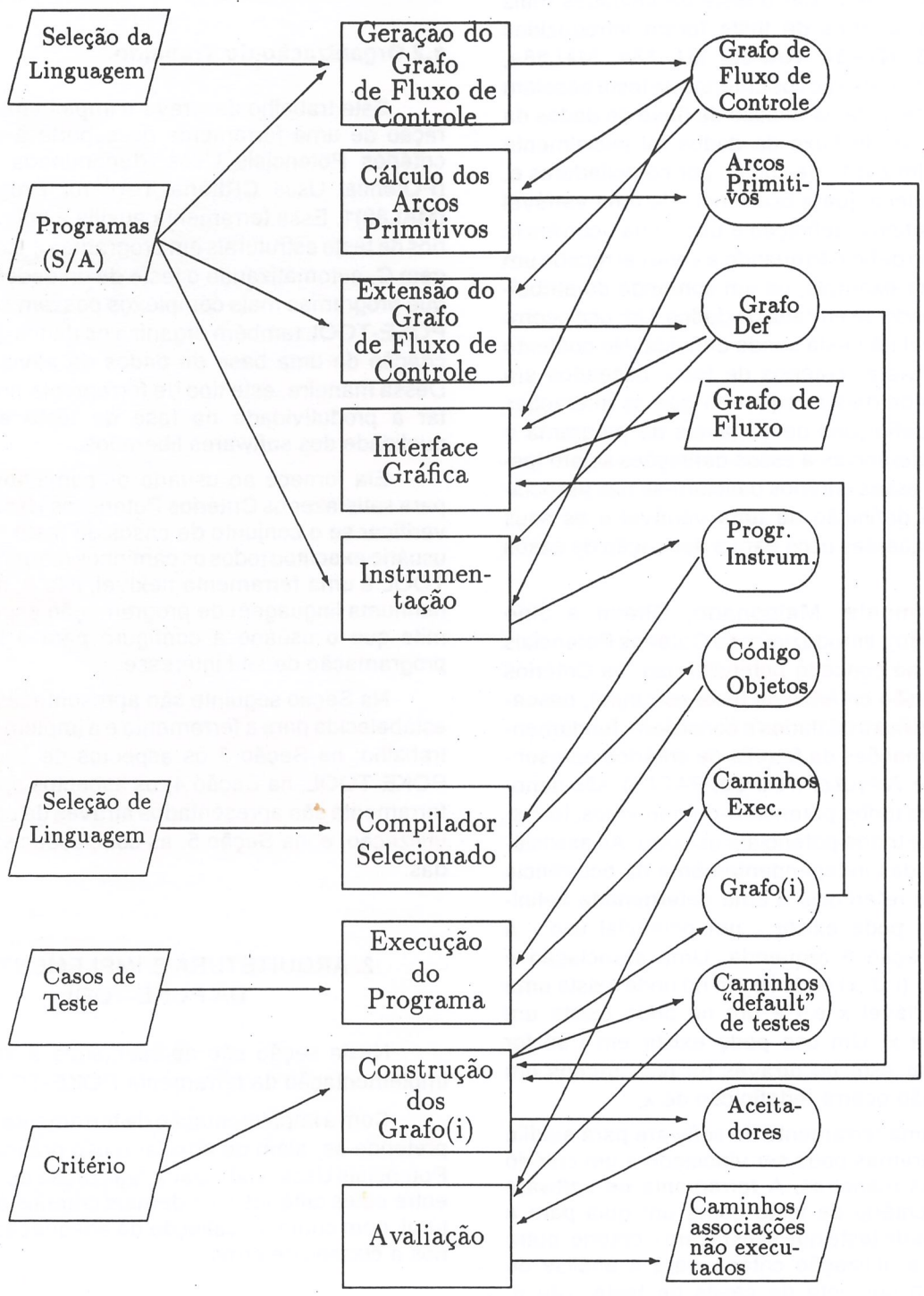


Figura 1 - Arquitetura funcional da Ferramenta de Teste Estrutural POKE-TOOL.

Inicialmente, foi implementada uma versão que apóia o teste de programas escritos na linguagem C e implementa todas as funções básicas que devem ser realizadas por uma ferramenta de teste estrutural de programas (DEU82).

Semelhantemente à ferramenta de teste implementada por Frankl e Weyuker - ASSET (FRA85) -, a POKE-TOOL tem como entrada o programa a ser testado, a seleção da critério a ser utilizado e um conjunto de casos de teste. Na hipótese do conjunto de casos de teste fornecido ser vazio, a POKE-TOOL fornecerá um conjunto de caminhos necessários para satisfazer o critério selecionado, orientando dessa forma a própria seleção de casos de teste. Se o conjunto de casos de teste fornecido não for vazio, será produzida uma relação de caminhos requeridos pelo critério mas ainda não executados.

De forma resumida, pode-se dizer que a partir do programa fonte, a POKE-TOOL determina o grafo de fluxo de controle (Geração de Grafo de Fluxo de Controle). A seguir, este grafo é estendido incorporando-se informações de fluxo de dados obtendo-se o grafo def (Extensão do Grafo de Fluxo de Controle); o conjunto de arcos primitivos é calculado, utilizando-se o algoritmo de redução de herdeiros de fluxo de dados (REHFLUXDA), obtendo-se o grafo com redução de herdeiros para fluxo de dados (Cálculo dos Arcos Primitivos) (CHU87). Esses arcos primitivos são utilizados para construir descritores (expressões regulares) dos caminhos (associações) requeridos. A Instrumentação auxilia na determinação dos caminhos efetivamente executados pelo conjunto de casos de teste fornecido. Os descritores são utilizados para verificar se o critério selecionado foi satisfeito; esta verificação dá-se pela implementação de aceitadores de expressões regulares. A partir do grafo def do conjunto de arcos primitivos constroem-se os grafo(i), caracterizando-se os arcos primitivos em cada um desses grafo(i) (Construção dos Grafo(i)). Em seguida, os descritores dos caminhos (associações) requeridos são elaborados. Na hipótese da POKE-TOOL ser utilizada na avaliação da adequação de um conjunto de casos de teste, são determinados os caminhos (associações) efetivamente executados e verifica-se se os aceitadores correspondentes aos descritores dos caminhos (associações) requeridos estão no estado final (o critério foi satisfeito); caso contrário, é fornecida ao usuário uma lista de caminhos requeridos pelo critério e não executados pelo conjunto de casos de teste (Avaliação).

Entre os caminhos requeridos pode existir um caminho não executável; a ferramenta POKE-TOOL, na sua primeira versão, não dá nenhum suporte para determinação da executabilidade de um determinado caminho - o que é uma questão indecidível; métodos heurísticos foram desenvolvidos e estão sendo incorporados à POKE-TOOL para tratar esse problema (FRA87) através de facilidades para a manipulação de caminhos não executáveis (VER92).

Observe-se que a ferramenta POKE-TOOL pode constituir-se também num suporte extremamente útil tanto às atividades de depuração como às de manutenção de programas.

2.2 Aspectos de Implementação da POKE-TOOL

A POKE-TOOL apóia a aplicação dos critérios Potenciais Usos para o teste de unidades. As unidades são entendidas aqui como procedimentos de uma linguagem procedimental.

A ferramenta pode ser configurada para testar programas escritos em linguagens procedimentais com gramáticas livres de contexto que satisfaçam algumas limitações. Basicamente, para configurar a ferramenta, é necessário que o configurador especifique o analisador léxico e o analisador sintático para a linguagem alvo. Chaim (CHA91) descreve como realizar estas tarefas; uma síntese desse processo é fornecida na Seção 3.

A POKE-TOOL é uma ferramenta interativa cuja operação é orientada para sessão de trabalho. Na sessão de trabalho, o usuário realiza todas as tarefas de teste, a saber: análise estática da unidade; preparação para o teste; submissão de casos de teste; avaliação de casos de teste; e gerenciamento dos resultados de teste. Na POKE-TOOL, a sessão de trabalho é dividida em duas fases: uma estática e outra dinâmica. Na fase estática a ferramenta analisa o código fonte, obtém as informações necessárias para a aplicação dos critérios e instrumenta o código fonte, com inserção de pontas de prova (instruções de escrita), que produzem um "trace" do caminho executado, gerando uma nova versão da unidade em teste - versão instrumentada -, que viabiliza a posterior avaliação da adequação de um dado conjunto de casos de teste. Terminada essa fase, a POKE-TOOL apresenta, sob solicitação do usuário, entre outras coisas, o conjunto de associações requeridas pelos critérios todos-nós e todos-arcos, o conjunto de caminhos requeridos pelo critério todos potenciais-du-caminhos e o conjunto de associações requeridas pelo critério todos potenciais-usos e todos potenciais-usos/du. Com estas informações o usuário pode projetar seus casos de teste a fim de que eles executem os caminhos ou associações exigidos.

A fase dinâmica consiste no processo de execução e avaliação de casos de teste. Porém, antes de executar os casos de teste, é necessário que se gere o programa executável a partir da versão instrumentada da unidade a ser testada. A POKE-TOOL, depois de execução dos casos de teste, realiza a avaliação destes de acordo com

um dos três Critérios Potenciais Usos. O resultado da avaliação é um conjunto de caminhos ou associações que restam ser executados para satisfazer os critérios e o percentual da cobertura do conjunto de casos de teste. Ainda nessa fase, a ferramenta fornece o conjunto de caminhos ou associações que foram executados, as entradas e saídas, bem como os caminhos percorridos na execução de cada um dos casos de teste. O processo de execução/avaliação deve continuar até que todos os caminhos ou associações restantes tenham sido satisfeitos (executados ou detectada a sua não executabilidade). Eventualmente, o usuário pode querer interromper a sessão de trabalho sem ter atingido uma das duas situações anteriores. Neste caso, a POKE-TOOL provê meios para a interrupção da sessão, armazenamento dos dados gerados e do estado da ferramenta até o momento. Posteriormente, pode-se recuperar a sessão de trabalho e recomeçar os testes.

A POKE-TOOL é composta de vários módulos que se comunicam através de arquivos. Esses módulos implementam funções ou parte de funções descritas anteriormente na arquitetura funcional da ferramenta. Na Figura 2 é apresentado um diagrama contendo os módulos e os diversos produtos gerados. Nessa figura, os retângulos representam os módulos, os losangos representam as entradas fornecidas pelos usuários à ferramenta e os círculos os produtos gerados; as linhas tracejadas representam o fluxo de controle e as linhas cheias o fluxo de informação.

O módulo **poketool** é o responsável pela comunicação entre a ferramenta e o usuário e pela seqüenciação das atividades de teste através da ativação dos demais módulos.

O módulo **li** é sensitivo à linguagem na qual está escrita a unidade em teste, pois realiza a tradução dessa unidade para uma versão escrita na linguagem intermediária (LI). A LI é uma linguagem cuja função é identificar os comandos que provocam desvio no fluxo de execução da unidade, ela possui dois tipos de comandos: comandos seqüênciais e comandos de desvio de fluxo. A cada comando da LI estão associados ponteiros que identificam o código fonte da unidade em teste relativo ao comando Li.

O módulo **chanomat** gera o garfo de fluxo de controle (GFC) da unidade - grafo de programa - e uma nova versão da unidade em LI onde cada comando está associado ao nó correspondente. Estes dois módulos implementam a função Grafo de Fluxo de Controle da POKE-TOOL.

O módulo **pokernel** é o responsável pelo restante da análise estática da unidade em teste, gerando as informações estáticas adicionais, necessárias ao teste dinâmico da unidade. O **pokernel** implementa as seguintes funções da POKE-TOOL: Cálculo dos Arcos Primitivos; Extensão do Grafo de Fluxo de Controle; Instrumentação, Construção dos Grafo(i) e Geração dos Descritores.

O módulo **gera executável** fornece as condições para a geração do programa executável da versão instrumentada e engloba a função Compilador Selecionado.

O módulo **executa caso de teste**, como o próprio nome diz, controla a execução dos casos de teste salvando as entradas, a saída e o caminho executado para cada caso de teste; implementa a função Execução do Programa da POKE-TOOL.

Finalmente, o módulo **avaliador** verifica quais os caminhos ou associações executados pelos casos de teste e fornece uma análise da cobertura do conjunto de casos de teste fornecido; implementa a função Avaliação da POKE-TOOL.

A atual implementação da POKE-TOOL foi desenvolvida para o sistema operacional MS-DOS e foi escrita na linguagem C, sendo que a configuração presente suporta o teste também de unidades (funções) escritas em C. Está em andamento a validação da configuração da POKE-TOOL para as linguagens COBOL e FORTRAN.

3. ASPECTOS DE CONFIGURAÇÃO DA POKE-TOOL

3.1. Introdução

A arquitetura da POKE-TOOL estabeleceu uma distinção entre as funções dependentes do código fonte das demais funções, permitindo que essas funções sejam isoladas em módulos distintos dos módulos que realizam funções independentes da linguagem. Isto permite que os módulos responsáveis por essas atividades possam ser reutilizados nas configurações da ferramenta para as diversas linguagens. Notadamente, o Cálculo dos Arcos Primitivos, a Geração dos Grafo(i) e Descritores e o mecanismo de Avaliação utilizado são completamente independentes da linguagem fonte.

De qualquer maneira, as informações dependentes da linguagem fonte são necessárias. Para obtê-las é necessário produzir um programa semelhante a um "front end" de compilador; este "front end" pode ser genérico ou específico para a linguagem fonte a ser tratada. O "front end" genérico é obtido através de algoritmos que são configurados para tratar os aspectos léxicos, sintáticos e semânticos específicos da linguagem em questão.

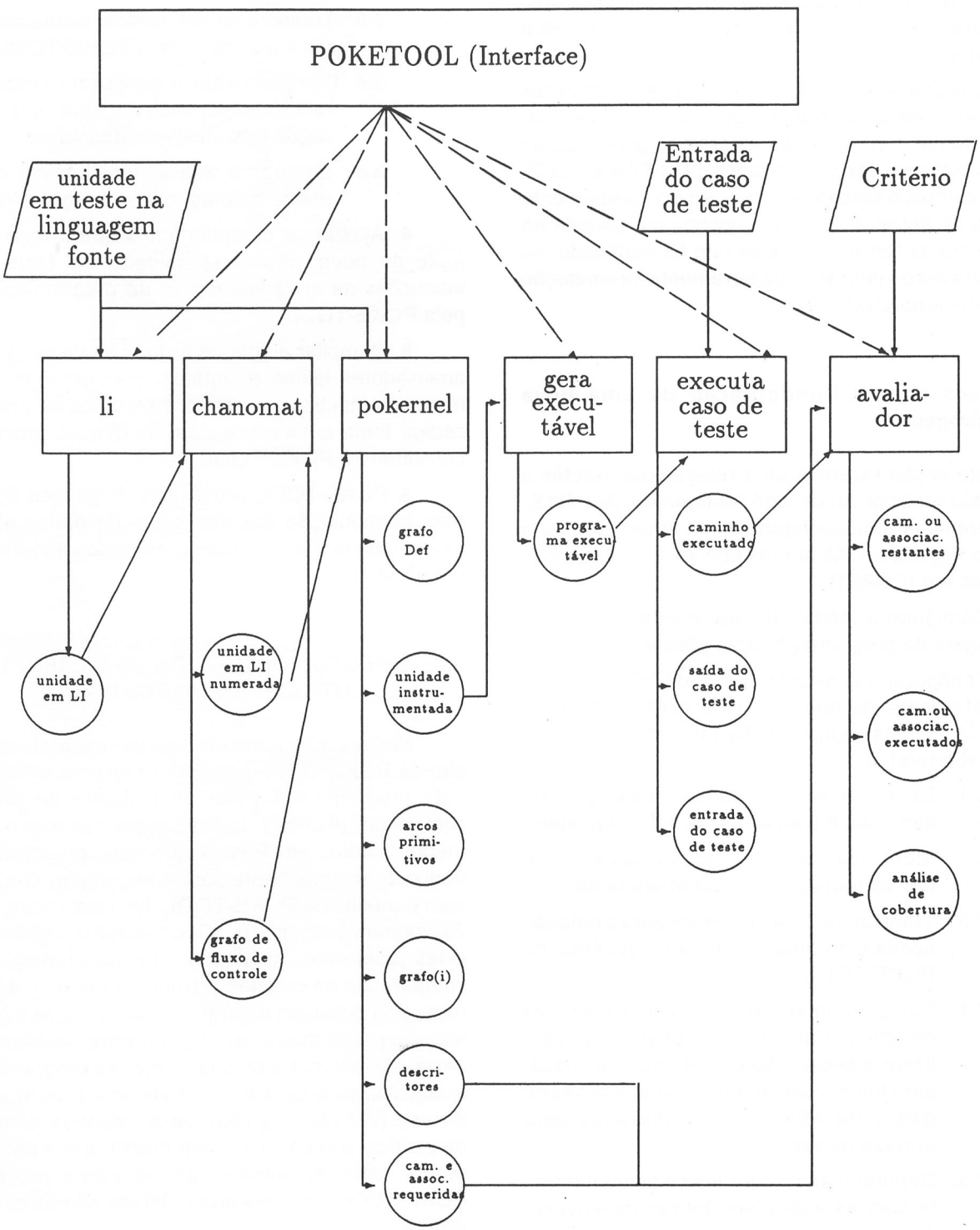


Figura 2 - Módulos da POKE-TOOL

A POKE-TOOL utiliza o enfoque genérico. A razão para essa opção foi privilegiar a reutilização de código (e esforço) em detrimento da otimização. Com o enfoque genérico, estamos reutilizando também parte do código usado nos módulos dependentes da linguagem e utilizando outras ferramentas, como descrito nos módulos **li** e **chanomat**.

Não obstante a existência de soluções otimizadas e, portanto, melhores do ponto de vista de eficiência, para o enfoque específico, os algoritmos genéricos de análise sintática e léxica utilizados pela POKE-TOOL têm uma complexidade (n log n, onde n é o comprimento da cadeia de entrada) que não constitui um gargalo no desempenho da ferramenta; este fato foi verificado, na prática, através da utilização da ferramenta na condução de um "benchmark"(MAL91).

3.2 Passos para a Configuração de uma nova Linguagem

Nesta seção fazemos uma relação das tarefas a serem realizadas por um usuário configurador da POKE-TOOL para obter uma configuração da ferramenta para uma nova linguagem. As tarefas abaixo descritas estão detalhadas em (CHA91).

1. Configurar o módulo **li** para realizar a tradução da linguagem de programação alvo para a LI.

2. Configurar o analisador léxico da POKE-TOOL. Essa tarefa demanda que o usuário conheça bem os aspectos léxicos da linguagem alvo e está dividida em algumas sub-tarefas.

   2.1. Desenvolver um autômato finito (SET81) que realize a análise léxica da linguagem.

   2.2. Identificar as ações semânticas associadas às transições do autômato finito.

   2.3. Transcrever esse autômato para a notação aceita pelo analisador léxico genérico da POKE-TOOL.

   2.4. Complementar o analisador léxico através de rotinas ("ações semânticas") que realizem a separação dos átomos da linguagem fonte, colocando-os nas estruturas de dados da POKE-TOOL utilizadas para armaze-ná-los.

   2.5. Depurar o analisador léxico conjuntamente com as ações semânticas desenvolvidas.

3. Configurar o analisador sintático da POKE-TOOL. Esta tarefa demanda que o usuário conheça bem a sintaxe e a semântica da linguagem alvo e está dividida em algumas sub-tarefas.

   3.1. Descrever a gramática da linguagem fonte na forma de grafo sintáticos (WIR76).

   3.2. Identificar os pontos onde ativar rotinas semânticas nos grafos sintáticos.

   3.3. Transcrever os grafos sintáticos para a notação aceita pela POKE-TOOL.

   3.4. Complementar o analisador sintático com rotinas semânticas que realizem a identificação das variáveis definidas.

   3.6. Depurar o analisador sintático conjuntamente com as rotinas desenvolvidas.

4. Ajustar os procedimentos que inserem código fonte na nova versão da unidade em teste com as instruções da nova linguagem de programação aceita pela POKE-TOOL.

5. Compilar e ligar os arquivos que constituem os analisadores léxico e sintático genéricos, as ações e rotinas semânticas, e os procedimentos de inserção de código fonte atualizados com os demais arquivos que constituem a POKE-TOOL.

A POKE-TOOL provê uma biblioteca de rotinas para manipulação das estruturas de dados utilizadas. Essas rotinas e as estruturas de dados são detalhadas em (CHA91).

4. ASPECTOS OPERACIONAIS E EXEMPLO DE UTILIZAÇÃO DA POKE-TOOL

Nesta seção apresenta-se um exemplo de utilização da POKE-TOOL para um programa escrito em C. Este programa faz parte do conjunto de programas contido em (KER81). Esses programas foram originalmente escritos em Pascal; um sub-conjunto deles foi traduzido manualmente para a linguagem C e testados com o auxílio da POKE-TOOL. No total foram testados 29 programas segundo os três critérios PU implementados; estes programas constituem um "benchmark" para a comparação de critérios estruturais de teste de programas, pois possuem algumas características interessantes: são programas escritos por programadores profissionais, são exemplos de boa técnica de programação e já foram utilizados para avaliar a eficácia de outros critérios de teste (WEY88, WEY90). Dessa maneira, com o intuito de verificar a eficácia dos critérios PU e sua relação com outros critérios, submeteram-se esses programas à POKE-TOOL; os resultados obtidos são discutidos em (MAL91).

O programa apresentado aqui é chamado **entab** (pag. 32, (KER81) e sua função é copiar a entrada via teclado para a saída, substituindo cadeias de espaços em branco por caracteres de tabulação de tal maneira que

visualmente a saída é igual à entrada, porém, com menor número de caracteres. Este programa encontra-se no arquivo ENTAB. C; por isso, a partir de agora, utilizaremos o nome de seu arquivo para referenciarmos este programa.

Para distinguir entre telas e mensagens do sistema, entradas do usuário e comentários, foi adotada a seguinte convenção: o que for relativo à ferramenta será descrito em "font" de máquina de escrever, o texto entrado pelo usuário é impresso em negrito e os comentários a respeito do funcionamento da ferramenta são escritos em itálico.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Vers. 0.2

Bem-vindo!!

INICIAÇÃO

Entre com o nome do arquivo que contem a unidade a ser testada (digite "fim"para terminar a sessão):

⇒**entab.c**

Mensagens .....

** Determinando o Grafo de Fluxo de Controle**

** Ferramenta Geradora de GFC's foi bem sucedida**

** Calculando os arcos primitivos...**

** Carregando Tabela de Transição léxica...**

** Fazendo a análise sintática do código fonte... **

** Gerando descritores... **

** O Núcleo da POKE-TOOL foi bem sucedido**

O usuário deve entrar com o nome do arquivo que contém a unidade em teste. Quando o usuário digita a tecla "ENTER", a POKE-TOOL dá início à fase estática da sessão de trabalho que consiste na análise do código fonte através dos módulos **li, chanomat e pokernel**. No decorrer desta fase, a ferramenta envia uma série de mensagens que indicam o andamento do processo de análise do código fonte.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

MENU PRINCIPAL

a. Resultados da Análise Estática
b. Resultados da Análise Dinâmica
c. Gera Programa Executável
d. Executa Caso de Teste
e. Avaliação Caso de Teste
f. Termina Sessão

Entre com a opção desejada:

⇒**a**

Mensagens .....

Terminando o processo de preparação da unidade em teste é apresentada a tela MENU PRINCIPAL, indicando que a sessão de trabalho pode prosseguir. Neste ponto a POKE-TOOL já determinou o grafo de fluxo de controle (GFC), o conjunto de arcos primitivos, o GFC estendido com as informações de fluxo de dados (grafo def), a versão instrumentada da unidade em teste, as associações e caminhos requeridos por cada critério Potencial Uso e os descritores utilizados na avaliação da cobertura desse mesmos critérios.

Selecionando a opção a no MENU PRINCIPAL, o usuário poderá visualizar os resultados obtidos da análise estática através da tela RESULTADOS DA ANÁLISE ESTÁTICA.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

RESULTADOS DA ANÁLISE ESTÁTICA

a. Arquivo Modificado
b. Grafo Def
c. N'os Requeridos pelo Critério Todos-N'os
d. Arcos Primitivos Requeridos pelo Critério Todos-Arcos
e. Potenciais Du-caminhos Requeridos
f. Associações Requeridas
g. Retorna para o Menu Principal

Entre com a opção desejada:

⇒**d**

Mensagens .....

O usuário, ao selecionar alguma das opções de menu acima, provoca uma chamada de sistema para o utilitário "more"; é apresentado na tela o arquivo associado à opção selecionada. Em seguida a tela RESULTADOS DA ANÁLISE ESTÁTICA é apresentada novamente.

As opções a, b, c, d, e e f do menu apresentam os resultados da análise estática da unidade. Na tela acima é selecionada a opção d; é então mostrado o arquivo ARCPRIM.TES que contém os arcos primitivos requeridos pelo critério todos-arcos.

ARCOS PRIMITIVOS DO MODULO ESTAB. C

arco (4, 5) e' primitivo
arco (4, 6) e' primitivo
arco (7, 8) e' primitivo
arco (9, 14) e' primitivo
arco (10, 11) e' primitivo
arco (10, 12) e' primitivo
arco (14, 2) e' primitivo
arco (14, 15) e' primitivo

Antes de executar um caso de teste é necessário gerar o programa executável que contém a unidade em teste instrumentada. Este programa será executado quando forem submetidos os casos de teste. Para gerar este programa deve-se selecionar a opção c do MENU PRINCIPAL.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

GERA PROGRAMA EXECUTÁVEL

OBSERVAÇÃO: Para gerar o programa executável para teste, você recebe o "prompt" do sistema operacional. Você deverá compilar e "linkar" o arquivo TESTEPROG. C junto com os seus outros arquivos no lugar do módulo a ser testado. O novo programa executável deverá ter o nome TESTEPROG.

Digite qualquer tecla para entrar no "shell" do sistema operacional...

Para retornar 'a POKE-TOOL digite "exit"...

A tela GERA PROGRAMA EXECUTÁVEL apresenta uma mensagem indicando como o usuário deve proceder para gerar o programa executável. Em seguida, o usuário recebe o "prompt" do sistema operacional para compilar a unidade em teste instrumentada e ligá-la com as demais unidades.

C:\USR\CHAIM\POKETOOL>make testeprog

Neste caso foi editado um arquivo "makefile" que compila e liga as unidades que compõem o programa executável.

C:\USR\CHAIM\POKETOOL>exit

Ao retornar à POKE-TOOL retorna-se à tela MENU PRINCIPAL.

Para executar um caso de teste basta selecionar a opção d e a tela EXECUTA CASO DE TESTE será apresentada.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

EXECUTA CASO DE TESTE

Seu programa necessita de parâmetros de entrada (S/N) (N)?

⇒**n**

Seu programa tem entrada pelo teclado (S/N) (N)?

⇒**s**

Mensagens .....

** A saída está sendo direcionada para o arquivo output.tes**

Na primeira vez que se executa um caso de teste a tela EXECUTA CASO DE TESTE pergunta se o programa aceita parâmetros de entrada na linha de comandos e se o programa aceita entrada via teclado. O usuário responde essas perguntas e o primeiro caso de teste é executado. Nas próximas execuções de casos de teste essas perguntas não serão feitas.

col 1 2 34 rest

col 1 2 34 rest

Tecle qualquer tecla para retornar à POKE-TOOL...

A unidade ENTAb.C não aceita parâmetros de entrada através da linha de comandos mas, se aceitasse, estes parâmetros de entrada seriam salvos. A entrada via teclado, a saída na tela do caso de teste e o caminho percorrido pelos casos de teste são também salvos. Depois de executado um caso de teste a tela MENU PRINCIPAL é reapresentada e o usuário pode decidir entre "rodar" mais um caso de teste ou avaliar os já executados com relação a algum critério PU.

Selecionando-se a opção e, obtém-se a tela AVALIAÇÃO.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

AVALIAÇÃO

a. Critério Todos-No's
b. Critério Todos-Arcos
c. Critério Todos-Potenciais-Usos
d. Critério Todos-Potenciais-Usos/du
e. Critério Todos-Potenciais-Du-Caminhos
f. Retorna para Menu Principal

Entre com a opção desejada:

⇒**c**

Mensagens .....

** Realizando a avaliação do caso de teste **
** Avaliação do caso de teste foi bem sucedida **
** Terminada a avaliação de um caso de teste **

O usuário tem três critérios PU, mais os critérios todos nós e todos arcos, implementados na POKE-TOOL; ele pode selecionar através do menu da tela AVALIAÇÃO um deles. O usuário, ao selecionar um dos critérios, por exemplo todos-potenciais-usos, faz com que a POKE-TOOL dê início ao processo de avaliação; durante a avaliação a ferramenta envia mensagens que indicam o andamento do processo. Terminada a avaliação, a POKE-TOOL apresenta o arquivo que contém as associações não executadas pelo conjunto de casos de teste avaliado. Depois de apresentado o arquivo, a tela AVALIAÇÃO é representa. Abaixo, apresentamos o ar-

quivo resultado obtido para a avaliação de oito casos de teste do exemplo ENTAB.C segundo o critério todos-potenciais-usos.

ASSOCIAÇÕES DO CRITÉRIO TODOS POT-USOS não executadas:

<1, (14,2), { col, tabstops }>
<1, (4,5), { col, tabstops }>
<2, (8,7), { newcol }>
<2, (7,8), { newcol }>
<4, (9,14), { newcol, tabstops }>
<4, (14,15), { newcol, tabstops }>
<4, (10,11), { newcol, tabstops }>
<5, (14,15), { col }>
<5, (2,3), { col }>
<5, (14,2), { col }>
<5, (9,14), { col }>
<5, (10,11), { col }>
<8, (14,15), { col }>
<8, (3,7), { col }>
<8, (6,3), { col }>
<8, (4,6), { col }>
<8, (4,5), { col }>
<8, (14,2), { col }>
<8, (9,14), { col }>
<8, (10,11), { col }>
<11, (10,12), { col }>
<11, (10,11), { col }>
<12, (7,8), { col }>

Cobertura Total = 71.250000
Média da Cobertura dos Grafo(i) = 70.431721

Neste ponto já é possível observar os resultados obtidos da fase dinâmica da sessão de trabalho. Para tanto, é necessário selecionar a tela RESULTADOS DA ANÁLISE DINÂMICA do MENU PRINCIPAL.

POKETOOL - Ferramenta de Apoio aos Critérios Potenciais Usos - Ver. 0.2

RESULTADOS DA ANÁLISE DINÂMICA

a. Entrada dos Casos de teste
b. Entrada do teclado
c. Saída dos Casos de Teste
d. Caminhos percorridos pelos Casos de Teste
e. No's executados para o Critério Todos-No's
f. Arcos Primitivos executados para o Critérios Todos-Arcos
g. Associações executadas para o Critério Todos-Potenciais-Usos
h. Associações executadas para o Critério Todos-Potenciais-Usos/du
j. Retorna para o Menu Principal

Entre com a opção desejada:

⇒**b**

Entre com o número do caso de teste (último caso de teste número 8):

⇒**7**

Mensagens .....

As opções **a, b, c** e **d** apresentam, respectivamente, os pârametros de entrada, entradas via teclado, as saídas na tela e os caminhos percorridos pelos casos de teste. Porém, essas opções estão relacionadas com vários arquivos porque existe um arquivo para cada caso de teste. Quando o usuário seleciona, por exemplo, a entrada do teclado, a POKE-TOOL pergunta a qual caso de teste o usuário está se referindo; se ele selecionar o sétimo caso de teste, a POKE-TOOL irá apresentar o arquivo com a entrada do teclado relativo a este caso de teste. Fato análogo ocorre com as outras opções acima e os conjuntos de arquivos associados.

col 1 2 34 rest

As opções **e, f, g, h** e **i** apresentam os nós, arcos, associações e caminhos efetivamente exercitados pelos casos de teste avaliados.

Para terminar a sessão de trabalho deve-se retornar ao MENU PRINCIPAL. Selecionando-se a opção **f** é iniciado o processo de término da sessão. A POKE-TOOL pergunta ao usuário se ele deseja que os arquivos gerados pela ferramenta sejam salvos em um sub-diretório com o nome da unidade; em caso afirmativo, a POKE-TOOL cria, se necessário, este sub-diretório e salva os arquivos nele; em caso contrário, a POKE-TOOL termina a execução sem salvar as informações geradas.

5. CONCLUSÕES

A POKE-TOOL analisa o código fonte obtendo o grafo de fluxo de controle e determinando os caminhos e associações requeridos pelos critérios PU; esses caminhos e associações são listados em relatórios que podem ajudar o usuário a projetar seus casos de teste; a POKE-TOOL gera também uma nova versão - versão instrumentada - da unidade em teste contendo instruções de escrita que fornecem o caminho percorrido pelo caso de teste; os casos de teste são avaliados e a ferramenta fornece relatórios que indicam quais os caminhos (associações) que não foram ainda satisfeitos para o critério selecionado, bem como os caminhos (associações) que foram satisfeitos para esse critério; ainda, a POKE-TOOL fornece uma série de informações (entradas, saídas e caminhos executados pelos casos de teste) que retratam como foi realizada e organizada a atividade de

teste. Como resultado da análise de cobertura, são apresentadas duas métricas que indicam a cobertura dos requisitos de teste (exigidos pelo critério selcionado) obtida para o conjunto de casos de teste.

A POKE-TOOL tem uma característica que a distingue das demais ferramentas de teste estrutural de software: ela é uma ferramenta configurável, ou seja, é possível obter instanciações da ferramenta para que ela suporte outras linguagens de programação.

A atual configuração da POKE-TOOL suporta o teste de programas escritos na linguagem C. Esta configuração foi validada através através da realização de um "benchmark"dos critérios PU utilizando um conjunto de programas da literatura (KER81). Este "benchmark" constituiu-se no teste de 29 programas segundo os três critérios PU implementados na POKE-TOOL. Os resultados obtidos desse "benchmark" (MAL91) são promissores e indicam que os critérios são factíveis de serem utilizados em ambiente industrial. mais ainda, a utilização da POKE-TOOL nesse "benchmark" mostrou que a ferramenta implementada é uma versão inicial de um produto e não apenas um protótipo.

A atual interface implementada na POKE-TOOL é simples e necessita o acréscimo de recursos gráficos que permitam apresentar ao usuário o grafo de fluxo de controle da unidade em teste e tornem a ferramenta mais amigável. A simplicidade da interface se deve à ênfase dada nesse trabalho à implementação e validação do cerne operacional da ferramenta; esses dois trabalhos estão em andamento e essas facilidades serão incorporadas na próxima versão da POKE-TOOL.

A POKE-TOOL é uma ferramenta que foi projetada para ser configurável para várias linguagens. A versão apresentada aqui está configurada para a linguagem C. Atualmente já existem outras configurações da POKE-TOOL para as linguagens COBOL e FORTRAN.

REFERÊNCIAS

(CHA91) CHAIM, M. L., MALDONADO, J. C. & JINO, M. Manual de Configuração da POKE-TOOL Campinas (SP), DCA/RT/008/91 - FEE/UNICAMP, 1991 (Relatório Técnico Interno).

(CHU87) CHUSHO, T "Test data selction and quality estimation based on the concept of essential branches for path testing" IEEE Trans. Software Eng., 13(5):509-517, May, 1983.

(DEU82) DEUTSCH, M. S. Software verification and validations, Englewood Cliffs, Prentice-Hall, 1982.

(FRA85) FRANKL, P. G. & WEYUKER, E. J. "Data flow testing tool" In: Proc. IEEE Softfair, San Francisco, CA, Dec. 1985 p. 46-53.

(FRA87) FRANKL, P. G. The use of data flow information for the selection and evaluation of software test data. Ph D. Thesis, New York University, 1987.

(HER76) HERMAN, P. M. "Data Flow Approach to Program Testing" Australian Computer Journal, 8(3), Nov., 1976.

(KER81) KERNIGHAN, B. W. & PLAUGER, P. Software Tools in Pascal. Addison-Wesley Publishing Company, Reading, Massachusetts, 1981.

(LAS83) LASKI, J. W. & KOREL B. "A data flow oriented program testing strategy" IEEE Trans. Software Eng., 9(3):347-354, May, 1983.

(MAL88a) MALDONADO, J. C. CHAIM, M. L. JINO, M. "Seleção de casos de testes baseada em fluxo de dados através dos critérios potenciais uso" In: Proc. Simp. Eng. Software, Canela, R. S., Out. 1988.

(MAL88b) MALDONADO, J. C. CHAIM, M. L. JINO, M. "Resultados do estudo de uma família de critérios de teste de programas baseado em fluxo de dados," Campinas (SP), DCA/RT/001/88 - FEE/UNICAMP, 1988 (Relatório Técnico Interno).

(MAL89) MALDONADO, J. C. CHAIM, M. L. JINO, M. "Arquitetura de uma Ferramenta de Software de Apoio aos Critérios Potenciais Usos" In: Proc. Cong. Nac. de Informática, 22, São Paulo, S. P., Set. 1989.

(MAL91) MALDONADO, J. C. Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software. Dissertação de Doutorado, Faculdade de Engenharia Elétrica UNICAMP, 1991.

(NTA84) NTAFOS, S. C. "On required element testing" IEEE Trans. Software Eng., 10(6):795-803, Nov., 1984.

(RAP82) RAPPS, S. & WEYUKER, E. J. "Data flow analysis techniques for test data selection" In: Proc. Int. Conf. Software Eng., Tokio, Japão, Sept., 1982 p.272-278.

(RAP85) RAPPS, S. & WEYUKER, E. J. "Selecting software test data using data flow information" IEEE Trans. Software Eng., 11(4):367-375, April, 1985.

(SET81) SETZER V. W. & de MELO I. S. H. A Construção de um Compilador terceira edição, Editora Campus, Rio de Janeiro, 1981.

(VER92) VERGILIO, R. S. Determinação de Caminhos Não-Executáveis na Ferramenta POKE-TOOL. Dissertação de Mestrado, Faculdade de Engenharia Elétrica UNICAMP, 1992.

(WEY88) WEYUKER E. J. "An Empirical Study of the Complexity of Data Flow Testing" In: Proc. Second Workshop on Software Testing, Verification, and Analysis, Banff, Canada, Jul. 1988, p.188-195

(WEY90) WEYUKER E. J. "The Cost of Data Flow Testing: An Empirical Study' IEEE Trans. Software Eng., 11(4):121-128, Fev., 1990.

(WIR76) WIRTH N. Algorithms + Data Structures = Programs. Englewood Cliffs, NJ, Prentice-Hall, 1976.

(URA88) URAL, H. & YANG, B. "A structural testing criterion" Information Processing Letters, 28(3):157-163, Jul., 1988.

UM MODELO DE INTERFACE DE APLICAÇÃO PARA AMBIENTES DE AUTOMAÇÃO INDUSTRIAL

AN APPLICATION INTERFACE MODEL FOR INDUSTRIAL AUTOMATION ENVIRONMENTS

Prof. Dr. Edmundo Roberto Mauro MADEIRA*
Prof. Dr. Manuel de Jesus MENDES**

ABSTRACT

This paper proposes a General Model of API ("Application Program Interface") for industrial networks with new functionalities added to the API from MAP ("Manufacturing Automation Protocol"). An implementation based on this Model for the MMS ("Manufacturing Message Specification") Protocol developed at UNICAMP is presented and some implementation issues are alanyzed.

KEY-WORDS: Computer Networks, Application Protocols, MAP Project, MMS Protocol, Application Interface.

RESUMO

Este artigo propõe um Modelo Geral de API ("Application Program Interface") para redes industriais com novas funcionalidades adicionadas à API do Projeto MAP "Manufacturing Automation Protocol"). Uma implementação baseada neste Modelo Geral para o Protocolo MMS ("Manufacturing Message Specification") desenvolvida na UNICAMP é apresentada e algumas questões de implementação são analisadas.

PALAVRAS-CHAVE: Redes de computadores, Protocolo de Aplicação, Projeto MAP, Protocolo MMS, Interface de Aplicação.

1 - INTRODUÇÃO

1.1 - Projeto MAP

O Projeto MAP foi criado pela General Motors no início da década de 80, com o objetivo de definir mecanismos e procedimentos de interconexão de equipamentos programáveis para a automação industrial. A meta era facilitar a interconexão das "ilhas de automação" existentes, para construir sistemas integrados da manufatura [MEN89]. Um outro Projeto com objetivos semelhantes aos do MAP, mas orientado para a automação de escritórios, foi criado pela Boeing no mesmo período e denominado TOP ("Technical and Office Protocol"). Uma promoção conjunta da General Motors com a Boeing na AUTOFACT em 1985, consistindo na exposição dos Projetos MAP e TOP, uniu os dois projetos num só, denominado MAP/TOP.

O Projeto MAP/TOP segue o Modelo de Referência OSI/ISO ("Open Systems Interconnection/International Organization for Standardization") e define os protocolos a serem adotados para cada uma das sete camadas do Modelo. Para a camada de aplicação, que provê serviços específicos orientados às aplicações, o Projeto MAP define o uso dos seguintes protocolos:

a - MMS ("Manufacturing Message Specification"), que suporta a transmissão de mensagens entre equipamentos programáveis, num ambiente de manufatura e controle de processos:

b - FTAM ("File Transfer Access and Management"), que provê serviços de gerenciamento e transferência de arquivos e de registros de dados;

c - DS ("Directory Service"), que provê serviços de administração de endereços e atribuição de nomes dos objetos OSI das diversas estações da rede;

d - NM ("Network Management"), que provê serviços de gerenciamento de rede relacionados com configuração, desempenho, falhas, contabilidade e segurança.

(*) Professor do Departamento de Ciências da Computação - IMECC/UNICAMP
(**) Professor da Faculdade de Engenharia Elétrica - UNICAMP/Porfessor do Instituto de Informática - PUCCAMP

O Projeto TOP define para a camada de aplicação os protocolos FTAM, DS, NM, VT ("Virtual Terminal") e MHS ("Message Handling System"). Os dois últimos providenciam os serviços de "login remoto" e troca de mensagens eletrônicas, respectivamente.

1.2 - Protocolo MMS

O Protocolo MMS [ISO9506] é o componente principal do Projeto MAP. Os equipamentos da manufatura são, em geral, simples, mas altamente especializados. Por exemplo, numa célula da manufatura, uma estação supervisora (PC) controla diversos equipamentos tais como CNR, CNC e CLP entre outros. Esses equipamentos podem ser interconectados numa rede local. Usando o MMS, os equipamentos da célula comunicação de uma maneira apropriada e padronizada. Uma estação da célula para comunicar com outros computadores da fábrica usará protocolos mais complexos, como o FTAM.

O Protocolo MMS é visto como um modelo "cliente/servidor", que aceita diversas requisições de serviços complexos do cliente para serem executadas em estações servidoras simples. O MMS modela os equipamentos programáveis (servidores) através de VMDs ("Virtual Manufacturing Devices"). Os VMDs tratam, de uma forma abstrata, dos elementos estruturais e dos objetos que compõem os dispositivos reais. Cada VMD contém pelo menos um domínio. Um domínio representa um subconjunto de recursos alocados de forma estática ou dinâmica (por exemplo, memória, entrada/saída e funções específicas de controle entre outros). Um VMD, além de um ou mais domínios, possui uma função executiva, estações de operador e uma memória de arquivos virtuais. A função executiva administra o acesso aos recursos do VMD. O MMS somente descreve o comportamento da servidora. Nesta, é necessário providenciar o mapeamento do VMD para os aspectos funcionais do Dispositivo Real da Manufatura.

Os serviços do MMS são agrupados nas seguintes unidades funcionais: Gerenciamento de Contexto, Suporte de VMD, Gerenciamento de Domínio, Gerenciamento de Invocação de Programa, Acesso a Variáveis Remotas, Gerenciamento de Semáforo, Comunicação de Operador, Gerenciamento de Evento, Gerenciamento de Jornal e Gerenciamento de Arquivos.

1.3 - Interface de Aplicação

Os protocolos de aplicação normalmente oferecem serviços complexos, apesar do fato de poderem ser modelados por máquinas de estados relativamente simples. Esse fato contrasta com o que geralmente ocorre nos protocolos das camadas mais inferiores do RM-OSI/ISO, onde máquinas de estados mais complexas modelam serviços comparativamente mais simples. A complexidade dos serviços de aplicação ocorre por serem estes em grande número e por seus parâmetros poderem ser de tipos estruturados. Os Protocolos MMS e FTAM são exemplos típicos.

Portanto, a camada de aplicação pode não oferecer seus serviços aos APs ("Application Processes") de uma maneira simples. Por outro lado, o AP normalmente deseja que os serviços dos sistemas de comunicação sejam oferecidos de uma maneira simples ("friendly"). Efetivamente, existe uma lacuna ("gap") a ser diminuída.

Uma solução possível é introduzir no sistema de comunicação uma interface entre a camada de aplicação e o AP para suavizar esta lacuna existente [FON89]. O Projeto MAP/TOP chama esta interface de API ("Application Program Interface") [MAP88a], enquanto que [MAD90b] chama de AI ("Application Interface") (Figura 1.1). A API é definida para os protocolos MMS [MAP88b], FTAM e comunicação privada, enquanto que a AI é uma generalização da API para os protocolos de aplicação existentes atualmente. A lacuna comentada pode ser reduzida a partir de adições de novas funcionalidades à AI.

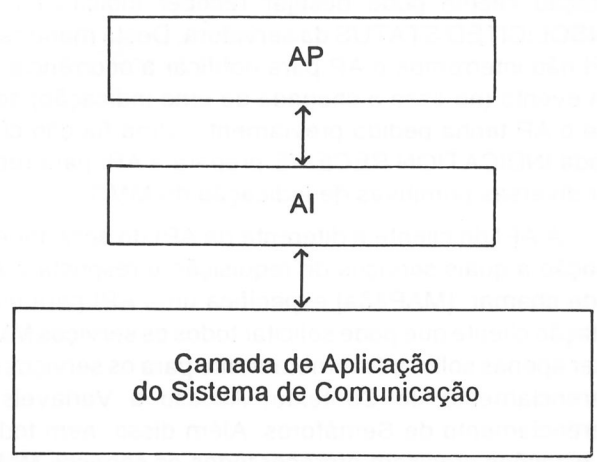


Figura 1.1 - Interface de Aplicação

O IEEE, através do Projeto POSIX ("Portable Operating System Interface") e de outros Projetos, está desenvolvendo APIs para alguns protocolos tais como FTAM, DS e MHS entre outros, para obter portabilidade de aplicações a nível de código fonte. Espera-se que estas padronizações do IEEE para APIs estejam concluídas no prazo de um a três anos [FLE92, HAZ92, NEW92].

A API é um conjunto explícito de funções e, portanto, provê portabilidade aos APs para diferentes ambientes através da padronização destas funções. A API

decrementa também o custo de programação e treinamento, pois os programadores podem usar a interface em ambientes múltiplos. A API executa algumas tarefas que seriam normalmente de responsabilidade do usuário, tais como: verificar se alguns valores de parâmetros das funções (serviços oferecidos) pertencem ao intervalo de valores válidos, preencher parâmetros "defaults" e dividir serviços complexos em serviços mais simples, entre outras.

A API para o Protocolo MMS é uma interface amigável e de alto nível para o usuário, oferecendo ao AP:

- Funções de baixo nível: somente uma primitiva de serviço MMS é invocada para cada chamada de função. READ e WRITE são exemplos de funções deste tipo para leitura e escrita de variáveis remotas, respectivamente:

- Funções de alto nível: diversas primitivas de serviços MMS são invocadas para uma chamada de função. FileGet é um exemplo de função deste tipo pois esta invoca três primitivas; fileOpen request, fileRead request e fileClose request;

- Funções respondedoras: a chamada de função prepara a API para receber primitivas de requisições remotas (denominadas indicações). Por exemplo, uma estação cliente pode desejar receber indicações de UNSOLICITED STATUS da servidora. Desta maneira, a API não interrompe o AP para notificar a ocorrência de um evento (no caso a chegada de uma indicação) sem que o AP tenha pedido previamente. Uma função chamada INDICATION RECEIVE prepara a API para receber diversas primitivas de indicação do MMS.

A API do cliente é diferente da API do servidor em relação a quais serviços de requisição e resposta o AP pode chamar. [MAP88a] específica uma API para uma estação cliente que pode solicitar todos os serviços MMS e ser apenas solicitada remotamente para os serviços de Gerenciamento de Contexto, Acesso a Variáveis e Gerenciamento de Semáforos. Além disso, nem todas as estações precisam oferecer todos os serviços MMS. Portanto, a API deve ser construída de maneira modular para permitir que apenas subconjuntos de funções sejam implementadas nas diversas estações, de acordo com as características próprias de cada estação.

2 - MODELO GERAL PARA INTERFACES DE APLICAÇÃO

Neste capítulo um Modelo Geral de Interface de Aplicação (AI), a partir da definição de API do Projeto MAP, será construído. A API do Projeto MAP é composta de três partes (Figura 2.1):

- Library (LIB): contém as funções que são utilizadas pelo AP para requisitar os serviços de rede desejados;

- Primitive Service Provider (PSP): é responsável pelo envio e recepção de primitivas para/da camada de aplicação;

- Três Procedimentos de Controle denominados:

- High Level Service Provider (HLSP): trata das funções de alto nível. Uma função de alto nível invoca diversas primitivas da camada de aplicação;

- Confirmed Service Provider (CSP): trata dos serviços com confirmação por parte da estação remota;

- Indication Filter and Arbitrator (IFA): trata da recepção de primitivas de indicação.

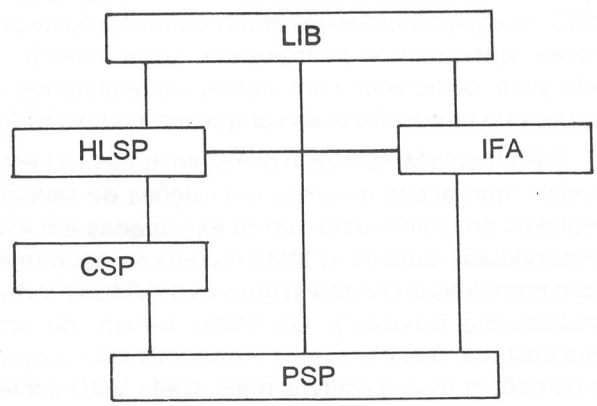


Figura 2.1 - Modelo de API do Projeto MAP/TOP

As particularidades do MMS podem levar a um esquema de resposta aos serviços solicitados remotamente em que a própria AI responda a estes serviços, tais como aqueles relacionados a gerenciamento de contexto, acesso a variáveis e gerenciamento de semáforos. Portanto, uma nova unidade funcional é adicionada à AI para processamento de indicações (solicitações remotas) denominada IP ("Indication Processing"). O IP tem as seguintes funções:

- Processar os serviços solicitados pelas primitivas de indicação aceitas;

- Acessar a memória principal e o sistema de arquivos local (chamando o VDRDB - ver abaixo);

- Solicitar ao CSP ou PSP (dependendo se o serviço é, ou não, confirmado) o envio de primitivas de resposta para as indicações aceitas.

Se a AI tem o bloco IP, o número de chamadas de INDICATION RECEIVE no AP pode ser decrementado, pois algumas primitivas de indicação são respondidas diretamente pela AI. Contudo, o AP precisa ainda chamar a função INDICATION RECEIVE para receber indicações que necessariamente são tratadas no AP, como por exemplo a indicação de ABORT e a indicação de CONCLUDE (para liberação da conexão).

Uma unidade funcional chamada VDRDB ("Virtual Device/Real Device Binding") é introduzida também na AI. Este bloco provê, quando necessário, o mapeamento entre os Dispositivos Virtuais da Manufatura (VMDs) e os aspectos funcionais dos Dispositivos Reais, além de ser usado pelo IP para acessar a memória e/ou o diretório de arquivos.

Por exemplo, a função auxiliar V-GET permite ao AP transformar a informação local (endereço e representação locais) em dado na representação MMS que pode ser enviado através da rede. A função auxiliar V-PUT transforma o dado MMS recebido da rede em representação local e coloca o dado em um endereço local. Estas funções usam o bloco VDRDB para realizar as transformações entre as representações MMS e local e para acessar a memória local.

Por outro lado, protocolos de aplicação interrelacionados são agrupados em Entidades de Aplicação (AEs - "Application Entities") na camada de aplicação. Por exemplo, a AE que contém o MMS deve conter também o Protocolo ACSE ("Association Control Service Element") para gerenciar as conexões. Desta forma, uma AI que requisita serviços de uma AE deve ter um componente lógico associado a cada protocolo de aplicação da AE. Os blocos funcionais TAOCF e RAOCF ("Transmission and Reception Association Object Control Function") supervisionam as interações entre os diversos componentes lógicos da AI para a transmissão e recepção de mensagens, respectivamente. No exemplo acima, os blocos TAOCF e RAOCF controlam as interações entre a AI-MMS e a AI-ACSE.

Por exemplo, na fase de estabelecimento de conexão, o AP providencia informação relacionada com o MMS e o ACSE para a AI-ACSE. A AI-ACSE interage com a AI-MMS para gerar e enviar uma primitiva para a camada de aplicação. Esta primitiva contém informação para o ACSE gerar a primitiva A_associate.req (requisição de estabelecimento de conexão) que tem no seu campo de informação a primitiva A_initiate.req (início de conexão MMS) gerado pelo MMS. Depois do estabelecimento da conexão, o AP chama funções da AI-MMS para realizar serviços específicos do MMS. A AI-MMS e a AI-ACSE têm, cada uma, os blocos LIB, HLSP, CSP, IFA, IP e VDRDB. Os blocos TAOCF e RAOCF supervisionam a interação entre os componentes lógicos AI-MMS e AI-ACSE.

O Modelo Geral proposto de AI é mostrado na Figura 2.2, e é analisado em [MAD90b].

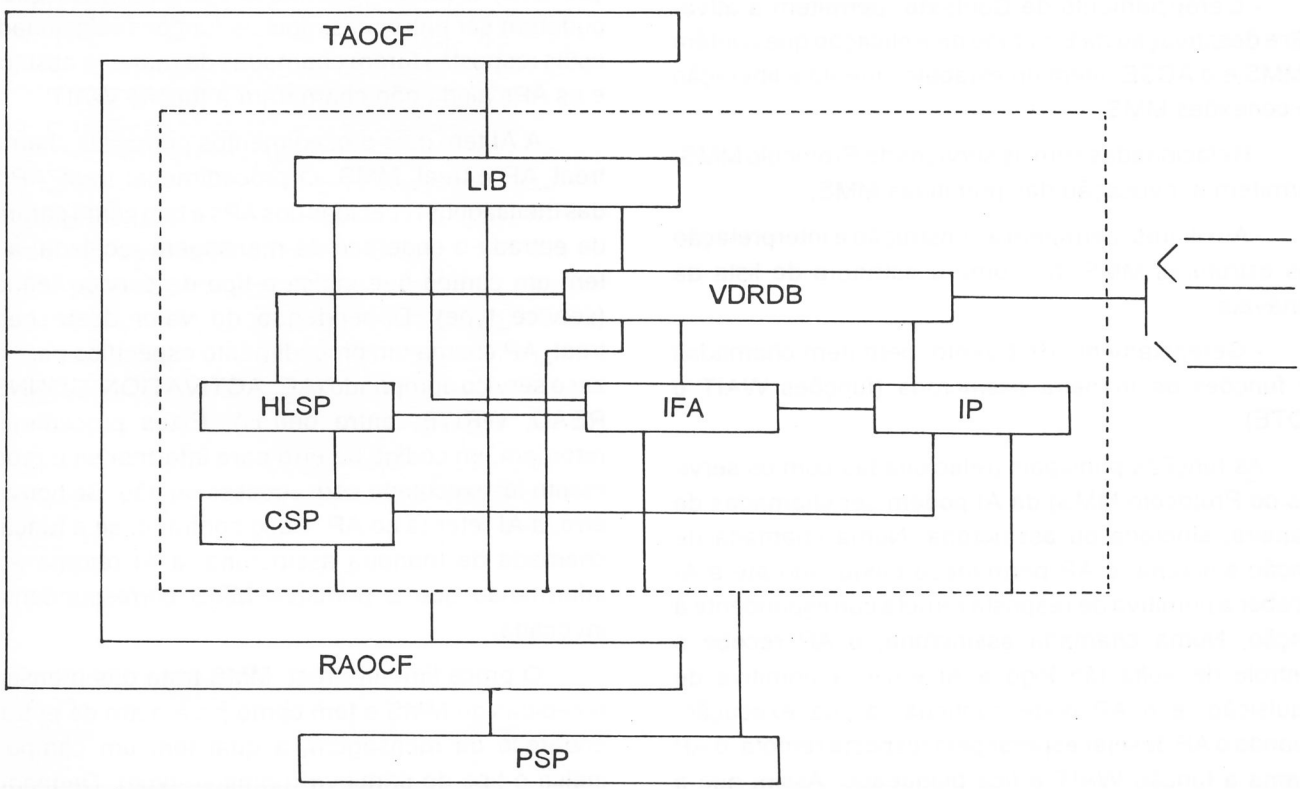


Figura 2.2 - Modelo Geral de Interface de Aplicação

O modelo proposto pode ser utilizado com outros protocolos de aplicação. Por exemplo, o Protocolo FTAM tem os conceitos de cliente e servidor e trabalha com uma estrutura de Sistema Virtual de Arquivos. O mapeamento entre o Sistema Virtual de Arquivos e o Sistema Real de Arquivos pode ser realizado na AI pelo VDRDB. O servidor é geralmente complexo e APs específicos respondem às requisições dos clientes. Contudo, o IP pode responder às requisições simples.

3 - EXEMPLO DE UMA IMPLEMENTAÇÃO DE INTERFACE DE APLICAÇÃO PARA O PROTOCOLO MMS

Usando o Modelo Geral proposto, uma AI para um subconjunto de serviços dos Protocolos MMS e ACSE foi implementada na UNICAMP, para suportar Gerenciamento de Contexto, Acesso a Variáveis e Suporte de VMD [MAD90a, MAD90b]. A AI foi escrita em C para microcomputadores compatíveis com o PC e roda sobre o sistema operacional DOS e um núcleo de tempo real que provê primitivas de comunicação e manipulação de tarefas. Uma nova versão da AI, também escrita em C, está sendo desenvolvida para rodar em estações de trabalho sobre o UNIX [MEN93].

Existem quatro tipos de funções da AI:

- Gerenciamento de Contexto: permitem a ativação e desativação da Entidade de Aplicação que contém o MMS e o ACSE, além do estabelecimento e liberação de conexões MMS;

- Relacionadas com os serviços do Protocolo MMS: permitem a invocação das primitivas MMS;

- Auxiliares: permitem a construção e interpretação das estruturas MMS, tal como a estrutura de lista de variáveis;

- Gerenciamento de Evento: permitem chamadas de funções de maneira assíncrona (funções WAIT e NOTE).

As funções principais (relacionadas com os serviços do Protocolo MMS) da AI podem ser chamadas de maneira, síncrona ou assíncrona. Numa chamada de função síncrona, o AP permanece bloqueado até a AI receber a primitiva de resposta remota correspondente à função. Numa chamada assíncrona, o AP recebe o controle de volta tão logo a AI envie a primitiva de requisição, e o AP pode continuar a sua execução. Quando o AP desejar esperar pela resposta remota, o AP chama a função WAIT e fica bloqueado. Assim que a primitiva de resposta remota chegue à AI, essa notifica o AP chamando a função NOTE.

A implementação utiliza os seguintes vetores [MAD92]: AE_LABEL_MAP, que define as invocações ativas da Entidade de Aplicação que contém os protocolos MMS e ACSE; CONNECTION_ID_MAP, que define as conexões estabelecidas, pendentes, abortadas e livres; INVOKE_ID_MAP, que define os serviços confirmados que ainda não obtiveram resposta remota; RETURN_EVENT_NAME_MAP, que define os nomes de eventos usados pelas chamadas assíncronas; LISTEN_MAP, que define as invocações de AE que estão esperando por pedidos remotos de conexão; WAIT_MAP, que define se um AP está, ou não, esperando por resposta remota de uma ou mais chamadas assíncronas e, finalmente, INDICATION_RECEIVE_MAP, que define se uma conexão está, ou não, esperando por primitivas de resposta remota.

As seguintes filas também são utilizadas: INTERFACE_API_AP, que contém mensagens (ainda com informações parciais) a serem retornadas aos APs como resposta às chamadas de funções da AI; INTERFACE_API_MMS, que contém as primitivas enviadas para o Protocolo MMS que requerem primitivas de resposta remota ou podem ser retransmitidas; e WAIT_QUEUE, que contém mensagens (com informação total) a serem retornadas aos APs, mas que não puderem ser entregues, pois as funções relacionadas a estas respostas foram chamadas de maneira assíncrona e os APs ainda não chamaram a função WAIT.

A AI tem dois procedimentos principais chamados treat_AP e treat_MMS. O procedimento treat_AP trata das mensagens recebidas dos APs e tem como parâmetro de entrada o endereço da mensagem recebida, a qual tem um campo que indica o tipo de serviço requerido (service_type). Dependendo do valor deste campo, treat_AP chama um procedimento específico para realizar o serviço apropriado (AE_ACTIVATION, CONNECT, READ, WRITE, entre outros). Estes procedimentos retornam um código de erro para informar se o procedimento foi executado com sucesso ou não. Se houve um erro, a AI retorna ao AP. Caso contrário, se a função foi chamada de maneira assíncrona, a AI retorna ao AP informando que a primitiva MMS correspondente foi invocada.

O procedimento treat_MMS trata das mensagens recebidas do MMS e tem como parâmetro de entrada o endereço da mensagem, a qual tem um campo que indica o tipo de primitiva (primitive-type). Dependendo do tipo de primitiva (resposta ou requisição remota - positiva ou negativa), um processamento específico é

executado. Depois, o treat_MMS verifica se a primitiva está relacionada a uma chamada de função síncrona ou assíncrona. Se esta chamada foi síncrona, o AP é notificado sobre o término do serviço. Se a chamada foi assíncrona, se o AP chamou previamente a função WAIT, o AP também é notificado sobre o término do serviço. Caso contrário, a mensagem de resposta ao AP é inserida na fila WAIT_QUEUE.

4 - UM EXEMPLO DE PROCESSO DE APLICAÇÃO

A Tabela 4-I mostra um exemplo de AP com chamadas à Interface de Aplicação para invocar serviços MMS. (1), (2), (11) e (12) são funções de gerenciamento de contexto; de (3) a (8) são funções relacionadas com o Protocolo MMS; e (9) e (10) são funções de gerenciamento de eventos.

Tabela 4-I

```
( 1)  mm_aeactivation (my_ae_name1,..., & ae_ label1, ...);
( 2)  mm_ connect (ae_label1,..., & con_id1);
( 3)  mm_status (con_id1,..., & generic_status1,...);
( 4)  mm_gnlist (con_id1,...,& name_type1,...);
( 5)  mm_gaattribute (con_id1,...);
( 6)  mm_gaattribute (con_id1,...);
( 7)  mm_write (con_id1, write_event, list1, & write_dcb1);
( 8)  mm_read (con_id1, read_event, listl, & read_dcb1);
( 9)  em_wait (wait_time1, & write_event);
(10)  em_wait (wait_time2, & read_event);
(11)  mm_conclude (con_id1,...);
(12)  mm_aedeactivation (ae_label1,...);
```

Inicialmente, o AP ativa a Entidade de Aplicação que contém os Protocolos MMS e ACSE chamando a função AE_ACTIVATION que tem como parâmetro de entrada o nome da AE, e como saída o AE_LABEL (identificador da invocação da AE). Depois, o AP requisita uma conexão através da função CONNECT que tem como parâmetro de entrada o AE_LABEL, e como saída o CONNECTION_ID (identificador da conexão). O AP pode requisitar outras conexões (outros CONNECTION-IDs serão providenciados). Depois disto, o AP chama algumas funções relacionadas com o MMS para pedir o estado da estação remota (Status), obter a lista de nomes de seus objetos (Get Name List), obter os atributos destes objetos (Get Access Attribute), e escrever e ler em objetos do tipo variável (Write e Read). Todas estas funções têm como parâmetros de entrada, pelo menos, o CONNECTION_ID e o RETURN_EVENT_NAME. As chamadas da função de gerenciamento de evento EM_WAIT determinam que o AP espere pela notificação de qualquer evento associado com chamadas assíncronas realizadas anteriormente. Finalmente, o AP requisita a liberação da conexão, chamando a função CONCLUDE, e então requisita o fim da invocação da AE, chamando a função AE_ACTIVATION.

5 - CONCLUSÃO

É recomendável apresentar os serviços de comunicação aos APs de uma maneira amigável devido à complexidade da camada de aplicação. Portanto, os sistemas de comunicação devem definir Interfaces de Aplicação. A complexidade destas interfaces depende das aplicações algumas vêzes, os APs podem requisitar diversos serviços da AI, tais como: confirmado, alto nível, mapeamento entre objeto real e virtual, respondedor, resposta automática, entre outros - outras vêzes, os APs podem requisitar apenas o envio e a recepção de mensagens genéricas. No primeiro caso, a AI pode ser construída de acordo com o Modelo Geral proposto. No segundo, a AI pode ter apenas uma biblioteca (LIB) simples além de Provedor de Serviço de Primitivas (PSP). Num caso intermediário, a AI pode, ou não, implementar alguns blocos e ligações do Modelo Geral, dependendo do tipo da estação, cliente ou servidora, e das particilariedades operacionais da estação.

A implementação mostrou que:

- Os blocos do Modelo Geral têm diversas atividades que independem de qual serviço é realizado. Portanto, as estruturas dos blocos são simplificadas;

- O esquema de processar requisições remotas na AI demonstrou ser eficiente, pois a AI não precisa chamar um AP para executar o serviço.

A AI também deve levar em consideração a existência de mais de um protocolo de aplicação numa AE para facilitar as tarefas dos APs. Os blocos TAOCF e RAOCF representam na AI esta característica da camada de aplicação.

REFERÊNCIAS

[FLE92] - FLEISCHMAN, E. - "An User's Guide to Data Communications APIs" - The Open Systems Newsletter; Parte 1: Maio 1992, pp. 1-8; Parte 2: Junho 1992, pp. 1-14

[FON89] - FONG, K. e REINSTEDLER, J. - "The Development of an OSI Application Layer Protocol Interface" - Computer Communication Review - ACM SIGCOMM, Julho 1989, pp. 21-57

[HAZ92] - HAZZARD, M. - "Distributed Services within IEEE TCOS/POSIX" - IEEE Network, Março 1992, pp. 22-23

[ISO9506] - ISO 9506 - "Manufacturing Message Specification" - Parte 1: "Service Specification"; Parte 2: "Protocol Specification", 1987

[MAD90a] - MADEIRA, E. R. M. e MENDES, M. J. - "Application Interface Model for Communication Software and an MMS Protocol Implementation Example" - Colloque International - CIM90 - Productique et Integrations, Bordeaux, France, Junho 1990

[MAD90b] - MADEIRA, E. R. M. e MENDES, M. J. - "An Application Interface Model for Communication Software" - IEEE Global Telecommunications Conference - GLOBECOM'90, San Diego, USA, Dezembro 1990

[MAD92] - MADEIRA, E. R. M. - "An Application Interface Model for Industrial Networks and its External View" - INFONOR'92 - Antofagasta, Chile, Dezembro 1992

[MAP88a] - MAP/TOP - "Application Interface Model and Specification Requirements", Junho 1988

[MAP88b] - MAP/TOP - "MMS Application Interface Specification", Junho 1988

[MEN89] - MENDES, M. J. - "Comunicação Fabril e o Projeto MAP/TOP" - IV EBAI - Santiago Del Estero, Argentina, 1989

[MEN93] - MENDES, M. J., MADEIRA, E. R. M. e outros - "SISDI-OSI: Sistema Didático para o Modelo OSI" - 11º Simpósio Brasileiro de Redes de Computadores - Campinas, SP, Maio 1993, pp. 3-19

[NEW92] - NEWNAN, O. - "SE-OSI: A Prototype Support Environmemt for Open Systems Interconnection" - Computer Communication Review - ACM SIGCOMM, Abril 1992, pp. 43-62

OPTMIX - UM MODELO PARA OBTENÇÃO DO MIX ÓTIMO DE PRODUÇÃO

OPTMIX - A OPTIMAL PRODUCTION PLAN MODEL

Denise H. Lombardo FERREIRA*
Walter CELASCHI**

ABSTRACT

Optmix is a model to determine the optimal production plan. The model uses linear and integer programming to evaluate the most profitable production program. Optmix model uses the most important variables and restrictions that affect production process and market place. The model needs GAMS software Version 2.05 for IBM-PC running under MS-DOS.

KEY-WORDS: Systems optimiization, manufacturing control system, Production Planning.

RESUMO

O Optmix determina o Mix Ótimo de Produção através de um modelo de programação linear e inteira. O modelo aponta o mix de produção de maior lucratividade e analisa o impacto das variáveis e restrições que afetam o mercado e o processo produtivo. O Optmix foi desenvolvido para microcomputadores IBM-PC com MS-DOS e utiliza o software GAMS - General Algebraic Modeling System - versão 2.05.

PALAVRAS-CHAVE: Otimização de Sistemas, Administração da Manufatura, Planejamento da Produção.

1. INTRODUÇÃO

O Brasil vive hoje um momento de transição em todos os aspectos que envolvem a produção industrial. Nossa indústria carece de organização, métodos, qualificação gerencial no nível médio, e também não possui níveis competitivos de preço e qualidade.

Algumas empresas que operam "no azul", carregam produtos e processos de lucratividade negativa, que não são identificados quando o resultado final é positivo. Baixos índices de produtividade e qualidade são observados nesta indústria que gera produtos com preços elevados e qualidade não compatível.

Com a abertura do mercado aos produtos estrangeiros é imperativo que nossa indústria adquira o mais rápido possível os padrões internacionais de eficiência, administração e controle que possibilitarão sua sobrevivência no novo cenário que se configura.

As empresas podem ter seu desempenho aferido por vários critérios, porém existe um que considera as principais variáveis operacionais, seu nome: Lucratividade, ou seja, a determinação do lucro decorrente de suas atividades produtivas em um determinado período de tempo.

O problema central da apuração da lucratividade reside no fato de a atividade industrial envolver um número muito grande de variáveis que têm correlação direta ou indireta com a lucratividade.

Manualmente é praticamente impossível avaliar a lucratividade de cada produto, pois seu custo depende de vários fatores, como por exemplo, o custo das matérias primas envolvidas, os custos diretos de fabricação e os custos indiretos da administração e de serviços auxiliares.

Além disso, o problema fica agravado pelo fato de a nossa moeda perder o valor no tempo, no caso dos pagamentos com data posterior à transação.

Levando-se em conta todos estes fatores, os autores propõem um modelo batizado de OPTMIX.

(*) Professora do Instituto de Informática - PUCCAMP
(**) Professor do Instituto de Informática - PUCCAMP e Consultor e microempresário de Informática

2. DESENVOLVIMENTO DO MODELO OPTMIX

O Optmix é um modelo que determina o Mix Ótimo de Produção, ou seja, o mix de produção que gera a maior lucratividade em função das principais variáveis e restrições que afetam a produção.

O modelo foi desenvolvido para indústrias de manufatura discreta com estrutura de produtos padronizada e com engenharia do processo baseada no conceito de tempo padrão.

O custo final é obtido através das matérias primas e dos processos industriais utilizados. Custos indiretos podem ser considerados como processos não industriais e rateados pelos diversos produtos.

A necessidade de produção pode ser obtida diretamente da carteira de pedidos ou de uma previsão de vendas. As capacidades de produção devem ser fornecidas pela engenharia industrial, assim como a disponibilidade de matérias primas nos próximos períodos ficam a cargo do departamento de compras, e finalmente os estoques máximos desejados devem ser definidos como estratégia da diretoria.

O Optmix foi concebido através de um projeto de pesquisa interdisciplinar que reuniu a experiência dos autores nas áreas de Pesquisa Operacional e Administração da Produção.

O modelo utiliza técnicas de Programação Linear e Inteira que podem ser vistas no protótipo apresentado a seguir que foi desenvolvido para microcomputadores padrão IBM-PC com sistema operacional MS-DOS e com o software GAMS versão 2.05 específico para problemas de pesquisa operacional.

OBJETIVO: Maximizar o Lucro Total considerando os dados e restrições abaixo:

DADOS:

1) Custo Padrão das matérias primas
2) Preço de venda dos produtos
3) Custo hora de cada centro de trabalho (hora normal/extra)
4) Estrutura dos produtos
5) Tempos de preparação de máquinas
6) Tempos de cada operação de produção

RESTRIÇÕES:

1) Disponibilidade de matérias primas nos períodos(*)
2) Demanda dos produtos nos períodos(**)
3) Capacidade dos centros de trabalho (hora normal/extra)
4) Estoques máximos dos produtos

(*) Possibilidade de reposição dos estoques por período.
(**) Pedidos em carteira ou previsão de vendas.

Os dados e restrições usados no modelo são apresentados no anexo I.

2.1 - DADOS USADOS NO MODELO

CPMP(M) CUSTO PADRÃO DA MATÉRIA-PRIMA M (DÓLAR POR UNIDADE)

/ M1 5 M2 8 M3 4 M4 12 M5 8 /

PV(P) PREÇO DE VENDA DO PRODUTO P (DÓLAR)

/ P1 14 P2 12 P3 12 P4 21 P5 16 P6 11 P7 35 P8 21 P9 16 P10 7

CPCT(C) CUSTO PADRÃO DO CENTRO DE TRABALHO C (DÓLAR POR HORA)

/ C1 40 C2 40 C3 50 C4 50 C5 30 /

CACT(C) CUSTO HORA EXTRA NO CENTRO DE TRABALHO C (DÓLAR POR HORA)

/ C1 60 C2 60 C3 75 C4 75 C5 45 /

TABLE N(P, M) USO DA MATÉRIA-PRIMA M NO PRODUTO P

	M1	M2	M3	M4
P1		1		
P2			1	
P3		1		
P4				1
P5			2	
P6	1		1	
P7		2		1
P8			1	1
P9		1		
P10	1			;

TP(C) TEMPO DE PREPARAÇÃO DO CENTRO DE TRABALHO C (HORAS)

/ C1 1.0 C2 0.5 C3 1.5 C4 0.5 C5 1.0/CACT(C)

TABLE HO(P, C) HORAS DO PRODUTO P (UMA UNIDADE) NO CENTRO C

	C1	C2	C3	C4	C5
P1	.01	.01	.01		
P2		.01	.02	.03	
P3	.01	.02			
P4			.03	.04	.05
P5		.02	.03	.04	
P6	.01			.01	.01
P7			.01		.01
P8	.01	.02	.01		
P9	.01		.01		.01
P10		.01	.02	.01	.02 ;

TABLE EIMP(M, H) DISPONIB. DA MATÉRIA-PRIMA M

	M1	M2	M3	M4
H1	3000	4000	4000	3000
H2	3000	4000	4000	3000

TABLE DE (P, H) DEMANDA (PEDIDOS) DO PRODUTO P (UNID/SEMANA)

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
H1	100	800	0	300	0	200	200	0	160	200
H2	200	0	700	0	700	100	300	100	120	0

CUSTO HORA EXTRA NO CENTRO DE TRABALHO C (DÓLAR POR HORA)

/ C1 60 C2 60 C3 75 C4 75 C5 45 /

TABLE CA(C,H) CAPACIDADE DO CENTRO DE TRABALHO C (HORAS/SEMANA)

44 horas para todos os centros de trabalhos

QUANTIDADE MÁXIMA DE HORAS EXTRAS NOS PERÍODOS 44 horas

EM(P) ESTOQUE MÁXIMO DO PRODUTO P (UNIDADES)

/ P1 200 P2 100 P3 0 P4 0 P5 200 P6 100 P7 0 P8 100 P9 200 P10 0 /

2.2 - EQUAÇÕES DE CÁLCULOS PARCIAIS

CPP(P) CUSTO PADRÃO UNITÁRIO DO PRODUTO P;

$$CPP(P) = \sum_{c} CPCT(C)HO(P,C) + \sum_{m} CPMP(M)N(P,M)$$

MPP(P) MARGEM PADRÃO UNITÁRIA DO PRODUTO P;

$$MPP(P) = PV(P) - CPP(P);$$

2.3 - RESULTADOS PRINCIPAIS/VARIÁVEIS DE DECISÃO

X(P,H) QUANTIDADE A SER PRODUZIDA DO PRODUTO P NO PERÍODO H

Y(C,H) HORAS ADICIONAIS NO CENTRO DE TRABALHO C NO PERÍODO H

W(P,H) CONTROLE DO TEMPO DE PREPARAÇÃO DOS CENTROS DE TRABALHO:

1 indica que o produto P é produzido no período H

0 caso contrário

Z LUCRO TOTAL (DÓLAR);

2.4 - FORMULAÇÃO MATEMÁTICA

MAXIMIZAR

$$Z = \sum_{P}\sum_{H} MPP(P)X(P,H) - \sum_{C}\sum_{H} CACT(C)Y(C,H) - \sum_{C}\sum_{P}\sum_{H} TP(C)CPCT(C)W(P,H)$$

RESTRIÇÕES

$$\sum_{P} N(P,M)X(P,H) \leq EIMP(M,H) \qquad \forall\ M,H \qquad (1)$$

$$\sum_{K\leq H} X(P,K) \geq \sum_{K\leq H} DE(P,K) \qquad \forall\ H,P \qquad (2)$$

$$\sum_{P} HO(P,C)X(P,H) + TP(C)W(P,H) \leq CA(C,H) + Y(C,H) \qquad \forall\ C,H \qquad (3)$$

$$\sum_{H} X(P,H) < EM(P) + \sum_{H} DE(P,H) \qquad \forall\ P \qquad (4)$$

$$X(P,H) - MW(P,H) \leq 0 \qquad \forall\ P,H$$

$$X(P,H) - W(P,M) \geq 0 \qquad \forall\ AP,H$$

$$X(P,H) \geq 0 \qquad \forall\ P,H$$

$$Y(C,H) \geq 0 \qquad \forall\ C,H$$

$$W(P,H) = 0 / 1 \qquad \forall\ P,H$$

3. ANÁLISE DOS RESULTADOS

Os resultados apresentados a seguir foram obtidos a partir das listagens emitidas pelo programa GAMS que processou as equações e os dados descritos no item anterior.

Inicialmente foram calculados o custo unitário de cada produto P considerando a utilização das matérias-primas e uso dos centros de trabalho, e a margem de lucro padrão de cada produto, obtida pela diferença entre o preço de venda e o custo padrão.

3.1 - CUSTO PADRÃO UNITÁRIO DO PRODUTO P

P1	9.300,	P2	6.900,	P3	9.200,	P4	17.000,	P5	12.300
P6	10.200,	P7	28.800,	P8	17.700,	P9	9.200,	P10	7.500

3.2 - MARGEM DE LUCRO PADRÃO UNITÁRIA DO PRODUTO P

P1	4.700,	P2	5.100,	P3	2.800,	P4	4.000,	P5	3.700
P6	0.800,	P7	6.200,	P8	3.300,	P9	6.800,	P10	-0.500

Da tabela acima pode-se observar que o produto P10 tem lucro negativo de 0.5 dólar por unidade, isto é, está sendo vendido abaixo do preço de custo.

Neste ponto é importante ressaltar que o mix de maior lucratividade não necessariamente maximiza os produtos de maiores lucros individuais. Isto ocorre pois o máximo global é diferente da soma dos máximos locais(1).

3.3 - QUANTIDADE A SER PRODUZIDA O PRODUTO P NO PERÍODO H (Var X)

	LOWER	LEVEL	UPPER	MARGINAL
P1 .H1	.	100.000	+INF	.
P1 .H2	.	400.000	+INF	.
P2 .H1	.	800.000	+INF	.
P2 .H2	.	100.000	+INF	.
P3 .H1	.	.	+INF	.
P3 .H2	.	700.000	+INF	.
P4 .H1	.	300.000	+INF	.
P4 .H2	.	.	+INF	.
P5 .H1	.	.	+INF	-5.250
P5 .H2	.	900.000	+INF	.
P6 .H1	.	400.000	+INF	.
P6 .H2	.	.	+INF	.
P7 .H1	.	500.000	+INF	.
P7 .H2	.	.	+INF	.
P8 .H1	.	.	+INF	-0.750
P8 .H2	.	200.000	+INF	.
P9 .H1	.	480.000	+INF	.
P9 .H2	.	.	+INF	.
P10 .H1	.	200.000	+INF	.
P10 .H2	.	.	+INF	-0.500

O **Mix Ótimo de Produção** é indicado por esta tabela. O modelo programou 100 unidades do produto P1 para o período H1 e 400 para H2. Este programa satisfaz a demanda de P1 (100 unidades para P1 e 200 para P2) e mantém a produção máxima no limite de 500 unidades.

Pode-se também observar que o modelo programou 400 unidades de P6 para o período H1 e zero para H2. Desta forma a produção do primeiro período atenderá as demandas de H1 (200 unidades), H2 (100 unidades) e sobrarão 100 unidades para o estoque. Os produtos P7 e P9 também tiveram seus programas de produção concentrados no período H1.

Os marginais negativos indicam redução no lucro total apresentado em 3.2) pelo aumento da produção no período. Exemplo: para cada unidade de P5 produzida no período H1 o lucro total será reduzido de US$ 5.25.

O modelo poderia considerar mais produtos e mais períodos tornando-se mais complexo em termos de alternativas possíveis para se obter a solução ótima. Entretanto, a formulação matemática seria a mesma.

3.4 - LUCRO TOTAL EM DÓLAR (Var Z)

LOWER	LEVEL	UPPER	MARGINAL
-INF	17809.000	+INF	.

O lucro máximo para os dados e restrições do anexo I é de US$ 17,809.00 sem restrições de limites inferior e superior.

A seguir serão analisadas as restrições impostas ao modelo e processadas pelo GAMS:

3.5 - RESTRIÇÃO DA DEMANDA DO MERCADO

	LOWER	LEVEL	UPPER	MARGINAL
P1 .H1	100.000	100.000	+INF	-0.750
P1 .H2	300.000	500.000	+INF	.
P2 .H1	800.000	800.000	+INF	-3.750
P2 .H2	800.000	900.000	+INF	.
P3 .H1	.	.	+INF	.
P3 .H2	700.000	700.000	+INF	.
P4 .H1	300.000	300.000	+INF	-1.250
P4 .H2	300.000	300.000	+INF	.
P5 .H1	.	.	+INF	.
P5 .H2	700.000	900.000	+INF	.
P6 .H1	200.000	400.000	+INF	.
P6 .H2	300.000	400.000	+INF	.
P7 .H1	200.000	500.000	+INF	.
P7 .H2	500.000	500.000	+INF	.
P8 .H1	.	.	+INF	.
P8 .H2	100.000	200.000	+INF	.
P9 .H1	160.000	480.000	+INF	.
P9 .H2	280.000	480.000	+INF	.
P10 .H1	200.000	200.000	+INF	-2.750
P10 .H2	200.000	200.000	+INF	.

Analisando a tabela de restrições de demanda pode-se concluir que o modelo conseguiu atender todos os pedidos dos produtos P1 a P10 para os períodos H1 e H2.

A coluna MARGINAL indica o aumento ou redução (caso negativo) no lucro pelo afrouxamento nas vizinhanças da restrição. No caso do produto P1 no período H1 encontramos -0.750. Este número indica a redução de US$0.75 no lucro total, a cada unidade adicional necessária do produto P1 no período H1.

3.6 - RESTRIÇÃO DE ESTOQUE DA MATÉRIA-PRIMA M

	LOWER	LEVEL	UPPER	MARGINAL
M1 .H1	-INF	600.000	3000.000	.
M1 .H2	-INF	.	3000.000	.
M2 .H1	-INF	1.580.000	4000.000	.
M2 .H2	-INF	1100.000	4000.000	.
M3 .H1	-INF	1200.000	4000.000	.
M3 .H2	-INF	2100.000	4000.000	.
M4 .H1	-INF	800.000	3000.000	.
M4 .H2	-INF	200.000	3000.000	.

Considerando que os níveis calculados tem folga com relação aos limites superiores, podemos concluir que as matérias primas disponíveis são suficientes para atentender a produção nos 2 períodos. Neste caso os "marginais" são iguais a zero significando que não heverá aumento no lucro caso aumente a disponibilidade de matérias primas.

3.7 - RESTRIÇÃO DAS CAPACIDADES DOS CENTROS DE TRABALHO

	LOWER	LEVEL	UPPER	MARGINAL
C1 .H1	-INF	16.800	44.000	.
C1 .H2	-INF	18.000	44.000	.
C2 .H1	-INF	14.500	44.000	.
C2 .H2	-INF	43.500	44.000	.
C3 .H1	-INF	44.000	44.000	75.000
C3 .H2	-INF	42.500	44.000	.
C4 .H1	-INF	44.000	44.000	75.000
C4 .H2	-INF	41.500	44.000	.
C5 .H1	-INF	39.800	44.000	.
C5 .H2	-INF	5.000	44.000	.

Esta tabela é importante pois possibilita a identificação dos gargalos de produção(1). Podemos observar que a capacidade de 44 horas dos centros de trabalho C3 e C4 foi esgotada no período H1. A coluna marginal indica que o lucro total será acrescido de US$ 75.00 a cada hora adicional na capacidade destes centros de trabalho.

3.8 - RESTRIÇÃO DA PRODUÇÃO MÁXIMA

	LOWER	LEVEL	UPPER	MARGINAL
P1	-INF	500.000	500.000	4.700
P2	-INF	900.000	900.000	5.100
P3	-INF	700.000	700.000	2.800
P4	-INF	300.000	300.000	.
P5	-INF	900.000	900.000	3.700
P6	-INF	400.000	400.000	0.050
P7	-INF	500.000	500.000	5.450
P8	-INF	200.000	200.000	3.300
P9	-INF	480.000	480.000	6.050
P10	-INF	200.000	200.000	.

3.9 - CONTROLE DO TEMPO DE PREPARAÇÃO DOS CENTROS DE TRABALHO

Houve a necessidade de se introduzir a variável inteira do tipo 0 ou 1 W(P,H) para controlar o tempo de preparação dos centros de trabalho. Ou seja, se W(P, H) = 0 não heverá preparação do centro de trabalho C que está representado na restrição CAPAC(C,H).W(P,H) = 0 não implica em produção zero para o período pois o centro pode já vir preparado do período anterior.

A variável inteira W(P,H) deve estar relacionada com a variável X(P,H) que representa a quantidade a ser produzida do produto P no período H.

Este fato pode ser representado pelas restrições.

$X(P,H) - MW(P,H) \leq 0$, onde M é um número muito grande

$X(P,H) - W(P,H) \geq 0$

$$\text{Se } W(P,H) = 0 \Longrightarrow \left\{ \begin{array}{l} X(P,H) \leq 0 \\ X(P,H) \geq 0 \end{array} \right\} \Longrightarrow X(P,H) = 0$$

$$\text{Se } W(P,H) = 1 \Longrightarrow \left\{ \begin{array}{l} X(P,H) \leq M \\ X(P,H) \geq 1 \end{array} \right\} \Longrightarrow 1 \leq X(P,H) \leq M$$

3.9 - HORAS ADICIONAIS NO CENTRO DE TRABALHO C NO PERÍODO H (Var Y)

	LOWER	LEVEL	UPPER	MARGINAL
C1 .H1	.	.	44.000	-60.000
C1 .H2	.	.	44.000	-60.000
C2 .H1	.	.	44.000	-60.000
C2 .H2	.	.	44.000	-60.000
C3 .H1	.	6.300	44.000	.
C3 .H2	.	.	44.000	-75.000
C4 .H1	.	1.500	44.000	.
C4 .H2	.	.	44.000	-75.000
C5 .H1	.	.	44.000	-45.000
C5 .H2	.	.	44.000	-45.000

O modelo também indica a quantidade de horas extras necessárias à produção do Mix ótimo. Neste caso o centro de trabalho C3 deverá ser utilizado por 6.3 horas no período 1 e o C4 por 1,5 hora no mesmo período. Os marginais negativos indicam que para cada hora adicional nestes centros de trabalho haverá redução no lucro total dos valores marginais.

BIBLIOGRAFIA

(1) GOLDRATT, E. M., Cox J., "A Meta, Administração dos Gargalos de Produção", IMAM, São Paulo, 1986.

(2) CORRÊA H. L., Gianesi. G. N., "Just in Time, MRP-II e OPT, um enfoque estratégico", Atlas, 1993.

(3) BROOKE A., Kendrick D. and Meeraus A., "GAMS - A User Guide" The Scientfic Press, South San Francisco, 1988.

(4) BAZARAA M. S., Jarvis J. J. and Sherali H. D., "Linear Programmning and Networks Flows", John Wiley, New York, 1990.

PROCESSAMENTO DE LINGUAGEM NATURAL: ATRIBUIÇÃO DE CASO AOS CONSTITUINTES DAS SENTENÇAS

NATURAL LANGUAGE PROCESSING: CASE ATTRIBUTION TO SENTENCE CONSTITUENTS

Adriano José Imbiriba CARNEIRO*
Andrea Barbosa MENANDRO**
José Eduardo CAMPOS**
Prof. João Luis Garcia ROSA***

ABSTRACT

This article describes the work that has been developed by the Research Group on Artificial Intelligence since 1991 when the group began its studies in the area of natural language processing with the aim of developing a system based on connectionist models for case attibution to sentence constituents of th Protuguese language. The developed system makes a semantic analysis of a stating phrase, that is, it indicates who the agent is, the patient of the action, the instrument made use of to accomplish the action and the modifyer (if so). In this article, tha functions and the features of the system are dealt with so as to understand how it works.

KEY-WORDS: Natural Language Processing, Connectionism, Neural Network, Semantic Analysis

RESUMO

Este artigo descreve o trabalho desenvolvido pelo Grupo de Pesquisa em Inteligência Artificial desde sua criação em 1991, quando iniciou estudos na área de processamento de linguagem natural, no sentido de desenvolver um sistema baseado em modelos conexionistas para atribuição de caso aos constituintes de sentenças da língua portuguesa. O sistema desenvolvido faz a análise semântica de uma frase declarativa, isto é, indica quem é o agente, o paciente da ação, o instrumento utilizado para realizar a ação e o modificador (se for o caso). Neste artigo, as funções e as características do sistema são abordadas de maneira a se entender o seu funcionamento.

PALAVRAS-CHAVE: Processamento de linguagem natural, Conexionismo, Rede neural, Análise semântica.

INTRODUÇÃO

A especificação de quem faz o quê para quem (atribuição de caso temático aos constituintes da sentença) não resolve todos os problemas, mas reflete um aspecto importante do processo de compreensão da sentença. Este processo envolve a consideração simultânea de diferentes fontes de informação, tais como a ordem da palavra, as restrições semânticas e o contexto. No sistema descrito, usou-se uma abordagem baseada em REDE NEURAL, pois ela provê um mecanismo que leva em consideração a ordem da palavra e as restrições semânticas.

(*) Aluno concluinte do Curso de Análise de Sistemas - PUCCAMP
(**) Graduados em Análise de Sistemas pela PUCCAMP
(***) Professor do Curso de Engenharia da Computação - PUCCAMP; Professor do Curso de Análise de Sistemas - PUCCAMP; Professor do Curso de; Especialização em Análise de Sistemas - PUCCAMP e Coordenador do Departamento de Eletrônica e Computação - II - PUCCAMP

FONTES DE INFORMAÇÃO

a. ORDEM DA PALAVRA

Por ordem da palavra entende-se a posição ocupada pela palavra analisada na frase.

(1) - O garoto quebrou o vaso.

(2) - O vaso quebrou a vidraça.

Podemos notar claramente que na frase (1) os casos dos constituintes da sentença seriam:

agente => garoto
verbo => quebrou
paciente => vaso

Entretanto, na frase (2) há uma mudança no papel da palavra vaso:

instrumento => vaso
verbo => quebrou
paciente => vidraça

b. RESTRIÇÕES SEMÂNTICAS

Por restrições semânticas entendem-se os casos possíveis de serem assumidos por uma palavra em uma frase.

(3) - O garoto quebrou a vidraça com a pedra.

(4) - O garoto quebrou a vidraça com a cortina.

Na frase (3) os papéis de caso dos constituintes da sentença seriam:

agente => garoto
verbo => quebrou
paciente => vidraça
instrumento => pedra

já na frase (4) os papéis de caso dos constituintes seriam:

agente => garoto
verbo => quebrou
paciente => vidraça
modificador => cortina

CONTEXTO

O sistema leva em consideração a sentença em que a palavra é empregada, uma vez que há casos em que somente a inserção desta em uma dada situação poderá levar à completa e perfeita compreensão da idéia descrita na frase.

FUNÇÕES DO SISTEMA

O sistema é capaz de aprender a fazer a atribuição de caso (baseado em experiência com sentenças e suas representações de caso), generalizar o que aprendeu para novas sentenças, selecionar contextualmente leituras apropriadas de palavras ambíguas e selecionar apropriadamente o caso dos constituintes de acordo com suas características semânticas.

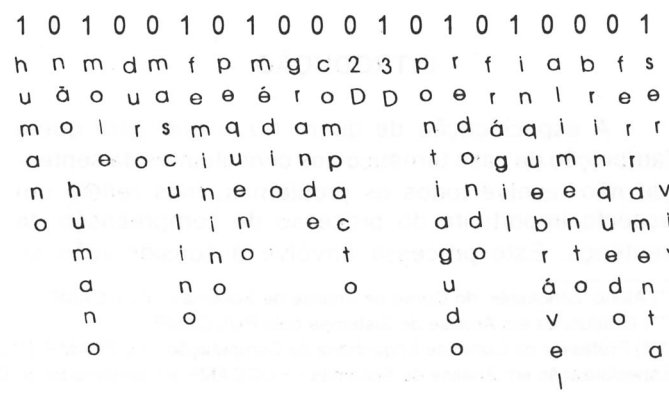
DESCRIÇÃO DO SISTEMA

O sistema consiste de uma rede neural com três camadas (onde a primeira camada representa a sentença a ser analisada e a terceira sua estrutura de caso) e um dicionário com palavras usadas pelo sistema (e suas respectivas microcaracterísticas semânticas). O algoritmo conexionista empregado foi o "backpropagation".

Cada palavra é representada como um conjunto de microcaracterísticas semânticas (20 no total), que neste caso foram escolhidas para capturar o que se considerou dimensões importantes nos significados das palavras, assumindo valores de acordo com a seguinte convenção [MCC86]:

0 = microcaracterística ausente;
0,5 = pode ser que sim/pode ser que não (o sistema decide);
1 = microcaracterística presente.

Por exemplo, na definição das microcaracterísticas do conceito MENINA tem-se:



O conjunto que representa a sentença a ser analisada é formado pela junção de todas as microcaracterísticas semânticas das palavras formadoras da sentença. Como as sentenças tratadas por este sistema têm no máximo quatro palavras, o conjunto tem 80(oitenta) microcaracterísticas no total. Para sentenças com menos de quatro palavras o sistema insere um conjunto nulo de microcaracterísticas {0,0,0,0,...,0} para completar a entrada.

Por exemplo:

O CACHORRO SE MOVEU
||
V
CACHORRO MOVEU
||
V
CACHORRO MOVEU NULO NULO

(0,1,1,0,...,0,1,1,0,...,0,0,0,...,0,0,0,0,0,...,0)

DESCRIÇÃO DO DICIONÁRIO

Usando as microcaracterísticas anteriores, o dicionário da rede fica como descrito na Figura 01:

(0,1,1,0,1,0,1,0,0,0,5,0,5,0,5,0,1,1,0,1,0,0,0),	{alimento}
(0,1,1,0,0,1,1,0,0,1,0,0,0,1,1,0,1,0,0,0),	{batata}
(0,1,1,0,0,1,1,0,0,1,0,0,0,1,0,1,0,1,0,0),	{bola}
(0,1,1,0,0,1,1,0,0,0,0,0,0,1,1,0,0,1,0,0),	{boneca}
(0,1,1,0,1,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0),	{cachorro}
(0,1,0,1,0,1,1,0,0,0,1,0,1,0,1,0,1,0,0,0),	{cenoura}
(0,1,0,1,0,1,1,0,0,0,1,0,0,1,0,1,0,1,0,0,0,1),	{colher}
(0,1,1,0,0,1,0,1,0,0,0,1,0,1,1,0,0,0,0),	{cortina}
(0,1,0,1,0,1,0,0,1,0,0,0,1,0,0,1,0,0,0,0),	{escrivaninha}
(0,1,1,0,0,1,1,0,0,0,5,0,0,0,1,0,5,0,5,0,5,0,0,0),	{galinha}
(0,1,0,1,1,0,1,0,0,0,1,0,1,0,0,1,0,0,0,1),	{garfo}
(1,0,1,0,1,0,0,1,0,0,0,0,0,1,0,1,0,0,0,0),	{garoto}
(1,0,1,0,1,0,0,0,1,0,0,0,0,1,0,1,0,0,0,0),	{homem}
(0,1,1,0,1,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0),	{leão}
(0,1,1,0,1,0,0,1,0,0,0,0,1,0,0,1,0,1,0,0,0),	{lobo}
(0,1,0,5,0,5,1,0,1,0,0,0,0,0,1,0,0,1,0,0,0,5,0),	{macaco}
(0,1,1,0,1,0,1,0,0,0,1,0,0,1,1,0,1,0,0,0),	{macarrão}
(0,1,0,1,1,1,0,1,0,0,0,1,0,0,1,0,1,0,0,1,0),	{martelo}
(1,0,1,0,0,1,0,1,0,0,0,0,0,1,0,1,0,0,0,0),	{menina}

(1,0,1,0,0,1,0,0,1,0,0,0,0,1,0,1,0,0,0,0),	{mulher}
(0,1,1,0,0,1,0,1,0,0,0,0,0,1,1,0,0,0,0,0),	{ovelha}
(0,1,0,1,0,1,1,0,0,0,0,0,1,0,0,1,0,0,0,0),	{pedra}
(0,1,0,1,1,0,1,0,0,0,0,1,0,1,1,0,0,0,0,0),	{prato}
(0,1,1,0,1,0,1,0,0,0,0,1,0,1,1,0,1,0,0,0),	{queijo}
(0,1,0,1,1,0,1,0,0,0,1,0,0,1,1,0,0,0,0,0),	{vaso}
(0,1,0,1,0,1,0,1,0,0,0,1,1,0,1,0,0,0,0,0),	{vidraça}
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),	{<nulo>}
(1,0,0,0,0,1,0,0,0,0,0,1,0,1,0,0,0,1,0,1),	{bateu}
(1,0,1,0,0,0,1,0,0,0,1,0,0,1,0,1,0,1,0,0,1,0),	{comeu}
(1,0,1,0,1,0,0,0,0,0,0,1,1,0,0,1,0,0,1,0),	{moveu}
(1,0,1,0,0,1,0,0,1,0,0,0,0,1,0,0,0,1,0,1),	{quebrou}
(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),	{avf}
(0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),	{avp}
(0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),	{avpm}
(0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),	{avpi}
(0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),	{ivp}
(0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0),	{pv}
(0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0),	{avs}

Figura 1

FUNCIONAMENTO DO SISTEMA

O sistema recebe como entrada uma representação canônica da sentença (o conjunto de microcaracterísticas dos elementos da sentença), sem os artigos, conjunções e quantificadores, fornecida pelo analisador sintático. A rede gera na saída uma única unidade ativa que representa a estrutura de caso da sentença (Figura 02).

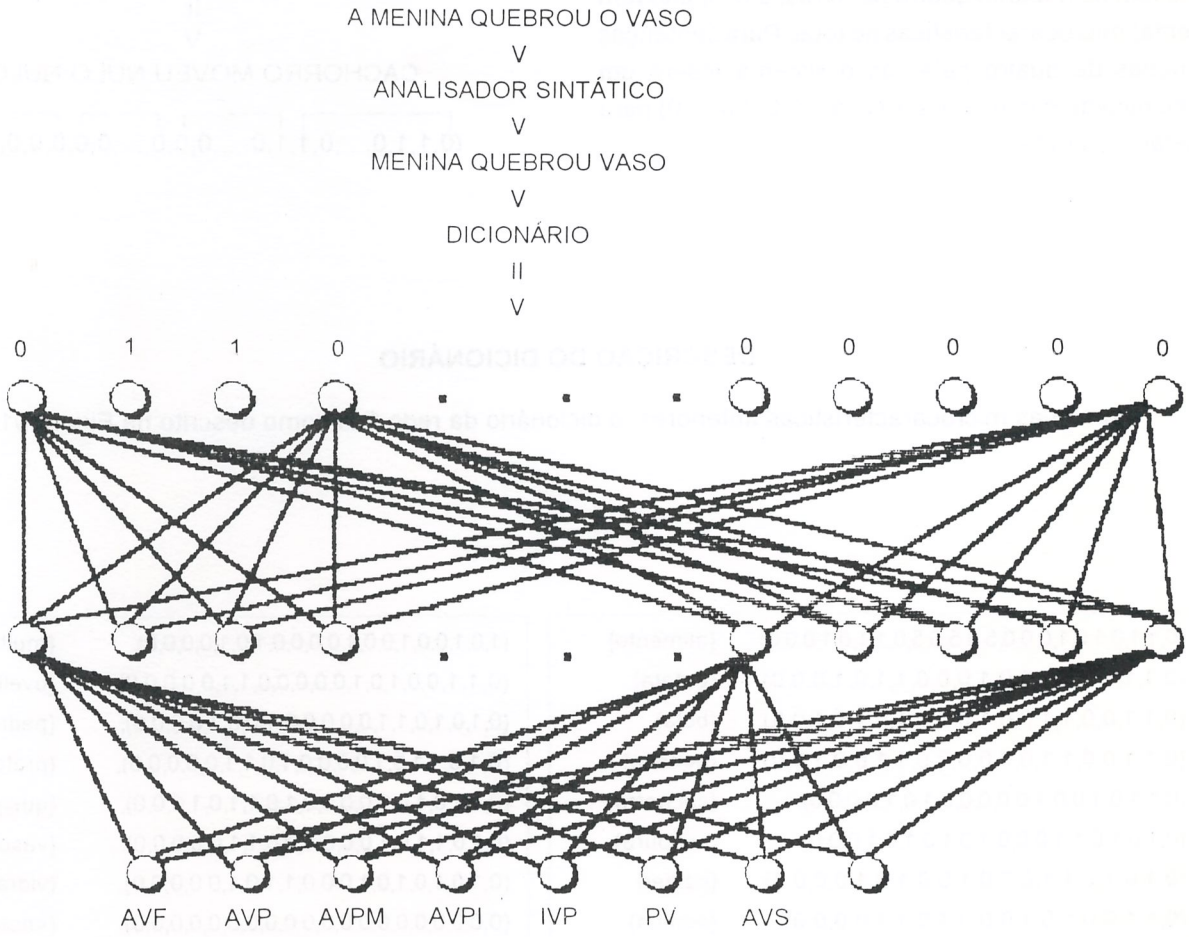


Figura 2

As estruturas de caso que o sistema pode atribuir à sentença são:

* AVF = Agente Verbo Paciente-Implícito
* AVPI = Agente Verbo Paciente Instrumento
* AVP = Agente Verbo Paciente
* IVP = Instrumento Verbo Paciente
* AVPM = Agente Verbo Paciente Modificador
* PV = Paciente Verbo
* AVS = Agente Verbo Paciente = Agente

DETALHES DO APRENDIZADO

Para realizar o aprendizado o sistema usou o algoritmo "Backpropagation" [RIC91] com fator de correção de erro Mi = 0.2 e função de ativação F(Alfa) = 1/(1+ exp(-Alfa)). Para gerar as sentenças usou-se um Gerador Aleatório de Sentenças que escolhe uma das estruturas geradoras de sentenças conhecidas e gera uma frase qualquer com aquela estrutura e sua respectiva estrutura de caso (ver Figura 03), onde:.

humano	[HOMEM, MENINA]
alimento	[CENOURA, GALINHA]
utensílio	[COLHER, GARFO]
animal	[GALINHA, CACHORRO]
predador	[LEÃO, LOBO]
presa	[GALINHA, OVELHA]
objeto-frágil	[VASO, VIDRAÇA]
quebrador	[BOLA, MARTELO]
coisa	[HOMEM, OVELHA]
propriedade	[BOLA, CACHORRO]
batedor	[BOLA, MARTELO]
objeto	[BOLA, MACACO]

Figura 3

ESTRUTURA GERADORA DE SENTENÇAS	ESTRUTURA DE CASO
O humano comeu	AVF
O humano comeu o alimento	AVP
O humano comeu o alimento com o alimento	AVPM
O humano comeu o alimento com o utensílio	AVPI
O animal comeu	AVF
O predador comeu a presa	AVP
O humano quebrou o objeto-frágil	AVP
O humano quebrou o objeto-frágil com o quebrador	AVPI
O quebrador quebrou o objeto-frágil	IVP
O animal quebrou o objeto-frágil	AVP
O objeto-frágil quebrou	PV
O humano bateu na coisa	AVP
O humano bateu no humano com a propriedade	AVPM
O humano bateu na coisa com o batedor	AVPI
O batedor bateu na coisa	IVP
O humano se moveu	AVS
O humano moveu o objeto	AVP
O animal se moveu	AVS
O objeto moveu	PV

CONCLUSÃO

O objetivo inicial do trabalho foi desenvolver um analisador semântico para a língua portuguesa, utilizando-se um sistema que levasse em consideração as particularidades das palavras envolvidas.

Os resultados obtidos, considerando-se simultaneamente tanto a ordem da palavra quanto as restrições semânticas, demonstraram que é possível determinar os casos dos elementos na frase.

Com a implementação do projeto e sua execução pode-se afirmar que seu objetivo foi atingido, possibilitando ainda uma continuidade deste trabalho, pois como é possível determinar o papel de cada palavra na frase, um terceiro módulo poderia ser acrescentado ao sistema, módulo este que seria responsável pelo armazenamento e reagrupamento das frases para consultas futuras.

BIBLIOGRAFIA

[DAH83] DAHL, Veronica **"Logic programming as a representation of knowledge"**. IEEE Computer, pages 37-41, October, 1983.

[WAL85] WALTZ, David L.; Pollack, Jordan B. **"Massively parallel parsing: A strongly interactive model of natural language interpretation"**. Cognitive Science 9, pages 51-74 - 1985.

[MCC86] MC CLELLAND, J. L.; Kawamoto, A. H. **"Mechanisms of sentence processing: Assigning roles to constituents of sentences"**. In 'Parallel Distributed Processing - Explorations in the microstructure of cognition' - Volume 2: Psycological and Biological Models - A Bradford Book, The MIT Press, 1986.

[CRO87] GROSSBERG, Stephen **"Competitive learning: From interactive activation to adaptative resonance"**. Cognitive Science 11, pages 23-63, 1987.

[LIP87] LIPPMANN, Richard P. **"An introduction to computing with neural nets"**. IEEE ASSP Magazine, Abril, 1987, pages 4-22.

[RIC91] RICH, E. & Knight, K. **"Artificial Intelligence"**. 2nd. edition. McGraw-Hill, 1991.

[ROS93] ROSA, J. L. G. & Netto, M. L. A. **"Redes Neurais e Lógica Formal em Processamento de Linguagem Natural"**. (artigo a ser publicado).

Revista do Instituto de Informática
Publicação Semestral do Instituto de Informática
PUCCAMP

A Revista do Instituto de Informática aceita colaborações que lhe forem espontaneamente enviadas, reservando-se o direito de publicá-las ou não, conforme avaliação dos Editores. Os temas abordados serão relacionados com as várias áreas de Informática. Os trabalhos podem ser artigos científicos (textos com 30.000 caracteres), ou opiniões (texto com 8.000 caracteres) e remetidos em 3 vias, seguindo o formato dos artigos e opiniões aqui publicados*.

* Os nomes dos autores, endereço e vinculação profissional devem aparecer em folha separada do texto, de modo a possibilitar, sem identificação, um julgamento do trabalho.