ISSN0104-4869

REVISTA DO INSTITUTO DE

# INFORMÁTICA

PUCCAMP - VOLUME 3 - Nº 2 JULHO/DEZEMBRO/1995

## CONTEÚDO/CONTENTS

### Artigos/Articles

**Revista do Instituto de Informática da PUCCAMP. -**

Volume 1, n. 1 (1992) - . Campinas:

Semestral

1. Informática-Periódico

**CDD 001.61**

**CDU 681.3**

# Revista do Instituto de Informática da PUCCAMP

**Publicação Semestral**

**Editor-Executivo:** Prof. Ricardo Pannain



**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. 3 | n. 2 | p. 1-52 | julho/dezembro/1995 |
|---|---|---|---|---|---|

# EDITORIAL

Após três anos do lançamento do primeiro exemplar da Revista do Instituto de Informática, esta se consolida como um meio alternativo para que professores, alunos e pesquisadores apresentem seus trabalhos à comunidade científica. Com um Conselho Editorial composto por nomes de expressão na área de Informática, em âmbito nacional, a Revista se impõe pela seriedade na escolha e avaliação dos artigos.

Todo o esforço teve à frente, até o último número, a Profª Maria Cristina L. F. M. Aranha que, com sua dedicação, soube elevar não só o nome deste veículo, mas também o do Instituto de Informática.

Tendo recebido as funções de Editor-Executivo, a partir deste número, gostaria de ressaltar o importante auxílio recebido da Profª Maria Cristina e do Prof. Carlos Miguel Tobar Toledo, na editoração da revista.

Na edição deste semestre, temos cinco artigos: um na área de engenharia de software, um na área de sistemas distribuídos, um na área de processamento de imagens para aplicações em astronomia e dois na área de projeto conjunto hardware/software.

O primeiro artigo, denominado Geração Automática de Dados de Testes, apresenta várias técnicas e ferramentas para auxílio na geração de dados para teste de sistemas de software. O segundo artigo, denominado Um Ambiente Distribuído para a Integração de Sistemas, discute uma proposta de especificação e implementação de um ambiente distribuído, heterogêneo e cooperativo para suporte à integração de sistemas aplicados à indústria automotiva.

O terceiro artigo, denominado Processamento de Imagens Ópticas em Astronomia: Um universo de Possibilidades, mostra como foi feito um sistema para processamento de imagens aplicada à astronomia. O quarto artigo, denominado Projeto Conjunto Hardware/Software, mostra os conceitos básicos deste tipo de abordagem e o quinto, denominado Prototipagem Rápida de Hardware para um Ambiente de Projeto Conjunto HW-SW, discute alternativas de implementação da parte hardware, para um sistema hardware/software.

Esperando poder manter o nível e a qualidade que a Revista do Instituto de Informática alcançou através das mãos da Profª Maria Cristina, agradeço as colaborações e sugestões presentes e futuras, de todos aqueles que, de uma maneira ou outra, contribuem com a Revista.

**Prof. Ricardo Pannain**
Editor-Executivo

# CONTEÚDO/CONTENTS

## Artigos/Articles

# GERAÇÃO AUTOMÁTICA DE DADOS DE TESTE

# AUTOMATIC TEST DATA GENERATION

**Carlos Miguel TOBAR Toledo***


## ABSTRACT

The process for the elaboration and execution of tests has received, more and more, the importance that it deserves. Because of its nature, this process requires great quantities of effort and time, which result in its placement on a second level. The automation of this process can result in a significant reduction of effort and time requirements. This paper presents a number of techniques and tools which provide support for the generation of test data (some times test cases) and proposes a simplified classification for them.


**KEY WORDS:** Test, Test Data Generator


## RESUMO

O processo de elaboração e execução de testes tem recebido, cada vez mais, a devida importância, porém, devido à sua natureza exige uma grande quantidade de esforço e de tempo, o que acaba resultando na sua colocação em segundo plano. A automatização desse processo pode contribuir significativamente para a redução de esforço e tempo. Este trabalho apresenta um levantamento de técnicas e ferramentas que auxiliam na geração de dados de teste (algumas vezes casos de teste) e propõe uma classificação simplificada para as mesmas.


**PALAVRAS-CHAVE:** Testes, Gerador Automático de Dados de Teste.

## 1. INTRODUÇÃO

A automatização do processo de testes pode levar à redução significativa de custos, pois os custos de testes correspondem a aproximadamente 50% do custo total do desenvolvimento de sistemas [4].

Uma das atividades que pode ser automatizada e corresponde a um dos mais importantes e difíceis aspectos do processo de testes, senão o mais importante e difícil, é a atividade de geração de dados de teste, ou seja, a atividade de identificar valores para as entradas do programa que satisfaçam determinado critério de teste.

A utilização de critérios de teste ocorre devido à impossibilidade, na maioria dos casos, de se realizar teste exaustivo. Idealmente, estabelece que a escolha de um subconjunto de entradas permite encontrar uma grande porção de defeitos e, portanto, é efetivo e permite um ganho de confiança em relação ao programa. Por outro lado, ao mesmo tempo, introduz a questão sobre a sua efetividade em testar.

Para sistemas reais, devido ao seu tamanho e complexidade consideráveis, a geração de um conjunto efetivo de dados de teste pode representar uma tarefa

(*) Coordenador dos Cursos de Especialização do II/PUCCAMP, Professor dos Cursos de Graduação em Análise de Sistemas e Engenharia de Computação do II/PUCCAMP, Mestre em Ciência da Computação pelo DCC-IMECC UNICAMP e Aluno de Doutorado em Engenharia de Computação e Automação Industrial no DCA-FEE UNICAMP. E-mail tobar@dca.fee.unicamp.br.

árdua envolvendo a análise de milhares de combinações, muitas vezes só realizável através da automação. Situação agravada se considerarmos mudanças nas especificações ou sua não corretude. Normalmente, para a realização de testes, assume-se que as especificações não apresentam problemas.

A ferramenta que auxilia na geração de dados de teste, seja completa ou parcialmente, é chamado TDG (Test Data Generator).

Foram pesquisados 6 diferentes artigos que descrevem TDGs. O objetivo desta pesquisa é o estudo de diferentes tipos de TDGs para identificar suas diferenças e propor uma classificação.

Existe uma série de conceitos que é necessária para o apropriado entendimento dos artigos pesquisados. Estes conceitos, via de regra, são discutidos nas primeiras partes dos artigos, e a terminologia envolvida, normalmente, varia de autor para autor.

## ARTIGOS PESQUISADOS E COMENTADOS

Em seguida são apresentados, resumidamente, os artigos pesquisados na forma de comentários.

## 2. AUTOMATED GENERATION OF TESTCASE DATASETS [5]

O TDG está baseado em teste caixa branca, através de cobertura de estrutura de controle, de acordo com algum critério pré-estabelecido. A geração de dados de teste ocorre através da simulação de execução em sentido reverso (backtracking) de cada caminho em consideração:

. a identificação da estrutura de controle do programa, que é usada, por sua vez, para identificar uma coleção de padrões (caminhos), de acordo com um critério prestabelecido, e que permita exercitar o programa de maneira ampla;

. a escolha de valores do domínio de entrada do programa para exercitar o programa da maneira pretendida; este processo envolve a referência específica ao fonte do programa;

. a análise de dados de testes; após a identificação de variados conjuntos de caminhos, estes podem ser associados às especificações funcionais do programa.Cada caminho representa uma instanciação por dados de uma condição formal de verificação do programa.

Concluindo, o trabalho não aborda diversos pontos importantes:

- como são tratados comandos repetitivos (loops) com número constante de iterações?
- como são tratadas condições compostas com operadores and, or e not?
- como são tratados caminhos infactíveis?
- como são tratados elementos de matrizes e registros, além de variáveis dinâmicas (ponteiros)?
- como são identificados os caminhos relativos a um critério? e
- mais fortemente, como se associam caminhos às especificações funcionais (mesmo que formais)?

## 3. TEST PLAN GENERATION USING FORMAL GRAMMARS [1]

O TDG em questão, é mais que um TDG, é um gerador de casos de teste, baseado em teste caixa preta, através de uma gramática formal que representa um autômata finito extendido (fsa). Este TDG é parte de um sistema maior que permite a execução dos casos de teste gerados.

O motivo da maioria do geradores se basearem na estrutura do programa (caixa branca) é a falta de técnicas formais para especificação de comportamento. Existem, no entanto, alguns tipos de sistemas que podem ser especificados através de gramáticas formais.

Dada a especificação do comportamento observável de um sistema, que pode ser modelado por um fsa, a geração de um conjunto de seqüências de testes ocorrerá através da análise da especificação e definição de seqüências de estímulos. Estes estímulos são, posteriormente, usados em uma simulação do automata para a definição dos resultados esperados.

A especificação é feita via uma descrição formal, que é usada como entrada por um Processador de Linguagem de Requisitos (RLP). O RLP é um compilador direcionado por tabelas (que dependem da aplicação e pode ser "customizado") e gera um fsa extendido, que, por sua vez, é usado como entrada para um Gerador de Planos de Teste (TPG). O TPG é responsável pela geração de seqüências de dados de teste e resultados esperados, que são usados como entrada por um Executor Automático de Testes (ATE). O ATE é o responsável pelos resultados dos testes.

O TPG produz um conjunto de scripts de testes executáveis. Cada script corresponde a uma seqüência de estímulos e das respostas esperadas.

Concluindo, o trabalho não aborda um ponto muito importante: como gerar aleatoriamente os scripts, a partir da gramática (autômata)?

Perceber que esta questão está relacionada diretamente à efetividade do teste, ou seja, quais e quantas combinações de entrada, das possíveis, devem ser escolhidas. Algo parecido com cobertura das especificações.

Reparar também que os casos de teste não têm condições de detectar partes implementadas que não são executadas (estão a mais). Para isso deveria ocorrer algum tipo de combinação com teste caixa branca.

## 4. AUTOMATIC GENERATION OF RANDOM SELF-CHECKING TEST CASES [2]

Os TDGs estão baseados em teste caixa preta de aplicações cujas entradas são especificações feitas em uma linguagem formal e geram dados de teste, isto é programas, de forma randômica. Estes programas-teste podem ser executados sobre o sistema que se quer testar e fazem com que o próprio sistema se auto verique, através de comparações parciais, de forma automática.

A motivação para a existência de TDGs Randômicos é a de que, existindo uma quantidade enorme de possibilidades para teste e uma vez definidos os dados de teste, estes apresentam "gaps" em relação à cobertura total das possibilidades. Mais que isso, a tendência é a manutenção desses dados de teste e, conseqüentemente, dos "gaps". Os TDGs Randômicos podem gerar dados de teste que permitem coberturas diferenciadas, cada vez que são executados. Note-se que essa motivação cria outro grave problema quando considera-se a importância dos testes de regressão.

Um TDG Randômico constitui uma ferramenta específica de cada sistema de software a ser testado.

A estratégia de geração randômica pode ser efetiva (não há comprovação), mas é contrária aos testes de regressão, que poderiam ser dispensados desde que a efetividade fosse comprovadamente alta. Novamente algo a haver com cobertura de especificações.

Reparar também que os casos de teste não têm condições de detectar partes implementadas que não são executadas (estão a mais). Para isso deveria ocorrer algum tipo de combinação com teste caixa branca.

## 5. AUTOMATED SOFTWARE TEST DATA GENERATION [4]

Os TDGs estão baseados em teste caixa branca e em execuções consecutivas do programa: para cada caminho (elemento requerido) realizam-se ajustes em um dado de teste gerado (inicialmente) de forma randômica, considerando fluxo de dados (definição e uso), até que o caminho seja percorrido; após cada ajuste acontece a execução parcial do programa.

TDGs orientados a caminhos têm como entrada o programa a ser testado e um critério de teste, a partir dos quais são gerados dados de teste que satisfaçam o critério.

Os TDGs orientados a caminhos realizam as seguintes operações básicas:

- construção do grafo de fluxo de controle do programa,
- seleção de caminhos que identifica o mais próximo conjunto de caminhos minimal que satisfaça o critério, e
- geração de dados de teste para cada caminho selecionado, de maneira a poder exercitá-lo.

Korel [4] apresenta uma abordagem alternativa, onde os dados de teste são desenvolvidos usando valores reais para as variáveis de entrada, denominada abordagem dinâmica, que está baseada em:

- execução real do programa,
- análise dinâmica de fluxo de dados, através de sua monitoração durante a execução do programa, e
- métodos de minimização de funções.

Durante o teste, se um fluxo de execução indesejável é observado em algum ramo, uma função com valores reais é associada ao ramo. Esta função resulta em um valor positivo quando o predicado do ramo é falso e em um valor negativo quando é verdadeiro.

Algoritmos de busca minimizada são usados para automaticamente localizar valores para as variáveis de entrada, que ocasionam o valor negativo para a função. Além disso, análise dinâmica de fluxo de dados é usada para determinar as variáveis de entrada, responsáveis pelo comportamento indesejável do programa, permitindo com isto o aumento de velocidade no processo de busca.

A abordagem dinâmica permite que matrizes e estruturas dinâmicas de dados possam ser manipuladas

adequadamente, pois durante a execução do programa todos os valores das variáveis, incluindo índices e ponteiros, são conhecidos.

Dado um caminho P, o objetivo do TDG é encontrar uma entrada x para o programa tal que P seja percorrido.

Concluindo, um ponto interessante para o qual não são apresentadas informações é a questão da seleção de caminhos, que de forma minimal, satisfaça o critério escolhido para o teste.

Salienta-se que esta proposta trata loops, elementos de matrizes, registros e variáveis dinâmicas.

A questão de condições compostas não é explicitada, mas é reconhecida como tratável.

## 6. CONSTRAINT-BASED AUTOMATIC TEST DATA GENERATION [3]

O TDG está baseado em teste caixa branca, através da produção de dados de teste que se aproximam da adequação relativa, ou seja, proporção de erros semeados em mutantes que são descobertos. O TDG é dito baseado em defeitos ou baseado em restrições.

Técnicas que escolhem dados de teste, para tentar mostrar a presença (ou ausência) de defeitos, são a origem para os TDGs baseados em defeitos, que são necessários devido à dificuldade inerente ao processo de geração de dados de teste.

Assume-se que o programa testado satisfaz suas especificações funcionais. Mais que isso, os operadores de mutação requerem explicitamente que os dados de teste satisfaçam critérios de cobertura de comandos e ramos, de valores extremos, perturbação de domínio e que modelem diretamente vários tipos de defeitos.

Uma das maneiras de gerar automaticamente dados de teste para matar mutantes é por meio do filtro de dados de teste não efetivos, o que pode ser feito através de restrições matemáticas, ou seja, através da solução de problemas algébricos.

As principais partes do TDG são o analisador de caminhos (AC), o gerador de restrições (GR) e satisfazedor de restrições (SR).

Para cada comando do programa original, o AC cria uma expressão/ restrição de caminho, tal que, se o dado de teste alcança o comando, a restrição será verdadeira. Idealmente seria interessante criar restrições para o inverso (se a restrição é satisfeita o comando será executado), porém isto é intratável.

Na prática, a satisfação de uma restrição de caminho garante a execução do comando alvo na ausência de loops.

O GR constrói as restrições de necessidade e de predicados e o SR pega cada restrição de necessidade, encontra o seu respectivo comando e faz a sua conjunção com a restrição de caminho. A solução dessa conjunção permite a geração do dado de teste. Se o dado de teste não pode ser gerado por algum motivo, a informação a respeito do motivo é apresentada ao testador.

Concluindo, tudo indica que os dados de teste gerados através de mutação são usados, posteriormente, para o teste efetivo do software, considerando que estes, por serem adequados ou relativamente adequados, são, potencialmente, capazes de detectar outros erros, que não os das mutações.

Falta à estratégia saber tratar loops e caminhos infactíveis.

Como nada se diz a respeito de como são produzidas as restrições de caminhos, nada se pode concluir quanto ao tratamento de elementos de matrizes, registros e variáveis dinâmicas.

A estratégia foi experimentalmente analisada quanto à sua efetividade e os dados parecem indicar que, apesar de uma escolha aleatória de valores em sub-domínios restritos, a adequação é superior a 90%.

## 7. AUTOMATICALLY GENERATING TEST DATA FROM A BOOLEAN SPECIFICATION [6]

O TDG está baseado em teste caixa branca, através da estratégia de impacto significativo de cada variável em uma especificação em formato de expressão booleana. A geração de casos de teste se faz através de um algoritmo que parcialmente soluciona a satisfatibilidade da expressão, mas também a sua insatisfatibilidade.

A base dos algoritmos é uma especificação em formato de expressão booleana, a qual é analisada para que cada uma de suas componentes possa impactar no resultado final, seja ele verdadeiro ou falso. Esta estratégia é não determinística.

A especificação é transformada na forma canônica disjuntiva (DNF canônica), ou seja, soma de produtos, onde cada produto contém uma instância de cada uma das variáveis participantes, seja negada ou não.

Um caso de teste é uma atribuição de valores verdadeiros ou falsos às variáveis da expressão (fórmula).

Uma instância de uma variável em um dos produtos é chamada de literal e tem um impacto significativo, em um determinado caso de teste, se todos os valores dos outros literais sendo os mesmos, o valor final da expressão depende do valor dessa variável.

A estratégia em questão envolve a seleção de casos de teste que demonstram o impacto significativo de cada variável nos possíveis valores da fórmula.

Um teste deve envolver pontos verdadeiros e pontos negativos, estes são chamados pontos se considerarmos o espaço das possíveis combinações de valores das variáveis da fórmula. Um ponto verdadeiro ocasiona a fórmula ser avaliada para verdadeiro e um ponto falso para falso.

A estratégia básica garante a detecção de alguns tipos de erros de implementação pois os pontos verdadeiros únicos garantem o resultado verdadeiro e os pontos falsos próximos garantem o falso. Ao mesmo tempo alguns outros tipos de erros podem não ser detectados ou não o serão.

Concluindo, esta estratégia preocupa-se com a efetividade dos casos de teste gerados.

Reparar também que os casos de teste não têm condições de detectar partes implementadas que não são executadas (estão a mais). Para isso deveria ocorrer algum tipo de combinação com teste caixa branca.

## 8. CONCLUSÕES

Segundo Korel [4] existem três diferentes tipos de TDGs:

- orientados a caminhos, com exemplo no próprio trabalho [4];
- de especificação de dados, com exemplo em [5] e [1];
- randômicos, com exemplo em [2].

Artigos apresentando as quatro referências estão resumidos na segunda parte da monografia, juntamente com os trabalhos de [3], apresentado como um TDG baseado em defeitos, e [6], apresentado como baseado em uma especificação lógica.

Notamos que os trabalhos [5] e [1] são bem diferentes, para poderem constituir um mesmo tipo de TDG, como proposto por Korel [4]. Há sim uma semelhança entre os trabalhos de [1] e de [6], pois os mesmos se baseiam em especificações formais.

Pode-se dizer que os trabalhos de [1], [2] e [6] apresentam abordagens caixa preta. Enquanto que [5], [4] e [3] apresentam a abordagem caixa branca.

Os trabalhos de [5] e [3] são semelhantes na medida em a geração de dados de teste é realizada a partir da solução de expressões algébricas. Porém a motivação e a forma de criação das expressões diferem nos dois trabalhos.

Outra semelhança que pode ser vista entre os trabalhos de [3] e [6] é o uso de expressões algébricas lógicas normalizadas, embora ressalte-se que o primeiro usa o programa fonte para produzir as expressões e a segunda usa as especificações.

Os trabalhos de [1] e de [2] baseiam-se em linguagens, através das quais é possível descrever seqüências de entradas para o programa que se quer testar. Mais que isso, ambos provêm, além de dados de teste, resultados esperados e executores automáticos de testes. As diferenças, no entanto, existem na medida em que em [1] o esquema é genérico e adaptável a diversos e diferentes tipos de sistemas, enquanto que em [2] os esquemas são específicos. Isto se deve ao próprio tipo de linguagem usado: em [1] a linguagem descreve um autômata finito e em [2] descreve especificamente uma linguagem de programação computacional de alto nível.

Assim, propomos uma classificação diferente para os TDGs apresentados nos artigos estudados. Esta classificação leva em consideração o tipo de entrada que cada TDG necessita e nada mais é que a classificação a respeito do tipo de teste utilizado como base (caixa preta ou caixa branca), assim temos:

- especificação formal;
- programa fonte do software a ser testado.

O primeiro tipo de TDG permite a geração de casos de teste e pode ter complexidade variada em relação ao tipo de formalismo usado:

- expressões booleanas como em [6] - a mais simples;
- linguagem de transição de estados como em [1] (autômata finito extendido) - complexidade média;
- linguagem genérica de programação como em [2] - a mais complexa.

O segundo tipo de TDG permite apenas a geração de dados de teste, devido à sua natureza de testador caixa branca. Explícita ou implicitamente requer um critério de cobertura:

- [5] e [4] requerem explicitamente o critério, dando flexibilidade ao testador na sua escolha e, indiretamente, na efetividade do resultado (considerando que cada tipo de cobertura tenha efetividade distinta, como proposto por Weyuker);
- [3] usa um critério de cobertura implícito, procura-se satisfazer o critério de morte de todos os mutantes não equivalentes ao programa original.

Ressalte-se que o trabalho de [4] recebe como entrada um critério de cobertura, que pode não ser baseado em fluxo de dados, porém utiliza fluxo de dados para melhorar o desempenho do processo de geração de dados de teste.

Existe um TDG caixa branca que se baseia em defeitos (mutação), porém não fica claro se os outros permitem a escolha de critérios de cobertura baseados em fluxo de dados ou, se apenas, fluxo de controle.

Os TDGs caixa branca orientam a geração do dado de teste em relação a cada elemento requerido pelo critério de cobertura, de forma a criar um predicado específico do elemento requerido.

Os TDGs estudados na sua maioria utilizam-se do recurso de simulação, quais sejam: os caixa branca ([5], [4] e [3] e os caixa preta complexos ([2]).

Todos os TDGs utilizam a geração randômica em graus variados:

- escolha sujeita a restrições como em [5];
- escolha livre, posteriormente adaptada, como em [4];
- escolha segundo um padrão de produção como em [1] e [2];
- escolha em um sub-domínio reduzido, devido à estratégia usada, como em [3] e [6].

Geração de dados de teste através de uma abordagem caixa branca representa o risco de: não se detectar a falta de funcionalidades e de apenas exercitar o software, sem validá-lo.

A utilização de TDGs, como de qualquer outra ferramenta automatizada, no processo de teste constitui mais uma variável a ser considerada na avaliação e depuração de resultados, diretamente relacionada com a corretude da própria ferramenta.

Alerta-se que os dados (ou casos) de teste obtidos a partir de TDGs não são suficientes para o teste. Há necessidade, de pelo menos, teste de unidade e de integração. Outros testes podem vir a ser necessários, caso se detecte pedaços não testados, para isso é necessário algum tipo de instrumentação para verificar a efetividade dos dados de teste gerados.

Existe ainda a necessidade de muita pesquisa e experimentação para a obtenção de TDGs genéricos ou que possam atender a qualquer classe de programas, todavia nota-se que para alguns tipos de programas a automatização do processo de testes é uma realidade.

## REFERÊNCIAS BIBLIOGRÁFICAS

[1] J. A. Bauer & A. B. Finger Test Plan Generation using Formal Grammars Proc. 4th. Int. Conf. on Soft. Eng'g, sep/79, pp. 425-432.

[2] D. L. Bird & C. U. MunoZ Automatic Generation of Random Self-Checking Test Cases IBM Sys. Journal, 22 (3), 1983, pp. 229-245.

[3] R. A. DeMillo & A. J. Offutt Constraint-Based Automatic Test Data Generation IEEE ToSE, 17 (9), sep/91, pp. 900-910.

[4] B. Korel Automated Software Test Data Generation IEEE ToSE, 16 (8), aug/90, pp. 870-879.

[5] E. F. Miller & R. A. Munoz Automated Generation of Testcase Datasets SIGPLAN Notices, 10 (6), jun/75, pp. 51-58.

[6] E. Weyuker, T. Goradia & A. Singh Automatically Generating Test Data from a Boolean Specification IEEE ToSE, 20 (5), may/94, pp. 353-363.

# AutoSTEP
# UM AMBIENTE DISTRIBUÍDO PARA A INTEGRAÇÃO DE SISTEMAS

# AutoSTEP
# A DISTRIBUTED ENVIRONMENT FOR SYSTEM INTEGRATION

**Eliane Gomes GUIMARÃES***
**Rogério Almeida BARRA****

## ABSTRACT

This paper presents a proposal for the design and implementation of a distributed,heterogeneous and cooperative environment for supporting the integration of systems in the scope of the automotive industry. This task will demand an open architecture where heterogeneous systems can interact dynamically. The needs for information sharing and interoperability between software systems encourage the introduction of standards like STEP ("Standard for the Exchange of Product Model Data") and CORBA ("Common Object Request Broker Architecture"). AutoSTEP is an open distributed environment being developed at CTI where the CORBA architecture is used as infrastructure for integrating cooperative applications, mainly those related to the STEP standard.

**KEY WORDS**: Open Distributed Systems, Cooperative Applications, STEP, CORBA Architecture.

## RESUMO

Este artigo apresenta uma proposta de especificação e implementação de um ambiente distribuído, heterogêneo e cooperativo para o suporte à integração de sistemas no âmbito da indústria automobilística. Isto requer uma arquitetura aberta onde sistemas heterogêneos interajam dinamicamente. A necessidade de compartilhamento de informações e interoperabilidade entre os sistemas de software motiva o surgimento de padrões, tais como STEP ("Standard for the Exchange of Product Model Data") e CORBA ("Common Object Request Broker Architecture"). AutoSTEP é um ambiente distribuído aberto em desenvolvimento no CTI onde a arquitetura CORBA é utilizada como infra-estrutura de integração de aplicações cooperativas, notadamente aquelas relacionadas com o padrão STEP.

**PALAVRAS-CHAVE**: Sistemas Distribuídos Abertos, Aplicações Cooperativas, Padrões STEP, Arquitetura CORBA.

## 1. INTRODUÇÃO

A necessidade do compartilhamento de informação e outros recursos motivou o surgimento de sistemas computacionais de natureza distribuída. Em tais sistemas um conjunto de componentes de hardware e software

(*) Pesquisadora da Fundação Centro Tecnológico para Informática (CTI).
(**) Pesquisador da Fundação Centro Tecnológico para Informática (CTI) - Professor do Programa de Mestrado em Informática da PUCCAMP.

interagem de forma harmoniosa e cooperativa. Via de regra, os sistemas distribuídos (SDs) escondem do usuário os detalhes inerentes a distribuição do processamento, tais como localização geográfica dos componentes, replicação de dados e funcionalidades, e ocorrência de falhas. Esta propriedade é denominada transparência e constitui um ponto crítico no desenvolvimento de sistemas distribuídos.

Historicamente, os primeiros SD's utilizavam a troca de mensagens como elo de comunicação entre os seus componentes. Este mecanismo emprega apenas os recursos oferecidos pela interface de programação dos protocolos de transporte (por exemplo, a interface de sockets comumente associada à pilha de protocolos TCP/IP).

Um marco importante no desenvolvimento de SD's foi a extensão da programação estruturada para ambientes distribuídos, através do modelo cliente/servidor. Neste modelo os componentes assumem papéis de clientes (requisitantes de serviços) ou servidores (provedores de serviço). Assim sendo, a aplicação é estruturada segundo serviços (implementados em procedimentos) que servidores colocam à disposição de clientes. Clientes evocam serviços através de um mecanismo de interação denominado Chamada de Procedimento Remoto (RPC: Remonte Procedure Call). Este mecanismo permite que um componente evoque procedimentos definidos fora do seu espaço de endereçamento, passando parâmetros e recebendo resultados, de forma similar a evocação de procedimentos locais. Esta propriedade torna a programação de sistemas distribuídos similar à programação de sistemas centralizados e estruturados.

Da mesma forma que o paradigma da programação estruturada foi extendido para ambientes distribuídos, o mesmo vem ocorrendo com o paradigma de programação orientada a objetos. Nesta extensão, os objetos se localizam em diferentes espaços de endereçamento e são capazes de conduzir computações em paralelo e de forma autônoma (isto é, os objetos são ativos). Nesta linha, vários modelos de referência e arquiteturas estão sendo propostos com o intuito de facilitar a implementação de objetos em ambientes distribuídos. Cita-se aqui o modelo ODP ("Open Distributed Processing") proposto pela ISO ("International Organization for Standardization") e as arquiteturas CORBA ("Common Object Request Broker Architecture") da OMG ("Object Management Group"); DSOM ("Distributed System Object Model") da IBM Corporation e COM ("Component Object Model") da Microsoft Corporation.

A maior motivação no emprego destes padrões é permitir que produtos complexos sejam integrados aos sistemas distribuídos sem a necessidade de adaptações (que inevitavelmente se tornam focos de problemas). Exemplo de tais produtos são: bases de dados, sistemas gráficos de visualização e pacotes de CAD ("Computer-Aided Design"). Para interoperarem, tais produtos necessitam aderir a um ou mais padrões de interoperabilidade. Neste aspecto, a arquitetura CORBA vem despertando grande interesse como elemento de interoperabilidade entre aplicações, sendo que produtos para o desenvolvimento de aplicações aderentes ao CORBA já estão disponíveis comercialmente.

A arquitetura CORBA [4] define uma arquitetura orientada a objeto que permite às aplicações interagirem sem qualquer conhecimento da infra-estrutura de comunicação (redes, sistemas operacionais, etc). Objetos definem suas interfaces através de uma linguagem comum e, com base nestas definições, um sistema de runtime denominado Agente de Requisições de Objetos ("Object Request Broker - ORB") intermedia a comunicação entre estes objetos. Esta intermediação inclui a localização e instanciação de servidores, a passagem dos parâmetros estipulados na interface e a coleta do resultado da evocação. A interação via ORB provê um nível de abstração maior na comunicação quando comparado aos mecanismos baseados em passagem de mensagem ou RPC.

Aplicações de interesse neste trabalho são da classe CIME ("Computer Integrated Manufacturing and Engineering"). Tais aplicações tipicamente requerem a habilidade para compartilhar e trocar dados em um ambiente distribuído, aberto, heterogêneo e cooperativo. Neste sentido, modelos de informação como proposto pelo padrão STEP ("Standard for the Exchange of Product Model Data"), de interoperabilidade como proposto pelos padrões do OMG e de acesso a informação como proposto pelo ODMG ("Object Data Management Group") devem ser considerados. Este trabalho examina algumas propostas de ambientes para desenvolvimento de aplicações CIME e introduz o AutoSTEP, um protótipo de ambiente computacional com esta finalidade.

O projeto AutoSTEP tem por objetivo especificar e implementar o protótipo de um ambiente computacional que integre aplicações no âmbito da indústria automobilística permitindo o compartilhamento de dados entre as mesmas. Além disto, o ambiente deve suportar a coordenação entre os agentes ao longo do ciclo de vida de um automóvel.

A Seção 2 deste artigo discute aspectos do problema que o projeto procura resolver. Na Seção seguinte, os

padrões relevantes para o AutoSTEP são introduzidos. Os padrões STEP e CORBA são caracterizados e os seus principais componentes são apresentados. A Seção 4 caracteriza o modelo de informação em que o projeto é baseado e discute sua relevância quanto à prática industrial no setor automobilístico. Uma vez que a arquitetura de sistema, metodologias e ferramentas computacionais a serem adotadas no projeto AutoSTEP terão como referência esforços anteriores, estes são apresentados na Seção 5. Em especial, será considerado o projeto CoConut ("Computer Support for Concurrent Design Using STEP") [5]. Outros esforços relevantes incluem o projeto PISA ("Platform for Integration of CIME Applications") [6] e a abordagem orientada a objetos para a implementação de sistemas de visualização baseados em STEP proposta em [1]. A Seção 6 caracteriza áreas em que o projeto AutoSTEP trará contribuições a estes esforços.

## 2. CARACTERIZAÇÃO DO PROBLEMA

Tecnologia de informação tem sido empregada pelas empresas do setor automobilístico em diversas funções do sistema produtivo, abrangendo, entre outros, sistemas CAD, CAE, CAM e aplicações para gerenciamento de configuração de produtos. Tais sistemas geram e utilizam dados de produtos em forma digital. Tipicamente, a comunicação de dados de produtos entre funções se dá via papel ou interfaces específicas. Tais interfaces se fazem necessárias, uma vez que aplicações requerem dados de produtos em diferentes formatos. A incompatibilidade de formatos conduz a uma redução dos benefícios advindos da introdução de tecnologia de informação devido a barreiras de comunicação entre aplicações.

Um dos principais fatores que contribuem para a situação descrita acima é o fato dos sistemas serem desenvolvidos de forma isolada, impossibilitando o compartilhamento de dados entre as aplicações. É importante salientar os custos associados a estas barreiras de comunicação entre os sistemas, os quais incluem:

- reentrada de dados no sistema produtivo;
- erros na transferência de dados entre sistemas;
- necessidade de compatibilização de dados presentes em mais de um lugar;
- dificuldade de acesso a informações requeridas por agentes do sistema produtivo.

Uma questão fundamental relativa a estes custos é que eles não agregam valor ao produto.

A principal característica de tal ambiente computacional é a ausência de um modelo de informação que capture a semântica comum às várias aplicações executando tarefas no sistema produtivo. Devido à inexistência deste modelo, vários fatores contribuem para a ineficiência global do sistema produtivo, entre os quais podem ser citados:

- a conversão de dados consome muitos recursos;
- a necessidade de conversão implica em operação seqüencial do sistema produtivo, impedindo atividades concorrentes;
- a necessidade de desenvolvimento de interfaces implica em resposta ineficiente a mudanças nos sistemas computacionais, requeridas por novas situações de negócios;
- recursos humanos são dispendidos para a localização e compatibilização de dados de produtos;
- a necessidade de conversão implica em redução da qualidade dos dados, conduzindo a decisões de qualidade inferior.

Em síntese, do ponto de vista dos objetivos estratégicos de uma empresa, a ineficiência de comunicação de dados de produtos entre sistemas resulta em:

- maiores custos;
- menor qualidade dos produtos;
- maior prazo para introdução de novos produtos no mercado;
- dificuldade de adaptação do sistema produtivo a novas situações de negócios;
- sistemas computacionais que provêm suporte deficiente aos negócios das empresas.

### 2.1 TECNOLOGIAS PARA DADOS DE PRODUTOS

A busca de soluções para as questões mencionadas no item anterior, exige uma análise sistemática da aplicação de tecnologia de informação a todos os aspectos concernentes ao desenvolvimento, manufatura e operação de produtos. Tecnologia para dados de produtos (TDP), como um conjunto de técnicas, métodos e ferramentas visando a efetiva comunicação de dados de produtos no âmbito de uma empresa e entre esta e seus fornecedores, surge como resposta a esta necessidade. Em última análise, TDP objetiva prover informação

correta, no lugar correto e em forma correta, no instante em que esta se faça necessária.

Conforme ilustrado na Figura 1, o componente central em TDP é um modelo de produto integrado, de tal forma que aplicações enxergam umas às outras através deste modelo, ou seja, enquanto o modelo descreve todas as informações ao longo do ciclo de vida do produto, diferentes aplicações representam visões particulares das informações sendo integradas.

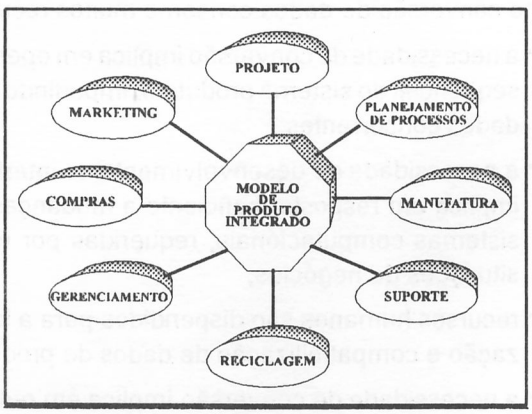


**Figura 1** - Modelo de produto em um sistema produtivo

Uma importante característica do problema que compõe TDP é a heterogeneidade do sistema produtivo, tanto no que diz respeito a hardware quanto a software. Esta heterogeneidade é causada pela necessidade de otimização das funções suportadas por aplicações computacionais, o que implica na escolha de um sistema segundo critérios econômicos e funcionais, de forma a atender os processos de negócios de uma empresa. Além disto, a crescente interação entre clientes e fornecedores, no âmbito da indústria automobilística, requer aplicações aptas, a priori, a se comunicarem entre si.

A abordagem mais efetiva, em termos de custos e tempo requerido para a integração de sistemas, consiste na utilização de padrões. A Seção seguinte descreve STEP e CORBA, componentes centrais da arquitetura aberta proposta neste projeto.

## 3. PADRÕES RELEVANTES AO PROJETO AutoSTEP

### 3.1 PADRÃO STEP

A necessidade de trocar dados de produto em forma digital entre diferentes sistemas computacionais que suportam as tarefas executadas ao longo do ciclo de vida de produtos, particularmente sistemas CAD, foi identificada no final dos anos setenta. Em conseqüência, vários padrões para troca de dados de produtos, tais como, IGES ("Initial Graphics Exchange Specification" - EUA), SET ("Standard d'Echange et de Transfert" - França) e VDA-FS ("Verband der Deutschen Automobilindustrie - Flachenschnittstelle" - Alemanha), foram desenvolvidos durante os anos oitenta.

Deficiências identificadas através da utilização destes padrões, tais como as ambigüidades de suas definições, as restrições referentes ao escopo de dados de produtos representados e a falta de requisitos formais para verificação de conformidade, conduziram à formação, em dezembro de 1983, de um comitê no âmbito da ISO ("ISO 10303 - Technical Committee 184 - Subcommittee 4") com o objetivo de especificar um padrão internacional para modelos de dados de produtos. Os objetivos deste esforço, informalmente conhecido como STEP, podem ser sintetizados da seguinte forma:

- prover a descrição completa, não-ambígüa e processável por computador das características físicas e funcionais de produtos ao longo de todo o seu ciclo de vida;
- prover mecanismos que possibilitem troca de dados de produtos e compartilhamento destes entre funções de um sistema produtivo.

Uma discussão mais detalhada dos requisitos impostos a STEP e da metodologia utilizada em sua especificação pode ser encontrada em [2] e [3]. Cabe, no entanto, salientar que enquanto a especificação do padrão envolve várias questões em aberto, do ponto de vista de pesquisa e desenvolvimento, seu elemento catalizador tem sido a demanda industrial por efetiva comunicação entre aplicações computacionais concernentes ao ciclo de projeto, manufatura e operação de produtos. Neste sentido, STEP conta com a participação de mais de 500 especialistas de dezenas de empresas líderes em seus ramos de atuação.

### FORMAS DE IMPLEMENTAÇÃO DO PADRÃO

Entre possíveis métodos para implementação de STEP, podem ser citados:

- **Troca de Arquivo:** nesta abordagem, a sintaxe e estrutura de um arquivo físico usado para armazenar ou transferir dados de produtos são especificadas. A codificação desta estrutura em texto legível é definida pela parte 21 de STEP [10].

- **Interface para Programação de Aplicações:** neste método de implementação, o programador de aplicações tem à sua disposição uma interface para programação, que dá a ele a ilusão, que os dados sendo manipulados, são representados no sistema responsável por seu armazenamento na forma como eles são descritos no modelo de informação.

  SDAI ("Standard Data Access Interface") [11] provê um mecanismo de acesso a dados cuja estrutura é definida usando uma linguagem denominada EXPRESS. Uma implementação SDAI envolve a representação da especificação da interface em uma linguagem de programação, por exemplo C ou C++. Utilizando uma implementação SDAI, programas de aplicação podem ser desenvolvidos independentemente da tecnologia empregada para armazenamento dos dados.

- **Bancos de Dados:** desde que um modelo de informação, relativo a uma aplicação, é especificado usando uma linguagem para modelagem de dados em alto nível, EXPRESS, ele pode ser mapeado para o esquema lógico de diferentes tipos de bancos de dados, tais como relacionais e orientados a objetos. Ferramentas para o mapeamento de modelos EXPRESS em SQL já se encontram disponíveis.

### 3.2 O PADRÃO CORBA

CORBA [4] é baseado no modelo de objeto OMG, que considera, tanto a informação quanto as aplicações, como objetos que implementam um conjunto bem definido de operações. As operações definem o conjunto de possíveis manipulações de informação, ao passo que, para as aplicações, as operações definem o comportamento que caracteriza as aplicações. Em ambos os casos, o algoritmo que implementa uma operação é considerado um método, que por sua vez é implementado por uma ou mais rotinas. Isto permite que se agreguem múltiplas implementações numa única definição de método. Métodos e suas implementações são invocados através do envio de mensagens ao objeto definindo o método.

A Figura 2 mostra os quatro elementos principais da arquitetura proposta pelo OMG. O OMG fornece um barramento comum de interconexão que conterá componentes clientes, serviços básicos necessários para esses componentes e recursos comuns para a colaboração entre componentes.

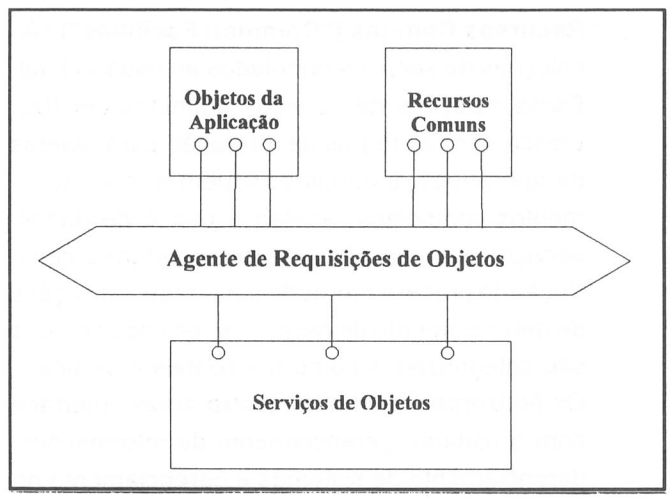


**Figura 2** - Arquitetura proposta pelo OMG

Os quatro elementos da arquitetura do OMG são:

- **Agente de Requisições de Objetos - ORB** ("Object Request Broker") é o meio que estabelece relações cliente/servidor entre objetos. É o mecanismo que permite que um objeto cliente possa invocar de forma transparente um método em um objeto servidor (local ou remoto). O ORB intercepta a chamada e procura um objeto que possa atender o requisito, fornecendo os parâmetros, invocando o seu método e retornando os resultados. Os clientes ignoram os mecanismos utilizados pelo ORB para comunicar, ativar ou armazenar objetos servidores, bem como quaisquer outros aspectos do sistema que não são parte de uma interface do objeto. O CORBA 1.1 definiu uma linguagem genérica de definição de interface IDL ("Interface Definition Language"), utilizada para definir as interfaces dos objetos que se comunicam dentro de uma implementação específica de ORB. O CORBA 2.0 especifica como os ORBs de diferentes fornecedores podem interoperar.
- **Serviços de Objetos ("Object Services")** são coleções de serviços com interfaces de objeto especificadas em IDL e proporcionam funções básicas para uso e implementação de objetos. Esses serviços ampliam as capacidades dos ORBs e incluem serviços de nomes, notificação de eventos, persistência de objetos, gerenciamento de

ciclo de vida, transações, controle de concorrência e relacionamentos. Alguns serviços estarão disponíveis em breve: consultas, licenciamento, propriedades e segurança.

- **Recursos Comuns ("Common Facilities")** são coleções de serviços orientados ao usuário final. Essas coleções de serviços definidos em IDL, prescrevem as regras de interação para objetos de aplicativos. Exemplos incluem e-mail, documentos compostos, acesso a banco de dados, serviços que especificamente suportam a construção de processadores de palavras e aplicações de gerenciamento de redes. Os recursos comuns são categorizados como horizontais e verticais. Os horizontais tratam de quatro áreas: interface com o usuário, gerenciamento de informações, gerenciamento de sistemas e gerenciamento de tarefas. Os verticais fornecem interfaces definidas em IDL para suportar a interação de coleções de objetos especializados para a saúde, varejo, finanças e outros domínios. A linha divisória entre recursos comuns e serviços de objetos é imprecisa. Os serviços de objetos precisam ser implementados em todo ORB, enquanto que os recursos comuns são opcionais.
- **Objetos da Aplicação ("Application Objects")** são objetos específicos para aplicações do usuário final. Estes objetos necessitam ser definidos em IDL a fim de participar em interações mediadas pelo ORB. Interfaces IDL são compiladas resultando na geração de STUBs: elementos utilizados na manipulação de dados entre os objetos e o ORB. Os objetos das aplicações, naturalmente, expandem os serviços proporcionados pelo ORB, pelos recursos comuns e pelos serviços de objetos. Uma aplicação é tipicamente construída de um número grande de classes de objetos básicos; alguns destes podem ser proporcionados pelos serviços comuns do OMG.

A especificacão CORBA mais recente (2.0) especifica alguns protocolos de transporte (como o TCP/IP) para permitir a interoperabilidade entre ORBs de fornecedores distintos. A solução, no momento, para este problema de comunicação é a seleção de uma única implementação que suporte todas as plataformas do sistema, ou o uso de objetos gateways, como recomendado na especificação CORBA.

## 4. UM MODELO DE INFORMAÇÃO PARA A INDÚSTRIA AUTOMOBILÍSTICA

Como parte de STEP, vem sendo definido um modelo de informação que suporta diferentes fases do processo de desenvolvimento de automóveis. Este modelo, especificado no protocolo de aplicação 214 ("Core Data for Automotive Mechanical Design Processes"), descreve informações relativas aos seguintes processos [15]: definição do produto, estilo, projeto, avaliação, planejamento da produção, projeto e construção do maquinário, e controle de qualidade.

Os seguintes aspectos de automóveis são capturados pelo modelo de informação sendo especificado pelo protocolo de aplicação 214:

- peças, montagens de peças, ferramentas e montagens de ferramentas produzidas por fabricantes de automóveis e seus fornecedores;
- dados de definição de produtos e dados para controle de configuração relativos à fase de projeto de um automóvel;
- dados de modificações de projetos e dados relativos à documentação do processo de modificação;
- vários tipos de representação geométrica da forma de peças e ou ferramentas;
- dados relativos à apresentação visual de produtos;
- representação de peças e ou ferramentas por "Form Features";
- propriedades de materiais utilizados nos produtos;
- dados que descrevem condições superficiais e tolerâncias;
- dados para análise cinemática;
- dados para análise por elementos finitos;
- dados relativos à liberação e aprovação de produtos e versões de produtos;
- dados que registram as modificações de versões de um produto.

O comprometimento da indústria automobilística mundial com este esforço tem sido inequivocamente demonstrado. Os grandes fabricantes europeus e norte-americanos assinaram um documento com o intuito de caracterizar o STEP como o padrão a ser utilizado para a representação e troca de dados de produtos no âmbito da indústria automobilística.

## 5. PROJETOS CORRELATOS

Nesta seção são discutidos projetos relevantes para os propósitos de AutoSTEP. Enquanto a arquitetura a ser especificada para o ambiente AutoSTEP terá como referência estes esforços, principalmente o projeto CoConut, na Seção subseqüente serão evidenciadas áreas em que AutoSTEP trará contribuições.

### 5.1 0 PROJETO CoConut

CoConut ("Computer Support for Concurrent Design Using STEP") [5] é um projeto sendo empreendido no Instituto Fraunhofer-IGD ("Institut für Graphische Datenverarbeitung"), em Darmstadt, Alemanha. CoConut provê um ambiente aberto para integração de aplicações envolvidas na fase de projeto de produtos. Permitindo efetiva comunicação entre agentes em um ambiente heterogêneo e provendo mecanismos para coordenação de suas atividades, o ambiente CoConut objetiva a redução do tempo e custos relativos à fase de projeto de um produto.

Um dos aspectos relevantes da arquitetura proposta em CoConut é suportar a integração de aplicações previamente existentes, como, por exemplo, sistemas CAD disponíveis comercialmente. Além disto, o ambiente inclui aplicações que suportam a cooperação entre agentes, utilizando técnicas CSCW ("Computer Supported Cooperative Work").

Em CoConut, a integração de dados de produtos é provida por um sistema gerenciador de dados distribuído, cujo modelo conceitual é especificado em conformidade com o protocolo de aplicação 203 ("Configuration Controlled Design") [14] do padrão STEP. Este protocolo suporta, além da descrição da forma do produto (geometria e topologia), a caracterização de sua estrutura e o controle de configuração. Em termos de implementação, um sistema gerenciador de bancos de dados orientado a objetos provê a funcionalidade requerida para gerenciamento de dados distribuídos em um ambiente heterogêneo, incluindo controle de concorrência e processamento de transações.

A Figura 3 apresenta a arquitetura para integração de um sistema CAD com o ambiente CoConut. Linhas contínuas representam transferência de dados, enquanto linhas tracejadas representam comunicação, permitindo troca de mensagens entre aplicações. Uma das aplicações suportadas pelo ambiente é um sistema de visualização cooperativo para modelos de produtos em conformidade com o protocolo de aplicação 203. A extensão desta aplicação com a finalidade de suportar modelos de produtos definidos pelo protocolo de aplicação 214 é uma das áreas em que o projeto AutoSTEP trará contribuições ao ambiente CoConut.

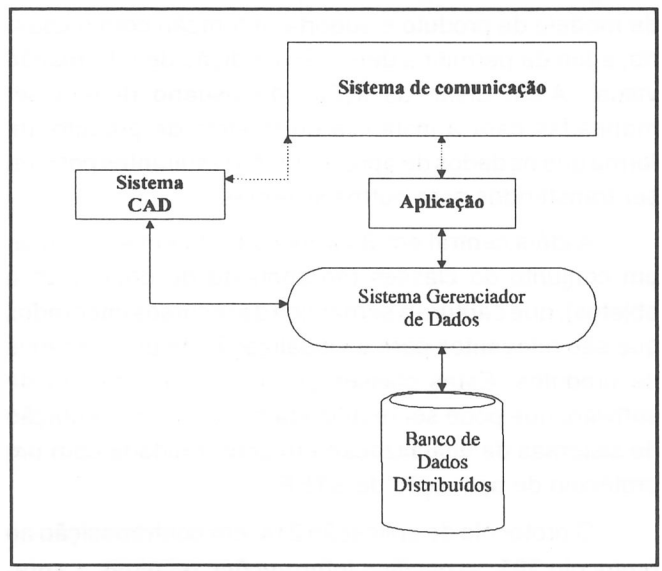


**Figura 3** - Arquitetura para integração de sistema CAD em CoConut [5]

### 5.2 O PROJETO PISA

PISA ("Platform for Information Sharing by CIME Applications") [6] é um projeto sendo empreendido por uma conjunto de empresas e instituições de pesquisa européias, como parte do projeto ESPRIT. O objetivo do projeto PISA é contribuir para o estabelecimento de uma infra-estrutura para o compartilhamento e troca de informações de produto e processo entre aplicações CIME ("Computer Integrated Manufacturing and Engineering"). Para alcançar este objetivo, o projeto PISA considera padrões emergentes, tais como, STEP, CORBA e ODMG na área de bancos de dados.

### 5.3 A ESTRUTURA DE SOFTWARE DO SVwork

SVwork é uma estrutura de software que explora o paradigma de orientação a objetos, com o objetivo de facilitar a implementação de sistemas de visualização em conformidade com STEP [1].

O modelo de apresentação em STEP, como parte dos recursos integrados, provê informações para a especificação de aspectos visuais de propriedades de

produtos, tais como forma e tolerâncias [13]. Dados de apresentação, em conjunção com dados que descrevem propriedades do produto, possibilitam a geração de imagens em conformidade com STEP.

Sistemas de visualização baseados em STEP devem gerar uma imagem correspondente a uma instância de modelo de produto e suportar interação com o usuário, a fim de permitir a definição e edição de informação visual. Além disto, as ações do usuário devem ser mapeadas para a instância do modelo de produto, de forma que os dados de apresentação resultantes possam ser transferidos para outros sistemas.

A idéia central em SVwork consiste em especificar um conjunto de classes (no contexto de orientação a objetos), que capture a semântica de recursos integrados que são relevantes para a visualização de propriedades de produtos. Estas classes provêm uma estrutura de software que pode ser reutilizada para a implementação de sistemas de visualização em conformidade com um protocolo de aplicação de STEP.

O protocolo de aplicação 214, em contraposição ao protocolo 203, especifica informações relativas a apresentação de produtos, através da interpretação de recursos integrados que constituem o modelo de apresentação em STEP. Sendo assim, para a implementação de sistemas de visualização baseados no protocolo 214, mostra-se oportuna a utilização do conjunto de classes definido em SVwork.

## 6. O PROJETO AutoSTEP

O CORBA é utilizado no projeto AutoSTEP para proporcionar a infra-estrutura básica necessária para a integração de sistemas distribuídos no âmbito da indústria automobilística. Um dos aspectos relevantes para alcançar este objetivo é a utilização de Serviços de Objetos ("Object Services") e Recursos Comuns ("Common Facilities").

Algumas propostas para a implementação de classes diferentes de Serviços de Objetos estão sendo estudadas para ampliar as capacidades do Agente de Requisicões de Objetos (ORB).

Os Recursos Comuns terão um papel fundamental no projeto de novos sistemas CIM ("Computer Integrated Manufacturing"), mais ágeis (rápida resposta a mudanças), distribuídos e reusáveis. Os Recursos Comuns proporcionarão funcionalidades que podem ser incorporadas através de muitos domínios da aplicação, tais como: manufatura, projeto, análise, simulação e práticas de negócios. Os Recursos Comuns juntamente com os Serviços de Objetos do OMG suportarão implementações modulares, baseadas em padrões que promovem a agilidade de desenvolvimento e modificação de sistemas.

Os trabalhos começaram com a definição de recursos verticais para manufatura [7]. Atualmente, foram identificadas três áreas específicas de especialização: Especialização de Serviços de Dados do Produto ("Product Data Services Specialization"), Especialização de Gerenciamento de Políticas Diversas ("Policy Variable Management Specializations"), Especialização de Gerenciamento de Históricos ("History Management Specializations").

A área de especialização de Serviço de Dados do Produto suporta o compartilhamento de informação em um ambiente distribuído que pode ultrapassar os limites da empresa. A integração de tecnologias de informação intra/inter-organizacional depende da criação de um modelo conceitual baseado em padrões. Entretanto torna-se necessário coordenar a integração e interoperação de padrões em vez de desenvolvê-los de forma isolada. Especificamente, o Serviço de dados do produto deve ser completamente aderente com os padrões STEP e métodos de implementação. Deve ainda utilizar o SDAI ("Standard Data Access Interface") para manter o Serviço independente de tecnologia de armazenagem de dados e para satisfazer os requisitos de compartilhamento de dados de produto, identificados pelos grupos que desenvolvem o SDAI. O Serviço de dados do produto de STEP necessita concordar com um padrão para implementação de tecnologia de dados distribuídos. O OMG proporciona um padrão de consenso internacional para esta tecnologia.

A área de especialização de Gerenciamento de Políticas Diversas suporta a configuração e gerenciamento de variáveis de gerenciamento de sistemas.

A área de especialização de Gerenciamento de Históricos suporta o armazenamento e acesso de dados armazenados em arquivos.

Estes recursos ainda estão em fase de estudos. No caso do projeto AutoSTEP, é relevante para este projeto a área de especialização de Serviços de Dados de Produtos e, como ainda estes serviços não estão disponíveis, propõe-se sua utilização via Adaptadores de Objetos ("Object Adapters") que conectam ferramentas de desenvolvimento STEP ao ORB (Figura 4).

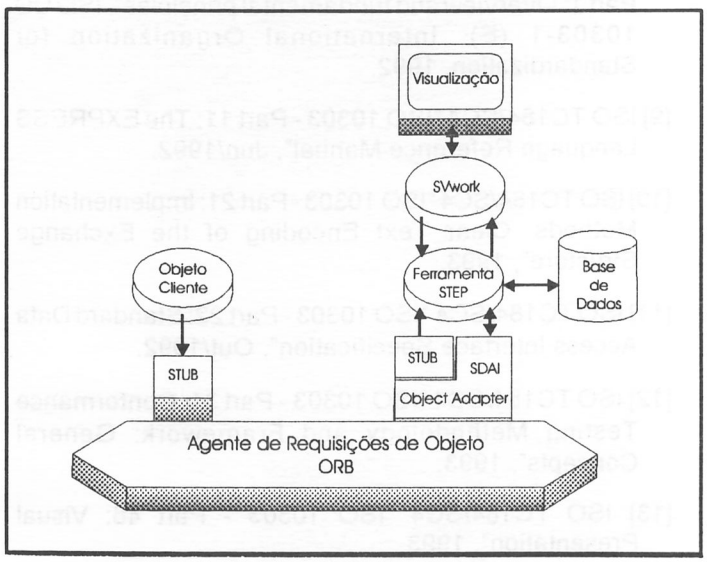


**Figura 4** - Arquitetura do Projeto AutoSTEP

O projeto AutoSTEP deve ser visto como uma extensão aos esforços descritos na seção anterior. Os principais aspectos do projeto são:

- o modelo de produto a ser suportado, especificado no protocolo de aplicação para a indústria automobilística, estende aquele referente ao protocolo suportado no projeto CoConut, abrangendo, além da fase de projetos, outras fases do ciclo de vida do produto;
- a abordagem utilizada no projeto PISA para interoperabilidade entre aplicações, baseada no padrão CORBA, deve ser incorporada caracterizando, com isto, uma arquitetura aberta;
- a utilização da estrutura de software SVwork permitirá a implementação de sistemas de visualização de produtos baseados em STEP mais flexíveis que aquele provido pelo projeto CoConut. O acesso ao SVwork se processará via ORB através de um adaptador de objeto específico;
- o desenvolvimento do SDAI Object Adapter servirá para possibilitar a integração de servidores de sistemas, que podem somente ser acessados via uma interface SDAI (ele traduz as mensagens recebidas do ORB em mensagens semanticamente equivalentes às chamadas SDAI), conforme ilustrado na Figura 4.

Numa primeira etapa o projeto AutoSTEP irá utilizar os seguintes componentes de software disponíveis comercialmente:

- uma implementação da arquitetura CORBA (por exemplo, o ORBIX da Iona Technologies);
- um conjunto de ferramentas de desenvolvimento STEP (por exemplo, o da STEP Tools Inc.). Estas ferramentas propiciam uma interface para programação de aplicações que torna transparente a forma como os dados são armazenados;
- uma base de dados orientada a objetos (por exemplo, o ObjectStore da Object Design) para armazenamento das estruturas de informação prescritas pelo STEP e manipulação destas por programas de aplicação.

## 7. CONCLUSÕES

Este artigo apresenta a motivação e os aspectos mais relevantes dos estudos preliminares para o projeto AutoSTEP. A importância do padrão STEP para o setor automobilístico é caracterizada e a necessidade da harmonização de STEP com o padrão CORBA é examinada.

Além disto, esforços correlatos aos objetivos de AutoSTEP são sumarizados.

Além de permitir a integração de aplicações, tais como sistemas CAD e sistemas para controle de configuração de produtos, o ambiente AutoSTEP deverá oferecer uma plataforma em que aplicações CSCW podem ser desenvolvidas, com o intuito de promover cooperação entre agentes envolvidos no projeto de um automóvel, o que vem de encontro a um dos pressupostos básicos da abordagem para o desenvolvimento de produtos conhecida como engenharia simultânea.

## REFERÊNCIAS BIBLIOGRÁFICAS

[1] Barra, R. A., "On the Implementation of Systems for Realistic Visualization of STEP Models", tese de doutorado, Universidade Técnica de Berlim, Dez/1994.

[2] Barra, R. A. e J. M. Adán, "STEP como um Componente Estratégio para a Integração de Sistemas de Manu-

fatura", Congresso sobre CAE/CAD/CAM e Computação Gráfica, SOBRACON, Abr/1995.

[3] Barra, R. A. e Guimarães, E. G., "AutoSTEP: um Ambiente Integrado para a Indústria Automobilística", III Simpósio Exposição CAE/CAD/CAM, SAE Brasil, Ago/1995.

[4] "The Common Object Request Broker: Architecture and Specification" (Version 1.2 Revision Draft), OMG Document Number 93.12.43, The Object Management Group, Dec/1993.

[5] Jasnoch, U., H. Kress, K. Schroeder e M. Ungerer, "CoConut: Computer Support for Concurrent Design Using STEP", IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Abril/1994, EEE Computer Society Press.

[6] Koethe, M., A. Nieva e F. Schoenefeld, "Product Data Exchange in Open Systems: The PISA Approach", Autofact, EUA, Dez/1993.

[7] "Common Facilities Architecture". OMG Document 95-1-2; Revision 4.0; Jan. 3,1995.

[8] "Product Data Representation and Exchange. STEP Part 1: Overview and fundamental principles", ISO CD 10303-1 (E), International Organization for Standardization, 1992.

[9] ISO TC184/SC4 "ISO 10303 - Part 11: The EXPRESS Language Reference Manual", Jun/1992.

[10] ISO TC184/SC4 "ISO 10303 - Part 21: Implementation Methods: Clear Text Encoding of the Exchange Structure", 1993.

[11] ISO TC184/SC4 "ISO 10303 - Part 22: Standard Data Access Interface Specification", Out/1992.

[12] ISO TC184/SC4 "ISO 10303 - Part 31: Conformance Testing Methodology and Framework: General Concepts", 1993.

[13] ISO TC184/SC4 "ISO 10303 - Part 46: Visual Presentation", 1993.

[14] ISO TC184/SC4 "ISO 10303 - Part 203: Application Protocol: Configuration Controlled Design", 1993.

[15] ISO TC184/SC4 "ISO 10303 - Part 214: Application Protocol: Core Data for Automotive Design Processes", 1994.

# PROCESSAMENTO DE IMAGENS ÓPTICAS EM ASTRONOMIA: UM UNIVERSO DE POSSIBILIDADES

# OPTICAL IMAGE PROCESSING IN ASTRONOMY: A UNIVERSE OF POSSIBILITIES

**Júlio César PENEREIRO***
**Carlos Henrique Dias CARNEIRO****
**Daniel DO VALLE****

## ABSTRACT

Dramatic changes have taken place during the past two decades in the Astronomy because of the new microelectronic technology. When the information in an image is expressed in digital form, it can be manipulated mathematically in order to give more informations about the object. It is described in this paper some detectors in Astronomy and given details about the Charge Coupled Device (CCD) wich are being used in the research project. Furthermore, we present some astrofisical informations about galaxy images, that are being collected through telescopes and being reduced in microcomputers, such as contour maps, surface brightness, major axis profile and structural and morfological parameters.

**KEY WORDS**: Image Processing, Astronomical Detectors, Scientific Divulgation.

## RESUMO

Ocorreram mudanças dramáticas na Astronomia nas últimas duas décadas devido à nova tecnologia microeletrônica. Quando uma imagem é expressa na forma digital, ela pode ser manipulada matematicamente e podemos obter mais informações a respeito do objeto. Descrevemos neste artigo alguns detectores em Astronomia e damos maiores detalhes sobre o Dispositivo de Cargas Acopladas (CCD) utilizado no projeto de pesquisa. Além disso, apresentamos algumas informações astrofísicas de imagens em galáxias, coletadas em telescópios e reduzidas em microcomputadores, tais como mapas de contorno, perfis de brilho superficial no eixo maior e parâmetros estruturais e morfológicos.

**PALAVRAS-CHAVE:** Processamento de Imagens, Detectores Astronômicos, Divulgação Científica.

(*) Professor-Doutor do Departamento de Física do I. C. E. da PUCCAMP.
(*) Observatório Municipal de Campinas "Jean Nicolini" (S. M. Cultura - Pref. Munic. de Campinas).
(*) Astrônomo visitante do Laboratório Nacional de Astrofísica (LNA - CNPq/MCT).
(**) Alunos de Graduação do Curso de Engenharia de Computação do I. I. da PUCCAMP.

## 1 - INTRODUÇÃO

A astronomia observacional se enfrenta com duas grandes preocupações técnicas: a detecção de objetos luminosos de fraco brilho e o aumento da resolução angular com finalidade de separar os mais próximos, ou seja, de vislumbrar as delicadas estruturas presentes no interior dos mais extensos objetos astronômicos. Tradicionalmente, desde que Galileo observou pela primeira vez através do telescópio, o esforço de astrônomos ópticos e engenheiros durante os quase três séculos e meio transcorridos desde então, está sendo dirigido para a construção de telescópios cada vez mais potentes, com maiores diâmetros de suas lentes ou espelhos. Isto é fácil de entender, pois ao aumentar o diâmetro aumenta-se a área coletora de luz e assim a possibilidade de reconhecer a emissão luminosa de objetos fracos. Por outro lado, as leis da difração da luz nos dizem que o poder resolutivo aumenta com o diâmetro da área coletora. No entanto, a tecnologia atual impõem limites de construção a tamanhos máximos de lentes e espelhos de peça única. A forma que foi encontrada para solucionar esta questão está no aparecimento de novos e maiores telescópios baseados em sistemas com espelhos múltiplos.

O problema foi também resolvido parcialmente com a entrada em operação desde 1990 do Telescópio Espacial Hubble que, situado em órbita terrestre e portanto livre da turbulência atmosférica, consegue alcançar as mais fracas magnitudes e as resoluções mais altas conseguidas até o momento.

Ainda que o telescópio e a atmosfera imponham limites físicos, as possibilidades de detecção e resolução está relacionada às suas máximas capacidades quando, ao reconhecer a imagem formada no plano focal do telescópio, colocamos um sistema de detecção capaz de extrair o máximo de informações possíveis ali presentes. O desenvolvimento de novos detectores foi o caminho que os astrônomos, físicos e engenheiros adotaram durante os últimos anos para atacar este velho problema.

Até pouco mais de um século atrás, o único detector empregado para captar os sinais emitidos na banda do visível, para os objetos astronômicos, foi o olho humano que, acoplado a uma ocular, serve para visualizar a imagem dada pela objetiva do telescópico. Ainda que o olho humano possua qualidades excelentes (alta sensibilidade, apreciação de cores, alta capacidade de contraste e de trabalho entre fluxos luminosos extremos), peca na possibilidade de contar os fótons que chegam em instantes diferentes, pois para que registre um detalhe, necessita que os fótons alcancem um mesmo ponto da retina num tempo inferior a 0.1 segundos, que é o tempo de persistência retiniana. Mesmo que a memória humana possa armazenar as características de uma imagem, esta é esquecida, se não toda, ao menos nos detalhes, em pouco tempo, introduzindo com isso uma componente subjetiva à observação. Por outro lado, o olho humano carece da possibilidade de efetuar medidas quantitativas precisas, sendo assim que o seu papel, como detector, tem sido ocupado por novos sistemas mais objetivos.

O primeiro dos detectores desenvolvido e introduzido em Astronomia durante o último século foi a emulsão fotográfica. Ainda que menos sensível que o olho humano, possue a capacidade de acumular os fótons que chegam em intervalos diferentes e em posições espaciais diferentes. Uma placa fotográfica, segundo seu formato, é constituída com grãos de cerca de 20 microns de tamanho que podem armazenar sobre sua superfície entre $10^8$ e $10^9$ pixels[1]. No entanto possue grandes incovenientes: (i) somente trabalha por cima de um certo fluxo de energia mínimo (um grama de sal de prata, para ser reduzido e aparecer no revelador, deve receber uma quantidade mínima de fótons), e inversamente para fluxos energéticos muito grandes pode saturar-se, o que faz com que o sistema seja altamente não linear (isto é, o sinal registrado não é proporcional ao fluxo recebido); (ii) o intervalo dinâmico (veja [2]) é restrito; (iii) possue o incômodo defeito de reciprocidade; e (iv) uma baixa eficiência de detecção. Apesar disso, pelo fato da placa fotográfica ser de simples manejo e relativamente pouco custosa, é ainda utilizada em alguns observatórios e especialmente em programas específicos (como por exemplo os levados a cabo com telescópios do tipo Schmidt, que possuem a vantagem de reconhecer grandes áreas do céu), onde se mostra como um elemento de pesquisa fundamental (veja por exemplo: [16]).

Durante a última metade deste século, a introdução e o desenvolvimento da eletrônica, e ainda o desenvolvimento paralelo de novos instrumentos (grande parte deles para tecnologia militar) deram uma gigantesca contribuição à astronomia observacional, em especial no que se refere à captação de sinais débeis, que agora encontram alta aplicação, não só em Astrofísica que se faz desde o solo, senão também a desenvolvida no espaço por sondas e satélites artificiais.

No que se segue, faremos uma descrição de um detector, dentre vários detectores modernos e empregados em Astronomia, o qual estamos empregando em

(1) Do inglês "picture element":, ou seja, elemento de imagem.

nossas pesquisas que visam o processamento de imagens de alguns objetos astronômicos extensos (galáxias) em microcomputadores da linha IBM-PC, junto ao LNA (Laboratório Nacional de Astrofísica - CNPq/MCT) e ao DF/PUCCAMP (Departamento de Física do I.C.E. da PUCCAMP). Previamente, apresentaremos alguns parâmetros básicos que servirão para quantificar as possibilidades globais da tarefa do detector que estamos utilizando, além de servir de subsídio para as discussões que serão apresentadas.

## 2 - EFICIÊNCIA QUÂNTICA DE DETECÇÃO SENSIBILIDADE ESPECTRAL INTERVALO DINÂMICO RESOLUÇÃO

A **eficiência quântica** (Q) de um sistema que detecta fótons (ou quantuns de luz) se define como a fração de fótons incidentes que em média produzem um sucesso mensurável (por exemplo, um fotoelétron, um grão escurecido na emulsão fotográfica, etc). Assim, podemos escrever que:

$$Q = \frac{N_{\text{Sucessos..Produzidos}}}{N_{\text{Fotons..Incidentes}}} \qquad (1)$$

No entanto, um detector, ainda que ideal, não pode fazer uma medida perfeita. Segundo as leis estatísticas que governam a emissão de fótons, uma medida de um número N de fótons está afetada de uma incerteza $\sqrt{n}$ denominada de "ruído fotônico". Ademais, num detector real outros fatores que contribuem ao ruído vêm limitar a precisão da medida. Este pode ser de três tipos: (i) térmico, que se origina no próprio sistema detector como conseqüência dos elétrons emitidos espontâneamente por agitação térmica da superfície sensível; (ii) de amplificação, produzido pelo sistema amplificador de sinal; (iii) de leitura, produzido pelo sistema de leitura do sinal. É por isso que para caracterizar um sistema real, convém definir um parâmetro mais realista conhecido como eficiência quântica do detector ($Q_D$), mediante a relação:

$$Q_D = \frac{(S/R)^2_{\ Detector..Real}}{(S/R)^2_{\ Detector..Ideal}} \qquad (2)$$

onde, (*S/R*) indica a relação do sinal frente ao ruído.

$Q_D$ é uma medida da eficiência do sistema real frente a um sistema ideal (perfeito) onde conta-se fótons. O critério de otimização de um detector é assim elementar: maximizar a relação (*S/R*) de forma que $Q_D$ adquira o valor mais alto possível. De grande importância são as variações de $Q_D$, frente ao comprimento de onda $\lambda$, que são as que definem a **sensibilidade espectral** do detector. Como exemplos citamos o olho humano e a emulsão fotográfica, para os quais $Q_D \sim$ 1-2% (isto é, de cada cem fótons incidentes apenas um ou dois são registrados), o do fotocátodo, onde é $Q_D \sim$ 10-20% no visível e, do sistema de diodos de silício (CCD), onde $Q_D \sim$ 80% próximo aos 7000 Å (1Å = $1.0 \times 10^{-10}$ m). A Figura 1 ilustra as diferenças entre estes detectores.

Um outro fator que joga um papel importante, quando se trata de detectar um detalhe fraco próximo a outro brilhante, é o denominado **intervalo dinâmico**, que se define como a diferença entre o número mínimo e o máximo de sucessos que pode registrar o sistema.

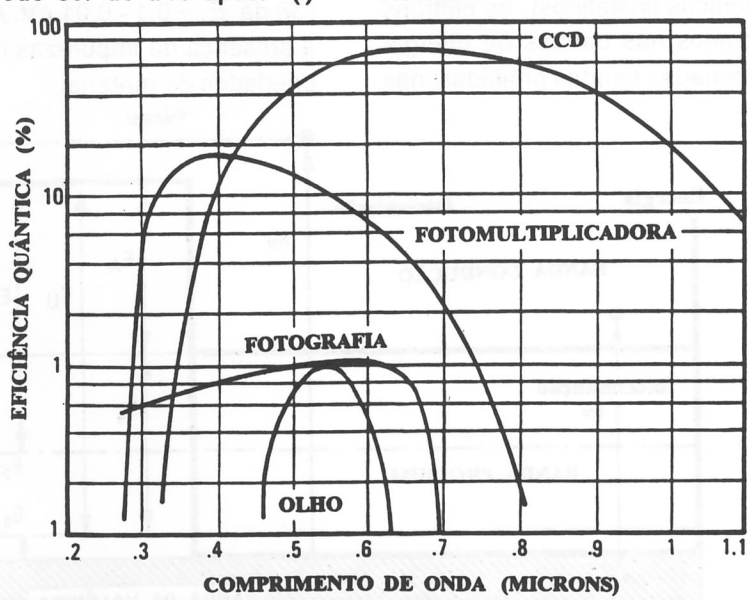


**Figura 1** - Comparação da eficiência quântica (em porcentagem) frente ao comprimento de onda para vários detectores em uso (incluindo o olho humano).

Para um sistema que produza imagens é conveniente também definir a **resolução**. Desde o ponto de vista elementar, chamamos resolução linear ao diâmetro mínimo de um elemento espacial, no qual os sucessos podem ser registrados e diferenciados daqueles outros elementos situados no dispositivo de detecção. No entanto, os astrônomos e ópticos determinam a resolução em pares de linhas por milímetro e para dispositivos deste tipo, a resolução vem dada como o inverso do diâmetro máximo do disco de confunsão. Uma definição mais rigorosa da resolução implica na denominada "função de transferência de modulação", mas isto está longe dos objetivos desde artigo (para detalhes ver: [1]).

## 3 - O EFEITO FOTOELÉTRICO SUPERFÍCIES FOTOEMISSORAS (FOTOCÁTODO)

O processo mediante o qual um fóton, ao incidir sobre um sólido, libera um elétron se denomina **efeito fotoelétrico**, que foi descrito audaciosamernte por Einstein (1905). Segundo Einstein, quando a energia do fóton incidente é superior à energia de ligação do elétron ao sólido, este pode liberar-se. Já que a intensidade de um gás luminoso monocromático é proporcional ao número de fótons, o número de elétrons liberados é proporcional a esta, com o qual o processo é linear.

A eficiência quântica de fotoemissão depende do tipo de sólido que tenhamos. Segundo a teoria de bandas (na qual se consideram as interações entre os elétrons de condução e os núcleos iônicos cristalinos), os elétrons nos cristais estão distribuídos nas bandas de energia separadas por regiões chamadas bandas proibidas, nas quais a entrada do elétron não é permitida [11]. A disposição destas bandas proibidas determina o caráter do sólido. Num material isolante, a energia da banda proibida, $E_G$, que separa a banda de valência da banda de condução é alta (~ 6 eV). Num metal, as bandas de condução e valência estão misturadas, de forma que é fácil arrancar elétrons. Mesmo que, em princípio, se utilizam superfícies fotoemissivas, sua alta capacidade refletora à radiação incidente, lhes fazem possuir valores baixos de $Q_D$ ~ 0.1% no visível. Uma boa superfície fotoemissora deve possuir, não só valores baixos de $E_G$, mas também, uma baixa afinidade eletrônica e baixa capacidade refletora. Estas características são compartilhadas pelos semicondutores, para os quais $E_G$ ~ 1 eV. Os compostos de Césio (*Cs*) e outros metais alcalinos são especialmente eficientes nas partes visíveis do espectro.

Na Figura 2 aparece o diagrama de níveis (bandas) de energia para um semicondutor. A fotoemissão ocorre quando o fóton incidente tem uma energia superior a $E_U$ dada por:

$$E_U = E_A + E_G \qquad (3)$$

onde, $E_A$ é a energia necessária para superar a interface cristal-vazio (afinidade eletrônica).

A presença de impurezas no cristal semicondutor (átomos doadores ou receptores), mesmo que em proporções pequenas, produz variações no nível de energia. Por exemplo, um átomo de Boro (*B*) numa rede de Silício (*Si*) dá $E_A$ ~ 0.1 - 0.01 eV. Apesar de tão baixos valores, a presença de impurezas influe drasticamente nas propriedades do material.

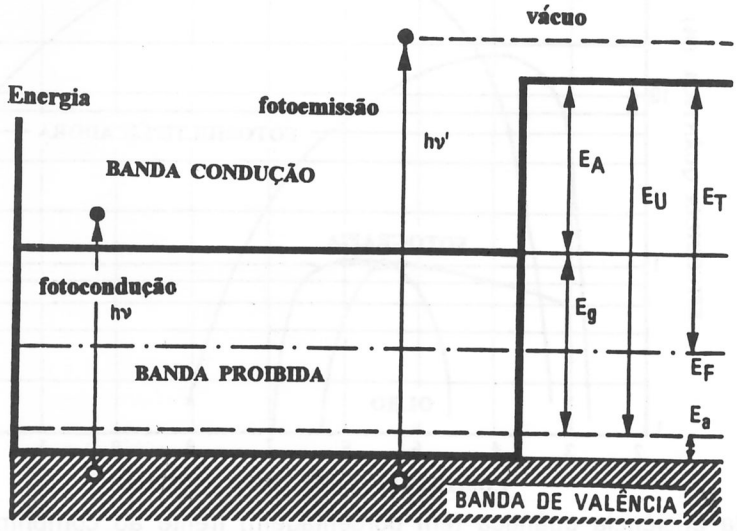


**Figura 2** - Diagrama das bandas de energia para um sólido semicondutor. A nomenclatura está descrita no texto.

Um problema já citado é a aparição de ruído devido a emissão secundária produzida pelo efeito termoiônico. Isto ocorre quando a energia $E_U$ é maior que $E'_U$, onde:

$$E'_U = E_A + E_F \qquad (4)$$

sendo, $E_F$ a denominada energia do nível de Fermi (que é a correspondente ao nível energético mais alto que chegam os elétrons na temperatura de zero grau absoluto), e que nos semicondutores se encontra no interior da banda proibida de energia, sendo que em geral $E_U < E'_U$ [11]. Esta emissão termoiônica pode ser reduzida pelo esfriamento a baixas temperaturas, utilizando por exemplo o Nitrogênio líquido.

A Figura 3 mostra a curva de sensibilidade espectral ($Q_D$ versus $\lambda$) para alguns fotocátodos semitransparentes comuns. Outros fotocátodos opacos, nos quais os fotoelétrons são emitidos desde a mesma superfície sobre a que incide a luz, têm valores de $Q_D$ de até 70%. Além disso, gozam de um alto intervalo dinâmico [3]. Mais tarde descobriram-se fotocátodos com afinidade eletrônica negativa, como é o caso do GaAs(Cs) que tem alta eficiência quântica no vermelho e infravermelho.

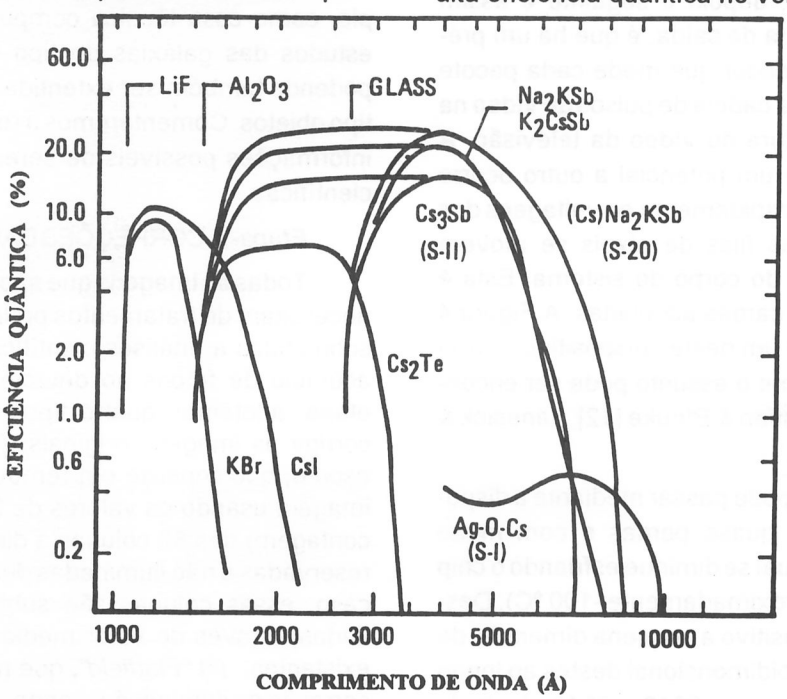


**Figura 3 -** Eficiência quântica ($Q_D$) frente ao comprimento de onda (l) para diversos tipos de fotocátodos semitransparentes usados com diferentes materiais para a janela de entrada do detector.

## 4 - O CCD (DISPOSITIVO DE CARGAS ACOPLADAS)

Durante os anos de 1969 e 1970, Boyle & Smith [2] desenvolveram nos Laboratórios Bell um semicondutor que apresentava uma alta capacidade de armazenamento de imagens ópticas, envolvendo a tecnologia de semicondutores de óxido de metal (MOS) como sensores de estado sólido para uso nos sistemas de TV. A este dispositivo denominaram de CCD (Charge-Coupled Devices). Posteriormente, graças à NASA, JPL e Texas Instruments, foram desenvolvidos CCD's com vistas à aplicação em astronomia espacial. Outros CCD's de imagem foram também construídos por diferentes casa comerciais e são disponíveis hoje em dia para diferentes aplicações.

Basicamente se trata de um dispositivo bidimensional de capacitores MOS que estão depositados sobre a superfície de um substrato de silício, capazes de armazenar e transferir sinais analógicos ou digitais a partir de entradas eletrônicas ou fotoeletrônicas.

Em geral e independentemente das modificações introduzidas pelos fabricantes, cada pixel de um CCD está composto de três camadas: uma superior de eletrodos metálicos, outra intermediária de dióxido de silício ($SiO_2$) e uma inferior constituída por um cristal de silício semicondutor dopado, seja positiva ou negativamente. A camada superior contém os eletrodos de alumínio que se organizam em série intercaladas de dois, três ou quatro. A camada intermediária separa os eletrodos do substrato inferior, contendo em alguns CCD's eletrodos de polisilício que alternam com os metálicos.

O princípio de funcionamento é o seguinte: a luz que chega ao semicondutor de silício (neste caso do tipo *p*) produz os portadores de carga. Aplicando uma voltagem apropriada no centro do eletrodo, cria-se ao mesmo

tempo um poço de potencial (por exemplo, para um do tipo *p* uma voltagem positiva cria uma região vazia de carga), na qual se preenche temporariamente com os portadores criados pelos fótons assim provenientes da difusão térmica. Estes últimos constituem a corrente de escuro. Cada poço de potencial está assim espacialmente bem definido (constituindo cada pixel de uns 15 a 30 microns) na superfície do semicondutor.Depois do período de integração de luz, isto é, a carga que se coletou em cada poço de potencial se move unidirecionalmente ao elemento adjacente (uma barreira impede o movimento em outra direção), depois o seguinte e assim sucessivamente, até a linha de saída, é que há um pré-amplificador e um amplificador que mede cada pacote de carga, saindo como uma cadeia de pulsos de vídeo na seqüência de uma varredura do vídeo da televisão. A transferência de carga de um potencial a outro ocorre modificando sequencial e rapidamente as voltagens dos eletrodos, isto é, todas as filas de pixels se movem simultaneamente através do corpo do sistema. Esta é basicamente a técnica de cargas acopladas. A Figura 4 ilustra esquematicamente um destes dispositivos. Uma análise mais profunda sobre o assunto pode ser encontrada nos trabalhos de Kristian & Blouke [12], Janesick & Blouke [13] e Khosla [10].

Deste modo a carga pode passar mediante a disposição dos eletrodos sem quase perdas e com muito pouca adição de ruído (o qual se diminue esfriando o chip CCD à temperatura de aproximadamente -100 °C). Destaca-se neste tipo de dispositivo a pequena dimensão de cada pixel e a disposição bidimensional destes ao longo do CCD, perfazendo algo como 2048 x 2048 pixels de área. Também sua estabilidade geométrica, longa vida útil, escasso consumo de energia, alta eficiência quântica (que para um MOS de silício chega a ser de 80% entre 5.000 e 7.500 Å), amplo intervalo dinâmico (~ 5000 Å) e alta fidelidade fotométrica fazem deste dispositivo eletrônico excelente meio de aquisição de dados astronômicos.

A entrada deste dispositivo na astronomia teve lugar no ano de 1975 ao utilizá-lo como câmara acoplada aos grandes telescópios espalhados pelo mundo (fotometria superficial de planetas, nebulosas, galáxias, etc), chegando assim a alcançar baixas magnitudes visuais (~ 27), até então impossíveis através de outra técnica. Devido à grande revolução que causou no meio científico na última década, câmaras CCD equiparam o Telescópio Espacial Hubble, muitos outros satélites e sondas espaciais, além de vários observatórios que estão atualmente em operação.

## 5 - A FOTOMETRIA SUPERFICIAL DE GALÁXIAS COM O CCD

O objetivo fundamental da fotometria superficial é o de extrair o máximo possível de informações contidas numa imagem bi-dimensional (seja através de uma placa fotográfica ou por um detector CCD). Com ela, é possível obter as propriedades morfológicas e estruturais de diferentes corpos celestes, de forma a fornecer meios de avaliar as teorias sobre a evolução e formação desses objetos.

Relataremos, após a coleta de dados pelo telescópio, como essa técnica computacional é aplicada aos estudos das galáxias do tipo elípticas e lenticulares, podendo também ser extentida para o estudo de outros tipo objetos. Comentaremos a respeito de alguns tipos de informações possíveis de serem obtidas para análises científicas.

*Etapa-1:* CORREÇÕES DAS IMAGENS ORIGINAIS:

Todas as imagens que são registradas por um CCD necessitam de tratamentos preliminares antes de serem submetidas a análises científicas. Isto ocorre porque o acúmulo de fótons no detector não é perfeito. Nesta etapa, adotamos quatro tipos de procedimentos para corrigir as imagens originais: *(i) Correção do ruído de escuro*, que consiste em remover o ruído adicionado à imagem usando os valores de DN ("Digital Number" ou contagem) das 80 colunas à direita do CCD, que foram reservadas e não iluminadas durante a exposição. Neste caso, essas colunas são subtraídas das linhas dos dados, através do valor médio inferido pelos pixels ali existentes. *(ii) "Flatfield"*, que representa a remoção da correção de iluminação, sendo a parte mais importante e demorada desta etapa. Ao analisar uma imagem visualmente, verifica-se que os pixels de um CCD não possuem a mesma sensitividade, mesmo quando iluminados uniformemente. Dessa forma, aparecem padrões característicos em alguns pixels, os quais não são informações associadas ao objeto de interesse. Assim, necessitamos dividir cada imagem original pelo "flatfield" para obter uma estimativa da imagem real da fonte observada. Na prática, a resposta do "flatfield" pode ser medida apontando o telescópio para uma tela branca fixada sobre a cúpula e esta, por sua vez, é iluminada uniformemente por lâmpadas incandescentes. O CCD registra essa radiação de forma uniforme, dando uma imagem "flatfield" de alta razão sinal-ruído. *(iii) Correções cosméticas*, são necessárias devido a que alguns CCD's apresentam colunas ruins, isto é, pixels que não transferem suas cargas efetivamente quando o detector é descarregado após o término da exposição. Neste caso, faz-se uma interpolação linear dos valores dos pixels das colunas degradadas, mesmo que tenhamos uma perda na reso-

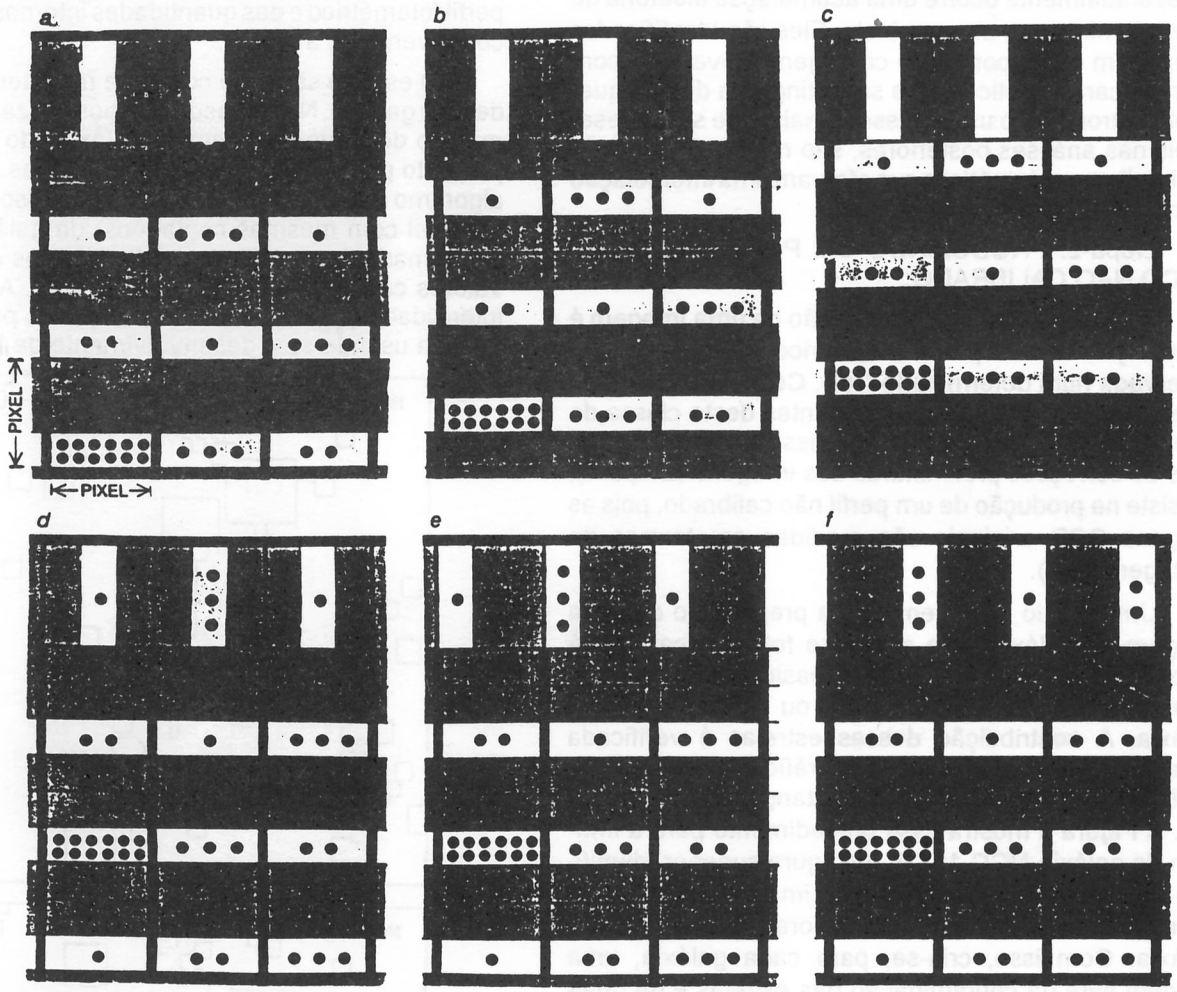


**Figura 4** - O princípio de funcionamento do CCD está esquematizado nestes diagramas, onde cada qual corresponde a um pequeno segmento próximo à parte mais profunda do dispositivo. As três barras verticais em cada diagrama são canais condutores de elétrons e representam uma secção do dispositivo imagiador; as barras horizontais na parte de cima são os registros de saídas seriais. Três pixels são mostrados em cada canal. Cada pixel está subdividido em três partes: uma parte baixa (potencial bom) e duas partes altas (barreira de potencial). Os pesos das três partes podem ser mudados por meio de um conjunto de eletrodos (não está representado aqui). Os elétrons estão se movimentando para fora através dos canais permanentemente verticais pelos canais de parada (linhas pretas finas). Em *(a)* o CCD começa a ser exposto. Fótons entram pelo dispositivo por detrás. Cada fóton pode liberar um elétron de uma estrutura cristalina retangular de silício. Os elétrons são imediatamente armazenados nas proximidades do potencial bom. Depois de terminada a exposição a imagem é lida pelo movimento do potencial bom, que vai passando de registro para registro num mesmo canal condutor *(b)*. O efeito desta operação é mover os elétrons de um pixel para o outro. Depois de dois desvios *(c,d)* um padrão inteiro de cargas foi movimentado para o canal horizontal, onde encontram-se os registros de saída. A mesma técnica é agora aplicada para mover os pixels ao longo destes registros de saída para a esquerda *(e,f)*. Um amplificador no final do registro de saída mede cada pacote de carga, lendo-o e armazenando-o na memória. O processo é então repetido, até que todos os pixels sejam lidos e o dispositivo imagiador descarregue as informações nele contidas.

lução da imagem final. *(iv) Remoção dos raios cósmicos*, pois, quando o CCD é exposto por um período prolongado, eventualmente ocorre uma acumulação aleatória de raios cósmicos em alguns pixels. Eles são identificados na imagem como pontos de contagens elevadas e com padrões característicos, que se distinguem de qualquer objeto astronômico usual. Esses sinais, que são indesejáveis nas análises posteriores, são removidos através de algoritmos automáticos que efetuam uma interpolação linear nas regiões afetadas.

*Etapa-2:* PRODUZINDO UM PERFIL FOTOMÉTRICO NÃO CALIBRADO:

A meta do processo de redução de uma imagem é a produção de um perfil fotométrico de uma galáxia observada num determinado filtro. Com isso, podemos obter características físicas marcantes desta classe de objeto astronômico. O primeiro passo neste processo, após as correções preliminares das imagens (Etapa-1), consiste na produção de um perfil não calibrado, pois as imagens CCD originais são medidas em termos de contagens (DN).

Um estágio inicial envolve a preparação de cada imagem de galáxia para a análise fotométrica. Isso é feito interativamente e consiste basicamente em: *(i) Remoção das estrelas* próximas ou superimpostas à galáxia. A contribuição dessas estrelas é verificada visualmente utilizando um cursor gráfico, com auxílio do qual foram delimitadas regiões retangulares de exclusão. A Figura 5 mostra esse procedimento para a imagem da galáxia NGC 1533[2]. Na figura superior identificou-se as regiões de exclusão da imagem e, na figura inferior, essas mesmas regiões foram superpostas à galáxia. Com isso, cria-se, para cada galáxia, uma imagem livre da contaminação das estrelas e de qualquer outro sinal indesejável. *(ii)* A *determinação da intensidade do nível de fundo do céu* é o passo mais delicado da fotometria superficial, pois envolve fontes de erros que afetam o perfil final da galáxia. Isso ocorre porque o brilho do céu durante a noite é consideravelmente maior que o das regiões mais externas das galáxias. O nível de fundo do céu é o número médio das contagens por pixel numa região da imagem que esteja livre de qualquer objeto astronômico, o que nem sempre é fácil de ser medido. Na prática, define-se de 3 a 4 regiões retangulares nas bordas da imagem o mais distante possível da galáxia e das regiões de exclusão impostas pela remoção das estrelas. Um algoritmo para determinação do céu foi desenvolvido onde mostra-se um histograma dos valores dos pixels e de um ajuste Gaussiano dessa distribuição. O valor central do ajuste provê a melhor estimativa do valor modal dos pixels e, por conseguinte, da intensidade do nível do céu. (iii) A *localização do centro da galáxia* envolve a determinação do centróide (posição onde se encontra o máximo do brilho do objeto de interesse) numa região escolhida próxima ao centro visual do objeto. É fundamental a correta determinação desta quantidade para o cálculo do perfil fotométrico e das quantidades internas desse perfil, como veremos a seguir.

O estágio seguinte constitue na obtenção do perfil de uma galáxia. Neste caso, estamos utilizando o mesmo método desenvolvido para as galáxias do tipo elípticas proposto por Djorgosvski [5] e referências ali citadas. O algoritmo desenvolvido assume que as isofotas (curvas de nível com mesmas contagens) da galáxia são bem aproximadas por uma série de contornos elípticos com valores crescentes do semi-eixo maior. A variação da intensidade ($I_{a,\phi}$) nos contornos elípticos pode ser aproximada usando-se o desenvolvimento de Fourier:

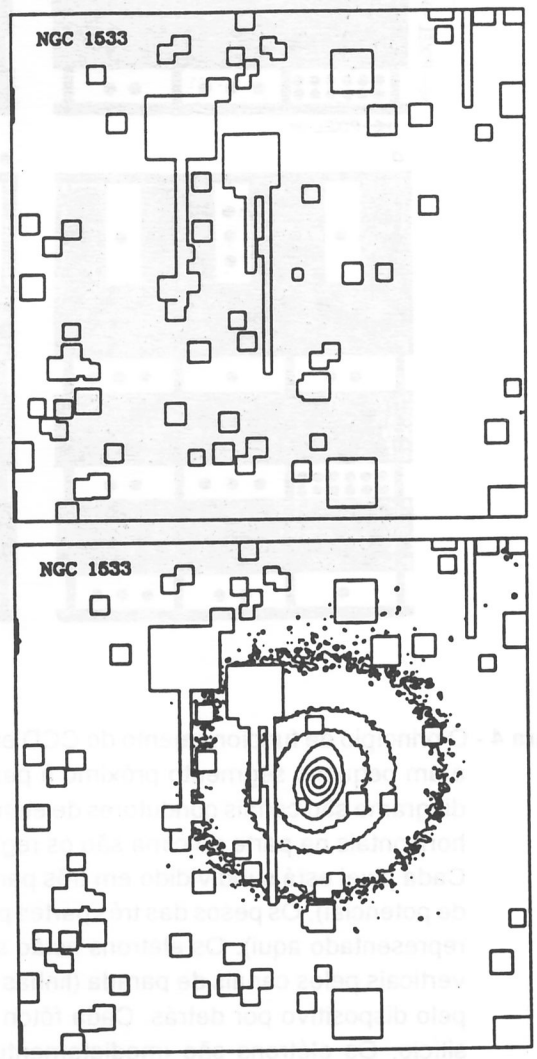


**Figura 5** - (*superior*) Regiões de exclusão escolhidas visualmente para a imagem CCD da galáxia NGC 1533. (*inferior*) As mesmas regiões de exclusão sobrepostas à imagem da galáxia, eliminando-se com isso, as estrelas e sinais espúrios à imagem.

(2) NGC significa: New General Catalogue e IC significa Index Catalogue, que são catálogos de galaxias brilhantes publicados por [6], [7] e [8].

$$I_{a,\phi} = I_a^0 + A_1 \cos\phi + B_1 \operatorname{sen}\phi + \\ + A_2 \cos 2\phi + B_2 \operatorname{sen} 2\phi \qquad (5)$$

onde, $\Phi$ é a anomalia excêntrica para cada ponto sobre a elipse definida na forma:

$$x = a(\cos\phi)$$
$$y = a[(1-\varepsilon)\operatorname{sen}\phi]$$

sendo, $I_a^0$ a intensidade média ao longo da elipse e $A_i$ e $B_i$ são os coeficientes de Fourier. Ao realizar a integração ao longo da imagem, nesta etapa também são computados os erros associados a cada grandeza de interesse.

Um exemplo de perfil fotométrico está ilustrado na Figura 6, onde mostramos os perfis ao longo do eixo-maior para o brilho superficial em unidades de DN/pixel, as intensidade em anéis elípticos e circulares, a elipticidade (e) e o ângulo de posição (q), extraídos para a galáxia NGC 1533. Nestes perfis, o passo radial foi de um pixel, de forma que há uma superposição de informações na parte mais externa de cada grandeza medida. Isto ocorre porque o passo nessa região acaba sendo muito pequeno e a relação sinal-ruído em cada iteração torna-se pobre. Uma forma de contornar este problema é fazendo uma reamostragem, isto é, fazendo uma média para os pontos na região mais externa dos perfis para distâncias radiais superiores a 30" (ou neste caso, 30 pixels) a partir do centro do objeto (Figura 7).

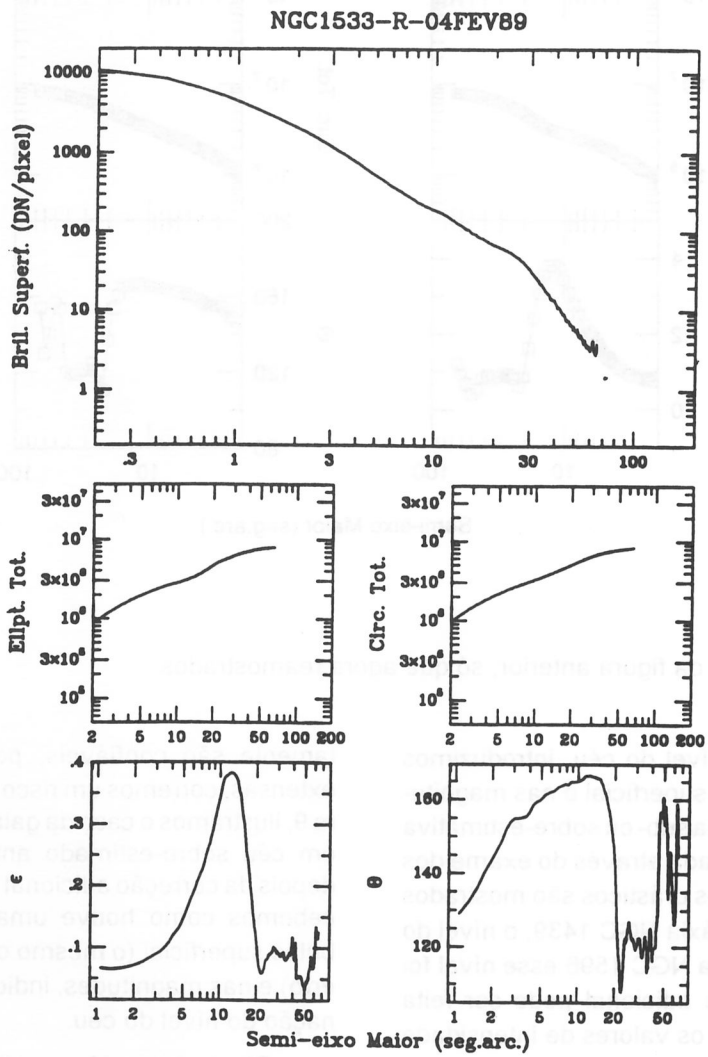


**Figura 6** - Exemplo dos perfis (brilho superficial, magnitude elíptica total, magnitude circular total, elipticidade e ângulo de posição) obtidos por um programa específico, para a galáxia NGC 1533.

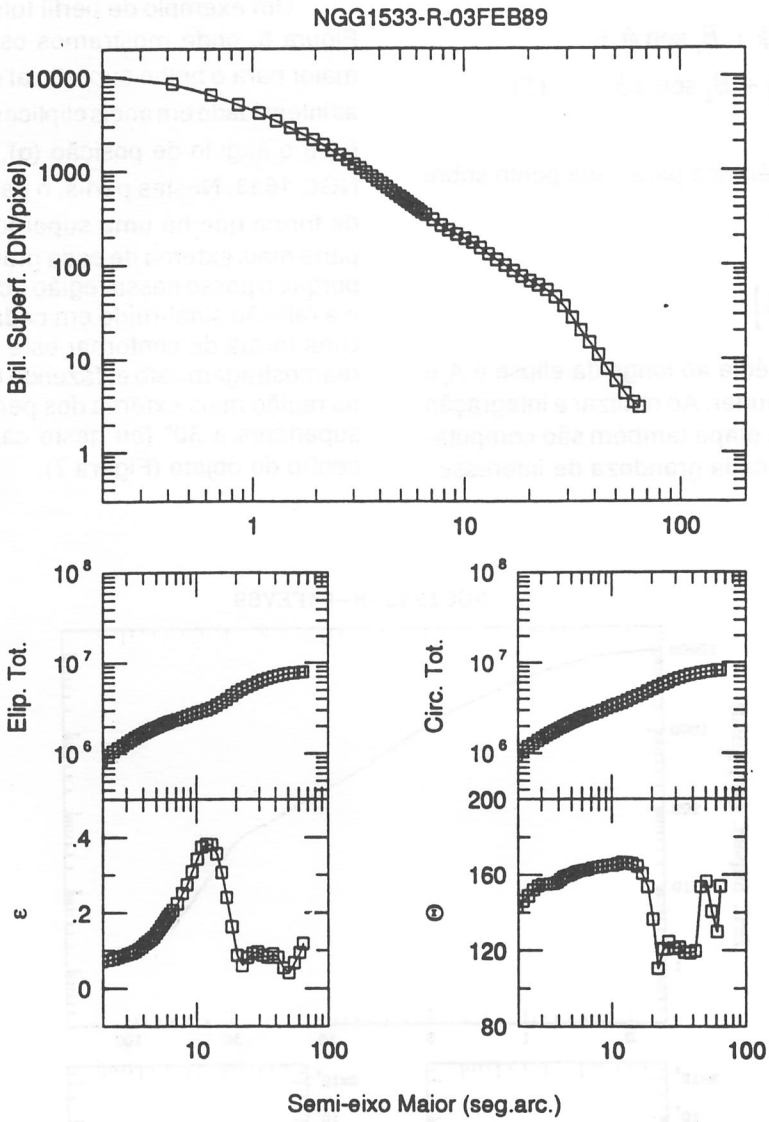


**Figura 7** - Os mesmos perfis da figura anterior, só que agora reamostrados.

Quando estimamos o nível do céu, introduzimos um erro sistemático no brilho superficial e nas magnitudes integradas, refletindo numa sub- ou sobre-estimativa do céu, e que pode ser apreciada através do exame dos perfis fotométricos. Dois casos drásticos são mostrados na Figura 8, onde, para a galáxia NGC 1439, o nível do céu foi sobre-estimado, e para NGC 1596 esse nível foi sub-estimado. Uma correção adicional pode ser feita para esses casos verificando os valores de intensidade para raios elevados e analisa-se que valor deve ser acrescido ou diminuído nas intensidades e nas barras de erro. Lembramos que, para aquelas imagens que são pequenas em relação ao detector, essas correções certamente são confiáveis, porém para imagens muito extensas, corremos um risco nessas correções. Na Figura 9, ilustramos o caso da galáxia NGC 1416 que possuía um céu sobre-estimado antes (retângulos abertos) e depois da correção adicional (retângulos fechados). Percebemos como houve uma significativa variação no brilho superficial (o mesmo ocorrendo para as barras de erro) e nas magnitudes, indicando uma correta determinação do nível do céu.

Diante da grande quantidade de dados que estamos coletando nos telescópios do *LNA* e que estão sendo reduzidos parcialmente no *DF/PUCCAMP*, ilustramos na Figura 10 alguns perfis de brilho já calibrados de diferen-

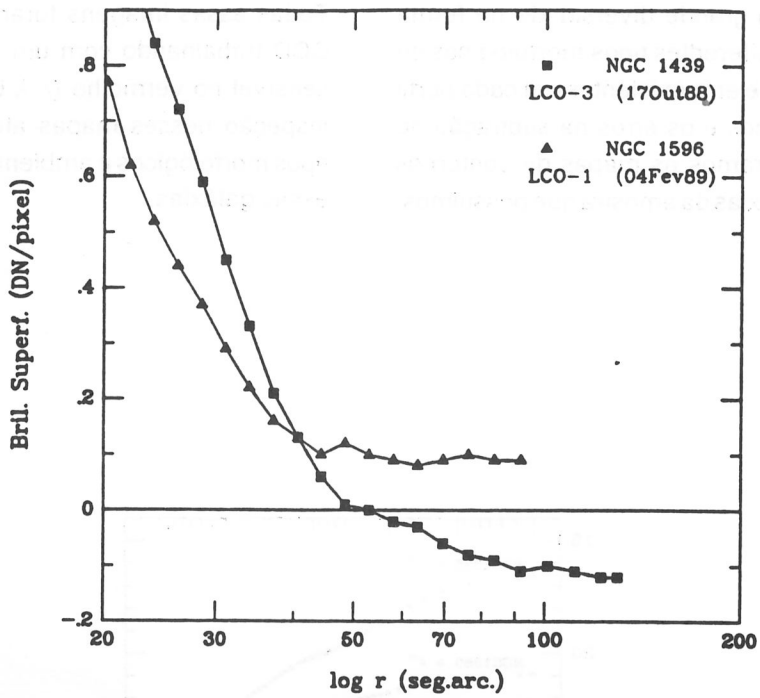


**Figura 8** - Exemplos de duas galáxias reduzidas com o céu sobre-estimado (NGC 1439) e sub-estimado (NGC 1596). Os perfis foram reamostrados, mas não calibrados. Obviamente, correções devem ser efetuadas em ambos perfis. O valor 0 (zero) representa o nível correto do fundo de céu.

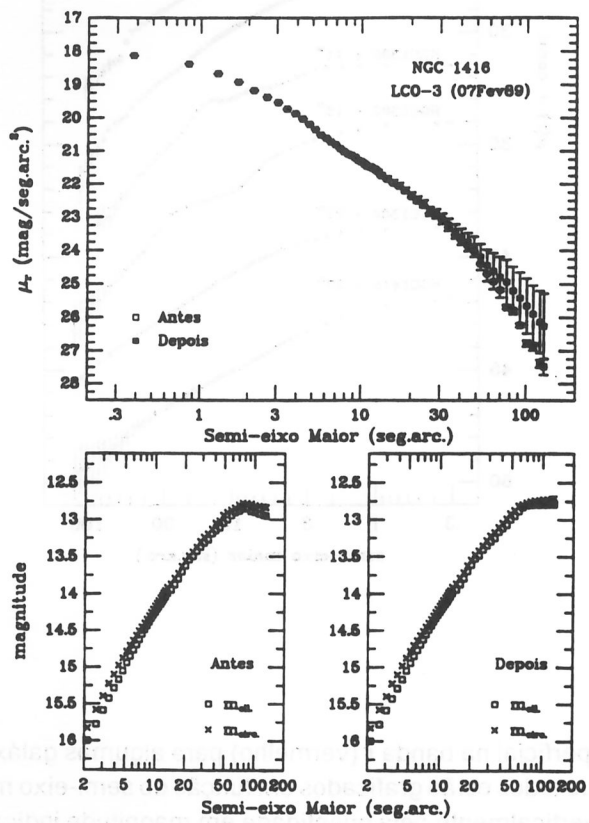


**Figura 9** - Os efeitos das correções no nível do fundo de céu para os perfis da galáxia NGC 1416: em brilho superficial (painel superior) e em magnitudes elíptica e circular (painéis inferiores).

tes galáxias. Notamos a grande diversidade na forma dos perfis, refletindo os diferentes tipos morfológicos de cada galáxia. As barras de erro existentes em cada perfil incluem os erros aleatórios e os erros na subtração do céu. Na Figura 11 mostramos os mapas de contornos isofotais de algumas galáxias da amostra que possuimos.

Todas essas imagens foram obtidas com um detector CCD trabalhando com um filtro **r** de Gunn-Thuan [19] sensível no vermelho ($\lambda\ \lambda$ 6550 ± 450 Å). Uma rápida inspeção nesses mapas atesta a ampla variedade de tipos morfológicos e ambientes em que estão localizados essas galáxias.

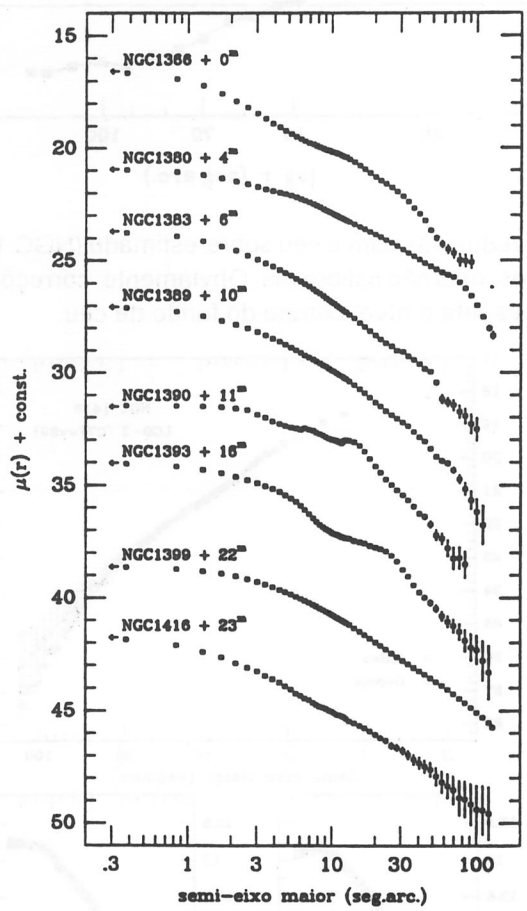


**Figura 10 -** Perfis de brilho superficial na banda **r** (vermelho) para algumas galáxias de que dispomos medidas fotométricas, sendo que todos estão graficados em função do semi-eixo maior do objeto. Os perfis individuais foram desviados verticalmente pela quantidade em magnitude indicada ao lado do nome da galáxia. As setas indicam os níveis de brilho superficial central. As barras de erros incluem as contribuições dos erros aleatórios e das estimativas dos erros na subtração do céu.

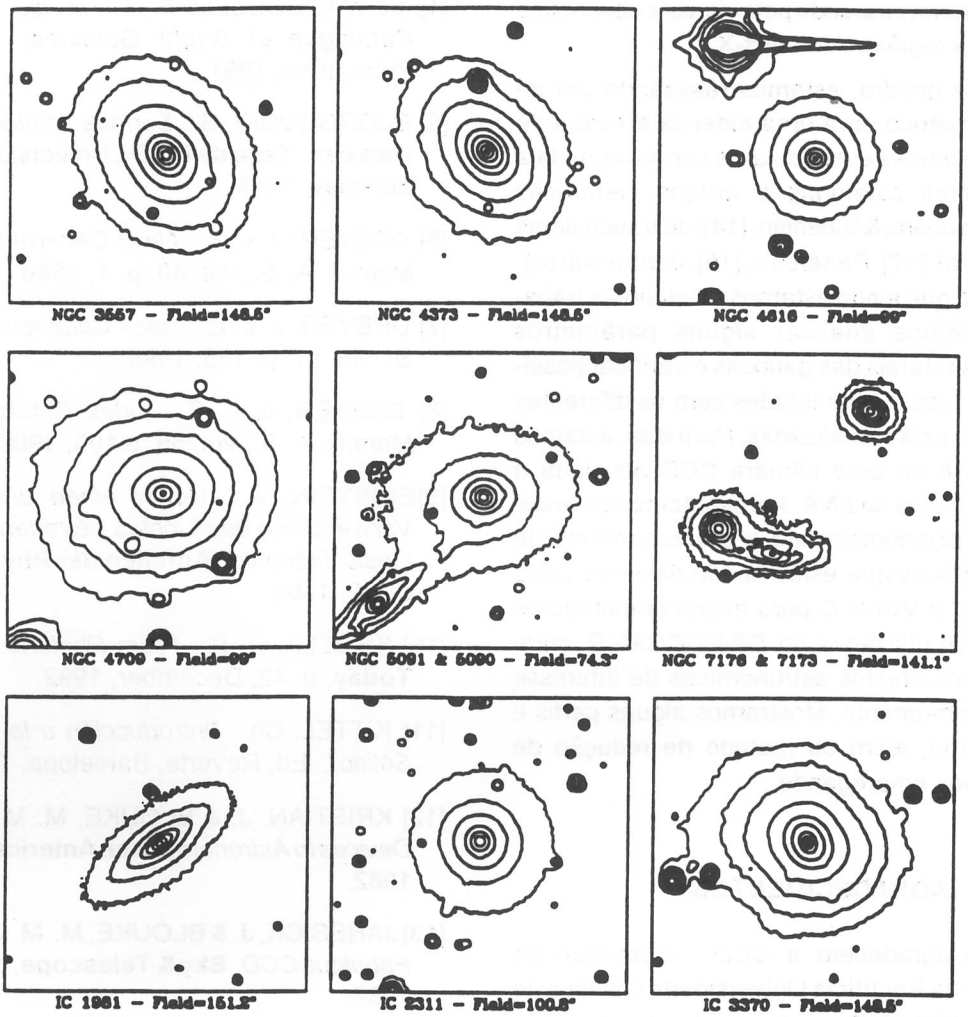


**Figura 11** - Seleção de alguns mapas de contorno na banda **r**, onde o CCD é mais sensível. O tamanho angular da imagem está discriminado na parte inferior de cada painel. O Norte está orientado para cima e o Leste, para a direita.

## 6 - COMENTÁRIOS FINAIS

Apresentamos sucintamente algumas informações sobre diferentes sistemas de detecção em astronomia. Basicamente o olho humano, a emulsão fotográfica, a fotomultiplicadora e o CCD. Na verdade existe uma ampla variedade de detectores e a escolha entre um e outro depende principalmente dos projetos observacionais a empreender. Os detectores undimensionais são mais simples e seu principal emprego se vislumbra na espectroscopia. Outros oferecem alta precisão fotométrica bidimensional e marcada dependência com a eficiência quântica com o comprimento de onda, o que pode ser decisivo em sua escolha. Alguns estão dedicados exclusivamente a observações de fontes luminosas fracas e outros a brilhantes.

Pode ser que no futuro os astrônomos e engenheiros tenderão a escolha de um a dois detectores trabalhando num amplo domínio de comprimentos de onda. Um candidato poderia ser o sistema de contador de fótons, dada sua fidelidade, possibilitando o trabalho em tempo real e para fluxos de luz muitíssimos baixos, o que seria viável para os estudos de fontes luminosas que variam com rapidez (por exemplo: fontes de raios-X) e para espectrofotometria e imagem bidimensional de fontes fracas (galáxias e quasares). Outro candidato é sem dúvida o CCD dadas suas propriedades de miniaturização e alta fidelidade. Atualmente, seu uso se extende a quase todos os domínios do espectro, indo

desde o infravermelho, passando pelo visível e ultravioleta e ingressando nas regiões dos raios-X.

Diante deste quadro, estamos realizando um levantamento fotométrico para uma extensa amostra de galáxias do tipo elípticas e lenticulares confeccionada a partir de diferentes catálogos e artigos científicos (Djorgovski, [5]; Lauberts & Valentijn, [14]; de Vaucouleurs *et al.*, [4]; Rubin *et al*, [17], Penereiro, [15]; dentre outros). Com esta amostra que ainda estamos coletando e trabalhando, pretendemos analisar alguns parâmetros morfológicos e estruturais das galáxias e verificar possíveis correlações dessas quantidades com os diferentes ambientes em que estão localizadas. Para isso, estamos empregando o uso de uma câmara CCD acoplada a telescópios disponíveis no *LNA*. Apresentamos os resultados de alguns algoritimos para o processamento de imagens astronômicas que estão sendo desenvolvidos em linguagem C$^{++}$ e Visual C para microcomputadores da linha IBM-PC e utilizados no *DF/PUCCAMP*, onde aplicamos a alguns objetos astronômicos de interesse observados até o momento. Mostramos alguns perfis e mapas de contorno, além do método de redução de dados que estamos empregando.

## 7 - AGRADECIMENTOS

Os autores agradecem a *CCD* - Comissão de Carreira Docente da Pontifícia Universidade Católica de Campinas - PUCCAMP, pelo apoio à pesquisa interdisciplinar que vem sendo realizada no *I.C.E.* e *I.I.* (processo Número 132/94). Agradecemos também ao pessoal de apoio técnico do *LNA* (Laboratório Nacional de Astrofísica, MCT-CNPq) pelas noites concedidas durante as missões observacionais realizadas nos telescópios ali disponíveis.

## REFERÊNCIAS BIBLIOGRÁFICAS

[1] BOCKSENBERG, A. *Optical Astronomy.* **Nature**, vol. 298, p. 795, 1982.

[2] BOYLE, W. W. & SMITH, G. E. *CCD A new optical detector.* **Bell Syst. Tech. J.**, vol. 49, p. 587, 1970.

[3] CARRUTHERS, G. R. *Eletronic imaging devices in astronomy.* **Astrophys. Space Sci.**, vol.14, p. 332, 1971.

[4] de VAUCOULEURS, G. *et al. Third Reference Catalogue of Bright Galaxies.* Springer-Verlag Publication, 1991.

[5] DJORGOVSKI, S. *Surface Photometric of Elliptical Galaxies.* **Tese de Ph.D.**, Universidade da Califórnia, Berkeley, 1985.

[6] DREYER, J. L. E. *Newl General Catalogue - NGC.* Mem.R. A. S., vol. 49, p. 1, 1888.

[7] DREYER, J. L. E. *Index Catalogue - IC.* Mem.R. A. S., vol. 51, p. 185, 1895.

[8] DREYER, J. L. E. *Index Catalogue - IC, Part-II.* Mem.R. A. S., vol. 59, p.105, 1909.

[9]EINSTEIN, A. *Ueber einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesischtspunkt.* **Annalen der Physik**, 4.Folge, XVII, p.132, 1905.

[10] KHOSLA, R. P. *From Photons to Bits.* **Physics Today**, p. 42, December, 1992.

[11] KITTEL, Ch. *"Introducción a la Física del Estado Sólido"*, Ed. Reverté, Barcelona, 1975.

[12] KRISTIAN, J. & BLOUKE, M. M. *Charge-coupled Devices in Astronomy.* **Sci. Americam**, p. 66, October, 1982.

[13] JANESICK, J. & BLOUKE, M. M. *Sky on a Chip: The Fabulous CCD.* **Sky & Telescope**, p. 238, September, 1987.

[14] LAUBERTS, A. & VALENTIJN, E. A. *The Surface Photometric Catalogue of the ESO Uppsala Galaxies.* European Southern Obs. (Special Public), 1989.

[15] PENEREIRO, J.C. *Um Levantamento em Fotometria Superficial CCD para Galáxias E's e S0's.* **Tese de Doutorado**, IAG-USP, São Paulo, 1993.

[16] REID, I. N. *et al. The Second Palomar Survey.* **P.A.S.P.**, vol.103, p. 661, 1991.

[17] RUBIN, V. C. *et al. Optical properties and dynamics of galaxies in the Hickson Compact Groups.* **Ap. J. Suppl. Ser.**, vol. 76, p. 153, 1991.

[18] RUDOLPH, R., *et al. Characteristics of Optical Multi-element Detectors.* **Astron. Ap.**, vol. 65, L5-6, 1978.

[19] THUAN, T. X & GUNN, J. E. *A new four-color intermediate-band photometric system.* **P.A.S.P.**, vol. 88, p. 543, 1976.

# PROJETO CONJUNTO HARDWARE/SOFTWARE

# HARDWARE/SOFTWARE CODESIGN

**Ricardo PANNAIN***
**Frank Herman BEHRENS****

## ABSTRACT

Hardware/software codesign is not a new approach. It has been used since the first microprocessors and microcomputers came out to the market. Advances in design, synthesis and verification of integrated circuits technology, and the increase of the necessity towards to developed integrated hardware/software systems, are driving hw/sw designers to a new metodology. In this paper we introduce some concepts about this approach and discuss some problems in the development of this systems.

**KEY-WORDS:** hardware/software codesign, ASIC's - application-specific integrated circuits, real-time embedded systems.

## RESUMO

O projeto de sistemas que integram software e hardware não é uma abordagem nova, mas nos últimos anos ganhou mais força devido ao aparecimento de um número cada vez maior de aplicações que necessitam um sistema integrado hardware/software, à necessidade de diminuição de custos de projetos e de testes e ao avanço tecnológico em algumas áreas, tais como, síntese lógica e métodos formais de descrição, que facilitam o desenvolvimento destes sistemas. Neste artigo procuramos introduzir alguns conceitos sobre metodologias para este enfoque de projeto, assim como mostrar as dificuldades encontradas no desenvolvimento do mesmo.

**PALAVRAS-CHAVE**: integração hardware/software, circuitos integrados de aplicação específica, sistemas de tempo real integrados.

## 1. INTRODUÇÃO

Um sistema integrado HARDWARE/SOFTWARE é aquele onde se tem a co-existência de hardware e software, cada qual com uma função específica, e que se comunicam através de interfaces, bem definidas e de alto desempenho [1 - 7]. A maioria destes sistemas, consiste de um processador de propósito geral, memória e um circuito integrado de aplicação específica (ASIC's- *Application Specific Integrated Circuits* ) interligados. Exemplos deste conceito podem ser vistos em [8 - 10].

A abstração para este modelo de sistema, mostrada na Figura 1, consiste na divisão em duas partes que se interagem. Estas partes são chamadas de domínios,

(*) Engenheiro Eletricista, mestre em Engenharia Elétrica e doutorando em Engenharia Elétrica pela UNICAMP. Atualmente é Coordenador e professor do Curso de Engenharia de Computação do I. I. da PUCCAMP e professor do Departamento de Ciência da Computação do IMECC/UNICAMP.
(**) Engenheiro Eletricista, mestre e doutor em Engenharia Elétrica. Atualmente é professor do Curso de Engenharia de Computação do I. I. da PUCCAMP.

portanto temos o domínio do software e o domínio do hardware. O domínio do software é formado pelas camadas de programas de aplicação, do sistema operacional, dos *drivers* de entrada e saída de dados e do barramento de entrada e saída de dados. O domínio do hardware é formado pelas camadas dos circuitos de aplicação específica, da interface entre este circuito e o barramento de entrada e saída e do barramento propriamente dito.

Ainda neste tipo de abstração, é possível dois tipos de interação: a interação vertical, intra-domínio, e a interação horizontal, inter-domínio. Na interação vertical uma camada interage com as suas camadas vizinhas, ou seja, no caso do domínio do software, os programas interagem com o sistema operacional, este com os *drivers*, que por sua vez interagem com o barramento. No caso do domínio do hardware, os circuitos interagem com a interface de barramento, que por sua vez interage com o barramento. Na interação horizontal, algumas camadas de um domínio interagem com determinadas camadas do outro, por exemplo, a camada de programas (domínio do software) interage com a de circuitos (domínio de hardware) através do envio e recebimento de mensagens; a camada de *drivers* (domínio do software) interage com a de interface (domínio do hardware) através de ações de escrita e leitura de registradores. O barramento é o meio físico utilizado para transações entre o sistema que executa o software e o sistema que comporta o circuito de aplicação específica. É importante destacar ainda que, a interação no nível mais baixo, mostra as transações da CPU, levando-se em conta como o hardware de aplicação decodifica, interpreta e reage a estas transações. No próximo nível de abstração, a decodificação de endereços e as interrupções ficam transparentes, permitindo considerar apenas transferências de dados entre os *drivers* e o hardware. Na interação no nível mais superior, detalhes de sistema operacional e drivers de dispositivos são transperentes.

Exemplos destes sistemas são encontrados em instrumentação médica, controle de processos, automatização de veículos e sistemas de comunicação e redes. Estes sistemas também são chamados de sistemas de tempo real integrados ( *real-time embedded systems* ), pois têm restrições de tempo em suas ações.

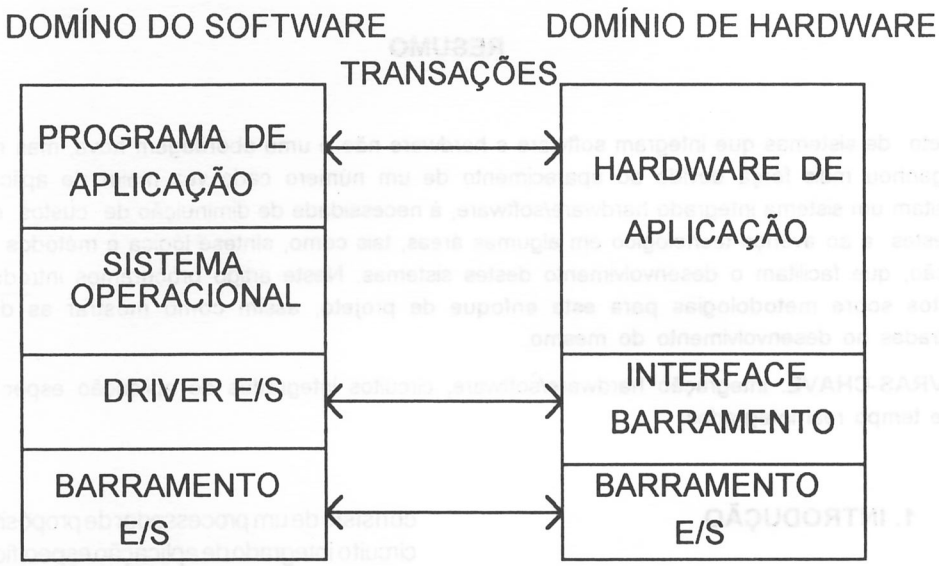


**Figura 1** - Modelo de interação hardware/software

Atualmente, os projetos destes sistemas são feitos sem ferramentas adequadas e a metodologia de projeto utilizada é totalmente *ad hoc*. Por isso, para que esta tecnologia seja amplamente aceita na área industrial, é necessário ainda muito trabalho na definição de metodologias específicas para este tipo de projeto. Estas metodologias devem permitir a especificação do sistema, sua partição e principalmente a especificação das interfaces entres as várias partes (software e hardware), além de seus projetos.

O desenvolvimento de ambientes integrados de ferramentas computacionais (os chamados *frameworks*)

[6], que atendam às metodologias desenvolvidas, é imprescindível para o sucesso de um projeto. Este ambiente deve integrar ferramentas de especificação de sistemas com características de definição de interfaces entre hardware e software, ferramentas de particionamento que ajudem a definir qual parte será implementada em software e qual será implementada em hardware, ferramentas de síntese de circuitos, ferramentas de simulação que comportem a simulação conjunta do hardware e do software, ferramentas de teste e outras. Vários esforços estão sendo feitos para propiciar um ambiente integrado que supra os requisitos para este tipo de projeto, com a preocupação principal de definir metodologias e ferramentas destinadas à especificação e descrição funcional de um sistema hardware/software.

Atualmente só existem ambientes integrados de auxílio a projetos, para sistemas de áreas específicas, como por exemplo processamento digital de sinais (DSP´s).

A seguir apresentaremos uma primeira abordagem de metodologia de projeto [1] [2] e discutiremos alguns aspectos práticos para este tipo de projeto.

## 2. UMA METODOLOGIA PARA HARDWARE/SOFTWARE CODESIGN

Realizar um projeto conjunto hw/sw significa projetar um sistema onde haja a co-existência entre o hardware e o software, ou seja, eles são os componentes do sistema e portanto o seu funcionamento conjunto visa alcançar os objetivos propostos na sua especificação. Com esta definição, temos várias estruturas que se encaixam como hw/sw *codesign*, tanto os ditos aceleradores, onde algumas funções de software são implementadas em hardware, como sistemas onde a concepção do hardware e do software são desenvolvidas em paralelo. Ambos os casos têm uma arquitetura semelhante a mostrada na Figura 2.

A metodologia geralmente utilizada para os aceleradores baseia-se na análise dos pontos críticos da especificação inicial do sistema, normalmente implementada em software, propondo uma partição que será sintetizada em hardware, o qual desempenhará as funções a ele atribuídas de forma mais eficiente do que se permanecesse sob forma de software.

No segundo caso, onde desde a especificação inicial leva-se em conta tanto o subsistema hardware como o software, a metodologia deve ser mais específica, possibilitando maior sucesso no resultado final. Esta metodologia deve possibilitar a co-simulação hardware/software, isto é, a possibilidade de verificar a funcionabilidade de descrições mistas de hardware/software. Ferramentas de síntese devem permitir que rotinas escritas em uma linguagem de programação possam ser implementadas em hardware. Estas metodologias devem também fornecer não só alternativas de particionamento, como um formalismo para descrição de todas estas alternativas.

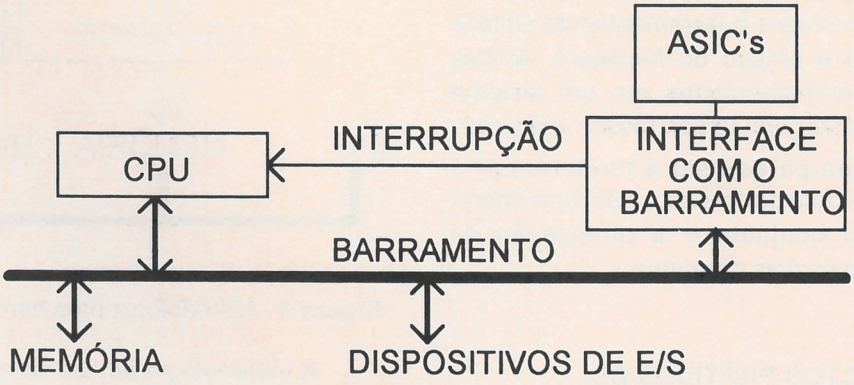


**Figura 2** - Arquitetura geral de um sistema hardware/software codesign

O enfoque descrito em [1], é uma metodologia baseada num modelo para simulação e síntese a nível de sistema, que permite um entendimento detalhado do seu comportamento e possui a capacidade de gerar alternativas de projeto, através de co-simulação e co-síntese. Uma característica importante na co-simulação é a ligação do comportamento da simulação de hardware dentro de um ambiente de software. A co-síntese envolve duas características de projeto, interrelacionadas, que são: a escolha do melhor particionamento hardware/software e a escolha do nível de concorrência do controle.

A partição hardware/software é definida pelo conjunto de funções de aplicação implementadas com o hardware de aplicação específica. O particionamento hw/sw leva em consideração explorar ao máximo os recursos do hardware de aplicação específica, visando atingir os objetivos do projeto. A concorrência do controle de um sistema é difinida pelo comportamento e interação entre seus processos. Algumas vezes, para se encontrar um nível de concorrência apropriado, é necessário redefinir os processos através da divisão ou união de alguns deles, objetivando atingir as metas de desempenho do sistema.

Na Figura 3 podemos ver a metodologia em discussão, que usa um ambiente de co-simulação para a descrição mista hardware/software, produzindo algo funcionalmente correto, mas às vezes não atendendo os objetivos das especificações ou mesmo sendo algo não realizável fisicamente, dependendo da tecnologia em questão. A co-síntese modifica a partição hardware/software e o nível de concorrência do controle, visando sempre atingir os objetivos de comportamento e desempenho do sistema.

Como resultado da co-síntese, obtêm-se as especificações do software e hardware para imple-mantação. Na implementação, usam-se técnicas padrão para o desenvolvimento de software e ferramentas de síntese comportamental para o projeto do hardware. Ambas partes são, então, interconectadas em um sistema computacional de propósito geral, tal qual mostra a Figura 2.

Esta metodologia parece ser a tendência para hw/sw *codesign*, pois cobre os dois aspectos mais importantes: a simulação conjunta e a otimização do particionamento entre os dois domínios.

## 3. ASPECTOS PRÁTICOS DO PROJETO CONJUNTO HW/SW

*Codesign* é uma prática que vem sendo executada desde os primórdios dos projetos dos primeiros computadores digitais, porém, de uma forma não estruturada [3]. Conceitualmente, é impossível o projeto de uma máquina digital programável, tal como a CPU de um computador, sem a intervenção direta dos usuários de programação, que em última análise, especificam os requisitos de software que esta máquina deve apresentar. Sem dúvida alguma, a indústria que projeta e fabrica microprocessadores e computadores aprendeu, às custas de tentativa e erro, como compatibilizar o projeto do hardware destas máquinas com o projeto do software que são executados nas mesmas. A isto denomina-se de metodologia *ad hoc* de projeto. Não obstante, percebe-se que inicialmente se objetivava o desenvolvimento de sistemas de computação (sistemas de hw/sw) de aplicação genérica.

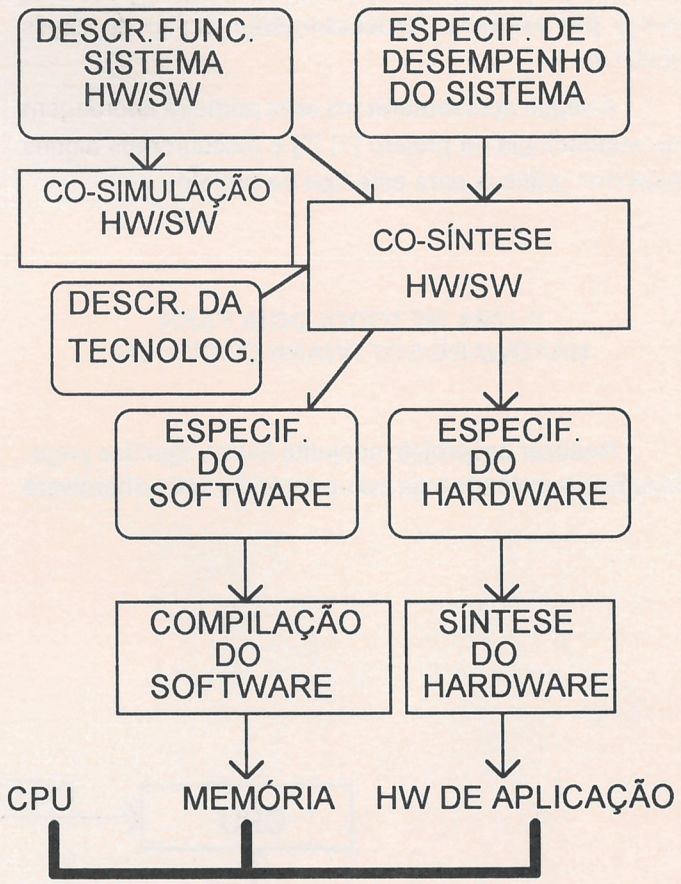


**Figura 3** - Metodologia para hardware/software codesign

A metodologia *ad hoc* se refere principalmente ao fato do grupo de projetistas de hardware/software (desde gerentes de projeto até os engenheiros projetistas de *staff*) serem desenvolvidos e treinados *in-house*, pelas condições de competitividade do fabricante. Exemplos

disto, tem-se a Intel, Motorola, IBM, Compaq, Sun, que estão presentes desde os primeiros desenvolvimentos de microprocessadores e computadores e, certamente, possuem em seus *staffs* a reserva de conhecimentos *ad hoc*, à nível gerencial, de projeto e de tecnologias desenvolvidas. Este panorama não é de todo ruim, se considerarmos a grande competitividade deste grupo de fabricantes.

Por outro lado, a tecnologia gerada pelo desenvolvimento dos sistemas de computação modernos é tal, que pode passar da esfera dos fabricantes primários para a esfera dos usuários, que têm interesse em desenvolvimentos específicos e não genéricos. Esta é uma tendência moderna, observada nos últimos cinco anos.

Tecnologias consolidadas de fabricação estão sendo postas à disposição de usuários, que atualmente podem especificar seus produtos baseados não só em sistemas de computação de propósito geral, como também em módulos específicos. A diretriz para este tipo de sistemas sempre foi: o hardware é de propósito geral, cabendo ao software a função de particularizar a aplicação.

O advento dos ASICs, uma tecnologia madura e bem disseminada, propiciou o aparecimento de um novo campo de desenvolvimento, qual seja, hardware específico segundo a especificação do usuário. A pergunta que a partir daí ficou pendente foi: em que condições utilizar hardware específico com vantagens competitivas sobre o hardware de propósito geral e qual a influência desta solução sobre o software?

Por outro lado, não parece sensato reprojetar uma máquina computacional toda a vez que alguma solução específica é buscada. Sensato seria aplicar o conceito de re-utilização de tecnologias (*technology reuse*) [3], que aproveita desenvolvimentos existentes e incorpora aos mesmos novas características. Desta forma, poder-se-ia atender à demanda de novos produtos, baseados em sistemas computacionais existentes, que contivessem em hardware, partes (algorítmos, funções, etc) antes implementadas em software.

Estas partes em hardware seriam implementadas sob forma de circuitos integrados (ASICs), se comunicando com a CPU de sistemas de propósito geral (microcomputadores, estações de trabalho), à maneira que um co-processador aritmético o faz com a CPU principal. Não se descarta a possibilidade de refabricar uma determinada CPU (8086, 486, Pentium, PowerPC ou Spark), integrando juntamente com a mesma as alterações ou complementos de hardware sob especificação do usuário. A forma final de implementação pode ser decidida com base no estudo de custo-benefício de cada opção: para um produto que se imagina um volume de produção entre pequeno-médio (5.000 a 50.000 unidades) e uma vida útil de poucos anos (2 a 3 anos), não se viabiliza economicamente a integração acima referida, e sim, a utilização de uma solução de CPU com ASIC's periféricos.

O próximo problema que se apresenta agora é como otimizar um desenvolvimento desta natureza (CPU de propósito geral + ASIC + software), garantindo o sucesso da implementação, a qualidade do produto final (confiabilidade) e prazos de desenvolvimento compatíveis com a competitividade da indústria (conceito de *time-to-market*). Particularmente a este último aspecto, para sistemas inicialmente inexistentes, deseja-se sobretudo abreviar ao máximo o tempo de desenvolvimento para obtenção de um protótipo do produto final, conceito identificado internacionalmente como Prototipagem Rápida (*Fast Prototyping*) [3] [4].

Nos primórdios da tecnologia para o desenvolvimento de ASIC's, a solução foi a definição e implantação de ambientes integrados de projeto (CAD - *computer aided design*) e de engenharia (CAE - *computer aided engineering*), que num extremo permitem o gerenciamento do projeto - prazos, metas, alocação de recursos, avaliação de custos - e no outro extremo, garantem a consistência de informações deste projeto em suas várias fases - especificação, partição, projeto, integração e testes.

Este panorama sugere que uma abordagem semelhante pode ser realizada para implemetar o projeto conjunto de hw/sw, incorporando também o conceito de prototipagem rápida. Tudo se inicia pela forma que será utilizada para especificar o sistema como um todo. O sucesso de um pacote de ferramentas integradas para *codesign* está diretamente relacionado à disponibilidade de um formato de descrição preliminar do sistema, que permita inicialmente o teste de consistência da especificação, por simulação por exemplo, a partição do sistema e o posterior mapeamento desta descrição em tecnologias disponíveis.

Particularmente, a partição do sistema deve ser o momento mais crítico do projeto, pois está implicitamente considerada a co-existência de hardware e software ainda num formato único e, de alguma forma, por meio de algorítmos adequados, será necessário decidir qual parte será implementada em hardware e qual parte em software. Convém salientar que a escolha da opção por software constitui a solução trivial do problema e a

escolha da opção por hardware caberá somente quando se detetar requisitos de desempenho na especificação do sistema, ou quando outros fatores além de desempenho assim o exigir.

## 4. CONCLUSÕES

Pensa-se em *codesign* como uma nova metologia de projeto de sistemas digitais, que pressupõe como usuário não somente o grande fabricante de computadores e equipamentos relacionados, mas também, e principalmente, o eventual usuário que desenvolve novos produtos configurados sobre sistemas de propósito geral, que detem o conhecimento de alguma especialidade ( instrumentação, aquisição de dados, controle industrial em tempo real, etc ) e observa alguma oportunidade de mercado para produtos altamente competitivos.

Dado o contexto acima, fica claro que é altamente desejável a disponibilidade de um ambiente que permita [3]:

- a especificação em alto nível do sistema alvo;
- a verificação de consistência desta especificação por meio de simulação (nos primeiros momentos da especificação, se possível);
- uma análise de alternativas de partição hw/sw;
- um mapeamento da especificação segundo tecnologias disponíveis para uso;
- o gerenciamento do desenvolvimento de cada partição;
- garantia de transparência entre os desenvovimentos das partes de hardware e de software, que eventualmente venham a ser realizados concorrentemente;
- finalmente, a especificação de testes sobre o protótipo obtido.

A idéia de *codesign* pode ser resumida, em termos da tecnologia atual, como engenharia concorrente de hardware e software, com a opção adicional de que se reserva especial atenção para o momento da partição do sistema, pois neste caso há espaço para questionar se algum software poderia ser melhor implementado em hardware.

Existem atualmente ferramentas de desenvolvimento excelentes em ambos os domínios de hardware e software. Falta uma ferramenta integradora (*framework*) [6] que observe e gerencie os aspectos conjuntos dos dois domínios e realize, parcial ou integralmente, a partição do sistema, sem utilizar a metodologia *ad hoc*. Inicialmente, pode-se conceber tal sistema como baseado em premissas de conhecimento de especialistas na área, nos moldes de um sistema especialista (*expert system*). Entretanto, deve-se procurar métodos formais que permitam o reconhecimento automático de estruturas, dirigindo por fim a opção de implementação.

Para circuitos em hardware, existem linguagens de descrição em estágio de evolução bastante avançado e de uso amplamente aceito pela indústria e comunidade científica. São os casos das linguagens VHDL (*Very High-level Hardware Description Language*) [11] e Verilog [1] [12], que permitem a descrição de circuitos digitais de grande complexidade, em alto nível, de forma independente da tecnologia de implementação, permitindo a simulação desta descrição também em alto nível, com possibilidade posterior de mapeamento em diversas tecnologias de *ASIC's*, tais como *gate arrays*, *standard cells* e dispositivos lógicos programáveis (PLDs), esta última opção particularmente interessante ao cenário brasileiro, devido à flexibilidade e baixo investimento [13]. Interessante notar que o VHDL possui bibliotecas de modelos de componentes comerciais, que inclui a maioria das CPUs existentes, permitindo desta forma compatibilizar o projeto de um módulo de hardware com uma dada CPU escolhida, inclusive a nível de simulação.

Para os módulos em software, não existe o equivalente VHSDL (Very High-level Software Description Language ) [3], mas sim as linguagens normalmente utilizadas para escrever programas (por exemplo, linguagem C). Alguns autores consideram pouco importante esta não-padronização da descrição do software: necessita-se, na verdade, de métodos sistemáticos para gerenciar a diversidade de linguagens, ao invés de se definir estilos unificados de projeto de software.

Aparentemente, está longe o dia de se ter um sistema completo e geral para *codesign*. Tudo indica que esta diretriz será atingida aos poucos, conquistando áreas de aplicação inicialmente específicas e particularizadas. Com a implementação dos primeiros ambientes, a prática indicará os aspectos ineficientes da metodologia que necessitarão ser otimizados.

## REFERÊNCIAS BIBLIOGRÁFICAS

[1] Thomas, E. D.; Adams, J. K. and Schimit, H.; A Model and Methodology for Hardware-Software Codesign. IEEE Design & Test of Computer, september 1993.

[2] Gupta, R. K.; Coelho, C. N. and Micheli, G. De; Program Implementation Schemes for Hardware-Software Systems. Computer, january 1994.

[3] _____, _____. A D & T Roundtable - Hardware/Software Codesign. IEEE Design & Test of Computer, march 1993.

[4] Wolf, W. ; Hardware-Software Codesign. IEEE Design & Test of Computer, september 1993.

[5] Tuck, B.; Technologies Move Toward Hardware/ Software Codesign. Computer Design, june 1992.

[6] Kumar, S.; Aylor J. H.; Johnson, B. W. and Wulf W. A.; A Framework for Hardware/Software Codesign. Computer, december 1993.

[7] Woo, N. S. and Dunlop A. E.; Codesign from Coespecification. Computer, january 1994.

[8] Carro, L.; Burger, R.; Junqueira A. and Suzim A.; Exploring Parallelism in HW/SW Codesign with a new RISC Architecture. Anais do VIII Simpósio de Concepção de Circuitos Integrados, novembro 1994.

[9] Carro, L.; Klauck L. and Suzim A.; HW/SW Parallel Operation in a Codesign Environment. Anais do VIII Simpósio de Concepção de Circuitos Integrados, novembro 1994.

[10] Barros, E.; Lima, M. de and Sampaio, A.; From Hardware/Software Partitioning to Layout Sysnthesis: A Transformational Approach. Anais do VIII Simpósio de Concepção de Circuitos Integrados, novembro 1994.

[11] Armstrong, J. R. ; Chip-level Modeling with VHDL, Prentice Hall, 1989.

[12] _____ , _____ Verilog - Ferramenta proprietária da Cadence Inc (manuais disponíveis).

[13] Behrens, F. H. ; Prototipagem Rápida de Módulos de Hardware para um Ambiente de Projeto Conjunto de HW-SW. Revista do Instituto de Informática - PUCCAMP, vol. 3 nº 2, 1995.

# PROTOTIPAGEM RÁPIDA DE MÓDULOS DE HARDWARE PARA UM AMBIENTE DE PROJETO CONJUNTO DE HW-SW

# FAST PROTOTYPING OF HARDWARE MODULES FOR A HW-SW CODESIGN ENVIRONMENT

**Frank Herman BEHRENS ***


## ABSTRACT

This work discusses the logic capacity specified for commercial Programmable Logic Devices (PLD's), from Altera and Xilinx, related to the gate array alternative. The gate count comparison is done based on some generic circuits and benchmark circuits. Preliminary results indicate a surprising gate count discrepancy for circuits designed in both PLDs and gate arrays. Gate array digital circuits of about 2500 equivalent gates can attain gate counts from 3113 to 5363 gates, roughly 25% to 115% more complexity, for Altera's development package. Considering Xilinx, for the same situation one can obtain estimated gate counts from 3672 to 7938, or 47% till 217% more gates. On the other hand, circuits comming from FPGAs' designs can reduce their gate counts by 60% when implemented on gate arrays. Therefore, one can not estimate a circuit complexity in PLD by simply computing *degree of utilization* versus *component logic capacity*. The transformation of a PLD into a gate array certainly has the benefit of a lower gate count. All these results can be applied to the development of a Hardware-Software Codesign environment, presently under study.


**KEY WORDS**: PLDs, benchmark, hardware design, integrated circuit


## RESUMO

Este trabalho discute a capacidade lógica especificada para Dispositivos Lógicos Programáveis (PLD's), comerciais (EPLDs da Altera Corp. e FPGAs da Xilinx Inc.), em relação a alternativa *gate array*, comparando circuitos lógicos genéricos e circuitos específicos para *benchmark* em termos de sua complexidade, esta expressa em portas lógicas equivalentes. Os resultados preliminares indicam uma surpreendente discrepância entre a contagem de portas lógicas nas metodologias de projeto *gate array* e PLD, inclusive, não havendo uma proporção entre estes valores. Circuitos digitais de cerca de 2500 portas em *gate array* podem atingir, no ambiente Altera, contagens estimadas na faixa de 3113 a 5363 portas equivalentes, correspondendo a um aumento de 25% a 115%. No ambiente Xilinx, obteve-se contagens estimadas na faixa de 3672 a 7938, ou seja, 47% a 217%. Por outro lado, no sentido inverso de implementação, circuitos inicialmente projetados em FPGAs, quando passados para *gate array*, sofrem uma redução da contagem de portas lógicas de cerca de 60%. Portanto, não se pode estimar a complexidade de circuitos projetados em PLDs com base apenas no produto taxa de utilização x capacidade lógica do componente. A transformação de um PLD em *gate array* é certamente beneficiada pela menor contagem de portas lógicas resultante. Estes resultados podem ser utilizados na definição de alternativas de implementação de ASIC's (Application Specific Integrated Circuits), para um sistema de Projeto Conjunto de Hardware-Software, presentemente sob estudo.


**PALAVRAS-CHAVE**: PLDs, benchmark, projeto de hardware, circuito integrado

(*) Professor Titular do Instituto de Informática da Puccamp.

Este trabalho foi inicialmente conduzido no Instituto de Microeletrônica da Fundação Centro Tecnológico para Informática, FCTI, Campinas, São Paulo.

## 1. INTRODUÇÃO

A idéia de prototipagem rápida de circuitos integrados para processamento de dados vem sendo considerada uma alternativa viável para o desenvolvimento de hardware específico, segundo algumas aplicações de uso específico.

Sistemas computacionais são, atualmente, baseados em hardware padrão (computadores pessoais, servidores, estações de trabalho, etc), configurados segundo programas aplicativos (software) desenvolvidos de acordo com alguma aplicação em vista.

Hoje em dia, aparentemente não há condições econômicas para desenvolvimento de sistemas ou partes de sistemas de hardware, resultando na utilização em larga escala de sistemas padrão, que são mais econômicos e mais flexíveis do ponto de vista do usuário. Esta realidade é particularmente verdade na atual engenharia computacional brasileira neste final década. Não se vislumbra facilmente condições econômicas e técnicas de competitividade para sistemas de hardware totalmente nacionais.

Entretanto, surgiu recentemente a idéia de **algorítmos em hardware**, sugerindo que a implementação física de circuitos digitais, que realizem processamento específico, pode levar à otimização de sistemas, quanto a um desempenho e velocidade maiores do que a alternativa usual de implementação totalmente em software.

Esta metodologia de projeto de sistemas tem o nome de **Projeto Conjunto de Hardware e Software** (HW/SW Codesign) [1], [2] e visa sobretudo desenvolver um sistema integrado de especificação de um sistema computacional, que, em algum momento adequado do projeto, possa decidir qual parte do algoritmo será codificada em software e qual parte tem vantagens em ser implementada em hardware [3].

Normalmente, esta implementação toma a forma de um circuito ou conjunto de circuitos integrados de aplicação específica (ASICs - Application Specific Integrated Circuits), que serão eletricamente conectados a algum sistema de hardware padrão, que rodará conjuntamente a parte de software. A parte de hardware funciona, neste caso, analogamente a um co-processador do sistema padrão, acelerando a realização das funções relativas à parte do algorítmo codificada em hardware.

A seguir, discute-se algumas alternativas de implementação de ASICs dentro da idéia de prototipagem rápida.

## 2. METODOLOGIAS PARA ASICs

Até o início dos anos 80, as metodologias *gate array e standard cell* constituiam as formas de implementação de ASICs mais usuais, úteis para prototipagem de CIs ou mesmo para produção em quantidades até 1.000 a 20.000 peças por ano.

Com o surgimento da segunda geração de PLDs, no final dos anos 80, e a sua rápida evolução em capacidade lógica e desempenho, criou-se dúvidas para o usuário quanto a escolha da melhor alternativa para implementação de ASICs, principalmente pela diversidade de arquiteturas e elementos de programação (fusíveis, antifusíveis, células SRAM, EPROM e E2PROM) [4].

Os *gate arrays* atuais atingem capacidades lógicas na faixa de 100K portas equivalentes, existindo, entretanto, famílias que cobrem a faixa de 2K a 20K portas de maneira eficiente em termos de desempenho e custo [5]. Os PLDs mais recentes, segundo os fabricantes, possuem de 10K a 20K portas equivalentes [5] [6]. Porém, a maneira como é estimada sua capacidade lógica é questionável, devido à manipulação lógica de síntese e mapeamento realizada pelos programas de desenvolvimento.

Este trabalho apresenta uma análise comparativa entre uma matriz *gate array* [8] e os componentes PLDs, particularmente os EPLDs da Altera e FPGAs da Xilinx, segundo a sua capacidade de implementação de circuitos digitais. Não se objetiva aqui comparar as famílias de PLDs destes fabricantes entre si, por ser este um assunto já abordado por estudos anteriores, tais como o PREP [9] [10]. Cabe ressaltar que há características distintas entre eles: o ambiente da Xilinx permite a intervenção manual no roteamento de interconexão entre componentes lógicos, o que não ocorre com o da Altera; por outro lado, o ambiente da Altera possibilita o particionamento automático do circuito em vários componentes quando ele não couber em apenas um PLD, o que não é viável com o da Xilinx.

Portanto, este trabalho avalia, sob um ponto de vista prático, o limite de capacidade lógica dos componentes PLDs em relação à alternativa de implementação em *gate array*, sendo que esta última permite uma avaliação mais precisa da complexidade dos circuitos digitais usados como padrão de comparação.

## 3. A CAPACIDADE LÓGICA DE PLDs

A medida da complexidade de circuitos digitais, expressa em portas equivalentes, tornou-se muito utilizada devido à possibilidade de se estimar o tamanho de placas de circuito impresso (PCB), de pastilhas de CIs

standard cells ou mesmo a capacidade lógica de matrizes de transistores pré-difundidas (*gate arrays*).

No caso dos PLDs, independentemente do critério utilizado para o cálculo da sua capacidade lógica máxima, um determinado circuito digital é implementado apresentando uma certa taxa de utilização do componente. O produto capacidade x utilização define, a princípio, a complexidade do circuito implementado. Nas Tabelas 1 e 2 são apresentados dados sobre alguns componentes da Altera e da Xilinx [6] [7], com ênfase na capacidade lógica máxima fornecida pelo respectivo fabricante.

Esta forma de definição de capacidade lógica não se apresenta tão precisa e adequada como no caso das matrizes *gate arrays*. Acredita-se que, devido à realização de manipulações lógicas pelos softwares de programação dos PLDs, é difícil saber, a priori, qual a complexidade resultante de um circuito após a sua compilação num componente PLD, em relação ao seu diagrama esquemático.

**Tabela 1** - Características de PLDs da Altera Corp.

| Família | Componentes | Macrocélulas (= nº flip-flops) | Capacidade em portas equivalentes* |
|---|---|---|---|
| Clássica | EP610 | 16 | 0,6K |
| | EP910 | 24 | 0,9K |
| | EP1810 | 48 | 1,8K |
| Max 5000 | EPM5016 | 16 | 0,6K |
| | EPM5032 | 32 | 1,2K |
| | EPM5064 | 64 | 2,5K |
| | EPM5128 | 128 | 5,0K |
| | EPM5130 | 128 | 5,0K |
| | EPM5192 | 192 | 7,5K |
| Max 7000 | EPM7032 | 32 | 1,2K |
| | EPM7064 | 64 | 2,5K |
| | EPM7096 | 96 | 3,6K |
| | EPM7128 | 128 | 5,0K |
| | EPM7160 | 160 | 6,4K |
| | EPM7192 | 192 | 7,5K |
| | EPM7256 | 256 | 10,0K |

(*) A quantidade de portas equivalentes utilizáveis é de 50%.

**Tabela 2** - Características de PLDs da Xilinx, Inc.

| Família | Componentes | CLBs | Células tri-state | Flip-flops | Capacidade (portas equiv.) |
|---|---|---|---|---|---|
| XC2000 | XC2064 | 64 | - | 122 | 0,8K - 1,0K |
| | XC2018 | 100 | - | 174 | 1,2K - 1,5K |
| XC3000 | XC3020 | 64 | 16 | 256 | 1,3K - 1,8K |
| | XC3030 | 100 | 20 | 360 | 2,0K - 2,7K |
| | XC3042 | 144 | 24 | 480 | 2,5K - 3,7K |
| | XC3064 | 224 | 32 | 688 | 4,0K - 5,5K |
| | XC3090 | 320 | 40 | 928 | 5,0K - 7,5K |

Outras características que dificultam a obtenção da estimativa em portas equivalentes dos PLDs são:

a) **A utilização irregular das células lógicas (CLs)**: os blocos lógicos configuráveis CLBs (Xilinx) ou macrocélulas (Altera), responsáveis pela implementação das funções lógicas existentes nas arquiteturas, podem ser utilizados em toda a lógica disponível, como também apenas implementar uma equação booleana muito simples, desperdiçando o restante da capacidade da CL, sendo também muito difícil manter uma regularidade no fator de utilização destes blocos.

b) **As portas lógicas básicas podem utilizar uma quantidade considerável da capacidade de uma CL**: uma porta OR exclusiva de 4 entradas, por exemplo, que corresponde a uma célula básica em bibliotecas *standard cells* e *gate arrays* (cerca de 2 a 3 portas equivalentes), utiliza toda uma macrocélula quando implementada nos componentes da Altera, devido a sua decomposição na forma de soma-de-produtos.

c) **O uso limitado de flip-flops e células tri-state devido à arquitetura fixa dos PLDs**: além de considerar a capacidade em portas equivalentes, deve-se observar também o número de flip-flops e células *tri-state* exigidos pelo ASIC, pois os componentes PLDs apresentam um número limitado destas estruturas em sua arquitetura.

A par destas considerações, foram feitos estudos para tentar relacionar o critério de cálculo de complexidade de um ambiente *gate array* e dos PLDs. Inicialmente, utilizou-se circuitos funcionais desenvolvidos durante a avaliação funcional do GA2500 [11], e numa segunda fase, elaborou-se circuitos *benchmarks* específicos para este estudo comparativo.

## 4. ANÁLISE INICIAL: CIRCUITOS FUNCIONAIS

A estratégia inicial deste estudo foi aproveitar circuitos já projetados, de complexidade lógica bem conhecida, implementando-os em alguns PLDs (vide Tabela 3). Foram utilizados os três circuitos descritos em [11]:

- **MUL_4x4** - multiplicador de 4 x 4 bits em arquitetura *pipeline*: realiza a multiplicação binária em quatro estágios de processamento (quatro ciclos de relógio);

- **ULA_16** - unidade lógica e aritmética de 16 bits: realiza 6 operações básicas com dois vetores de 16 bits (soma, subtração, AND, OR, OR exclusivo e complemento);

**Tabela 3** - Circuitos funcionais para análise inicial

| Circuito | Complexidade (portas equiv.) | % lógica seqüencial | % lógica combinacional | Flip-flops | Células tri-state |
|---|---|---|---|---|---|
| QUIM | 265 | 80,2 | 19,8 | 47 | 8 |
| MUL_4x4 | 439 | 63,5 | 36,5 | 41 | - |
| ULA_16 | 654 | 19,5 | 80,5 | 32 | - |
| Ouvidor | 1457 | 25,0 | 75,0 | 65 | 32 |

- **Ouvidor** - circuito "observador" de barramentos que auxilia na testabilidade de sistemas digitais através da implementação de equações polinomiais.

O quarto circuito fucional utilizado provém de um projeto realizado a nível de *standard cell* [12]: **QUIM** - parte digital de uma interface para um sistema de aquisição de dados de sensores químicos.

Os resultados da compilação destes circuitos dentro dos componentes da Altera e da Xilinx são apresentados na Tabela 4.

**Tabela 4** - Compilação dos circuitos funcionais nos PLDs.

| Circuitos | Max5000 | Max7000 | Max2000* | Max3000* |
|---|---|---|---|---|
| **QUIM** | EPM5128 (43%) 1075 p.e. | EPM7096 (59%) 1062 p.e. | - | XC3020 (39 CLBs, 47 ff, 8 tri-state) 792 p.e. |
| **MUL_4x4** | EPM5064 (73%) 913 p.e. | EPM7096 (52%) 936 p.e. | XC2064 (43 CLBs, 41 ff) 538 p.e. | XC3020 (41 CLBs, 41 ff) 833 p.e. |
| **ULA_16** | EPM5192 (83%) 3113 p.e. | EPM7160 (92%) 2944 p.e. | XC2018 (80 CLBs, 32 ff) 960 p.e. | XC3030 (72 CLBs, 32 ff, 1440 p.e. |
| **Ouvidor** | EPM5130 (75%) 1875 p.e. | EPM7128 (76%) 1900 p.e. | - | XC3064 (133 CLBs, 65 ff, 32 tri-state) 2375 p.e. |

Obs.: Abreviações - p. e. = portas equivalentes; ff = número de flip-flops utilizados.
(*) A complexidade em portas equivalentes nos PLDs da Xilinx foi avaliada através do número de CLBs utilizadas, tomando-se o melhor valor da faixa apresentada na tabela 2.

Observa-se, primeiramente, que para ambos os fabricantes, Altera e Xilinx, **a utilização da capacidade lógica máxima dos PLDs** empregados não correponde a complexidade dos circuitos definida no ambiente *gate array* (Tabela 3), não se verificando nem mesmo uma proporção entre ambas.

A título de exemplo, para os circuitos QUIM e MUL_4x4, não foi possível utilizar o componente EPM7064, que possui capacidade lógica utilizável estimada de 1250 portas, embora este valor fosse numericamente suficiente. Também, os circuitos ULA_16 e Ouvidor utilizaram mais do que 50% dos CLBs disponíveis nos componentes Xilinx XC3030 e XC3064, respectivamente, o que corresponde a complexidades estimadas superiores às da tabela 3.

Notou-se que o **aproveitamento dos PLDs** ocorre de acordo com as características funcionais dos circuitos (síncrono, assíncrono, combinacional) e com o tipo de portas lógicas utilizadas nos diagramas esquemáticos (funções baseadas em lógica combinacional ou em tri-state). Como exemplo, o circuito ULA_16, que possui uma complexidade bem menor do que o Ouvidor (Tabela 3), teve que utilizar PLDs maiores quando compilado pelo ambiente da Altera. No caso dos circuitos MUL_4x4 e QUIM, de complexidade distinta, tiveram uma utilização bastante semelhante de CLBs pelo ambiente da Xilinx.

Deve-se chamar a atenção, também, quanto às **limitações do número de células tri-state e flip-flops** nestes componentes, particularmente para os componentes Xilinx: os circuitos QUIM e Ouvidor não puderam ser compilados na família XC2000 por esta não possibilitar o uso de células lógicas *tri-state*; a mesma limitação fez com que o Ouvidor tivesse que ser implementado no componente XC3064, embora o XC3042 apresentasse um número suficiente de CLBs para este circuito.

## 5. ANÁLISE COMPLEMENTAR: BENCHMARKS ESPECÍFICOS

O uso de apenas quatros circuitos funcionais não permite uma amostragem significativa para esta comparação de capacidade lógica entre uma matriz *gate array* de 2500 portas lógicas equivalentes e os PLDs da Altera

e da Xilinx. O fato de serem circuitos funcionais em média pequenos e que não cobrem certas aplicações, como máquinas de estados, pode levar a resultados tendenciosos.

A segunda fase deste trabalho consistiu na aplicação dos princípios de comparação implantados pelo PREP (*Programmable Electronics Performance Corporation*). O PREP é uma organização formada pelas empresas com interesse em componentes PLDs e ferramentas de CAD (*Computer Aided Design*) para estes ambientes. Tem por objetivo principal realizar comparações de capacidade e desempenho entre estes componentes e respectivos ambientes de projeto, a fim de esclarecer ao projetista dúvidas quanto à eficiência de arquiteturas e dos elementos de programação (fusíveis, EPROM e semelhantes) [9] [10].

Porém, a comparação realizada pelo PREP não questiona a medida de complexidade dos PLDs através da contagem de portas equivalentes, tal como apresentado na Tabela 4, apesar dos fabricantes fornecerem através desta contagem o tamanho dos seus componentes.

A filosofia de criação dos *benchmarks* pelo PREP é utilizar circuitos específicos, tanto combinacionais quanto síncronos ou mistos, como contadores, máquinas de estado, acumuladores, unidades aritméticas ou mesmo *data paths*. Estes circuitos possuem uma complexidade baixa (entre 100 e 300 portas equivalentes) e são repetidos o maior número de vezes dentro do componente.

Assim, é possivel conhecer a eficiência da utilização dos PLDs, segundo algumas topologias de circuito de interesse, que expressam características operacionais típicas às existentes em aplicações reais.

A estratégia utilizada nesta etapa de estudos foi desenvolver 8 *benchmarks*, segundo as especificações do PREP, e implementá-los numa matriz *gate array*, pelo princípio de repetição, para então definir os circuitos a serem compilados nos PLDs, conforme Tabela 5. É importante enfatizar, novamente, que não está sendo avaliada a capacidade de roteamento de cada ambiente, sendo este um estudo a parte.

Os resultados das implementações dos circuitos bench1 a bench8 (circuitos resultantes da repetição dos *benchmarks*) podem ser vistos na Tabela 6.

**Tabela 5** - Benchmarks específicos para comparação de capacidade lógica

| Benchmarks | Complexidade* (portas equiv.) | Nº de repetições | Circuito resultante (complexidade) |
|---|---|---|---|
| **data path** | 160,0 (56% / 44%) | 15 | bench 1 (2400 p. e.) |
| **timer** | 283,2 (47% / 53%) | 8 | bench2 (2266 p.e.) |
| **máquina de estado (8 bits)** | 94,0 (18% /82%) | 26 | bench3 (2444 p.e.) |
| **máquina de estado (16 bits)** | 205,2 (11% / 89%) | 12 | bench4 (2462 p.e.) |
| **circuito aritmético** | 246,4 (18% / 82%) | 10 | bench5 (2464 p.e.) |
| **acumulador de 16 bits** | 89,6 (100% / 0%) | 27 | bench6 (2419 p.e.) |
| **contador up** | 237,2 (38% / 62%) | 10 | bench7 (2372 p.e.) |
| **contador down** | 225,2 (40% / 60%) | 11 | bench8 (2477 p.e.) |

(*) Os valores percentuais correspondem às porcentagens de lógica seqüencial e combinacional do circuito, respectivamente.

**Tabela 6** - Resultados dos *benchmarks* nos PLDs.

| Circuitos | Max5000* | Max7000* | XC3000* |
|---|---|---|---|
| **bench1** | EPM5128 (72%)<br>EPM5192 (76%)<br>4650 p.e. | EPM7256 (93%)<br>4650 p.e. | XC3090<br>(254 CLBs, 240 ff)<br>3969 p.e. |
| **bench2** | - | EPM7128 (81%)<br>EPM7192 (89%)<br>5363 p.e. | XC3090<br>(235 CLB's, 192 ff)<br>3672 p.e. |
| **bench3** | - | - | XC?<br>(508 CLBs, 78 ff)<br>7938 p.e. |
| **bench4** | - | - | - |
| **bench5** | - | - | XC?<br>(341 CLBs, 80 ff)<br>5328 p.e. |
| **bench6** | - | - | XC3064<br>(219 CLBs, 432 ff)<br>3911 p.e. |
| **bench7** | EPM5192 (83%)<br>3113 p.e. | EPM7192 (85%)<br>3188 p.e. | XC3090<br>(300 CLBs, 160 ff)<br>4688 p.e. |
| **bench8** | EPM5192 (91%)<br>3413 p.e. | EPM7256 (71%)<br>3550 p.e. | XC?<br>(329 CLBs, 176 ff)<br>5141 p.e. |

(*) Em alguns circuitos foi realizado o particionamento automático do ambiente Altera.
(**) O critério para contagem do número de portas equivalentes é o mesmo utilizado na tabela 3.

A análise destes resultados é surpreendente. Poder-se-ia dizer que todos os circuitos *benchs* apresentam uma complexidade aproximada de 2500 portas equivalentes. Nota-se que apenas os circuitos bench1, bench2, bench6 e bench7 puderam ser implementados nos componentes da família XC3000 (XC3090).

Os circuitos bench1, bench2 e bench6 para Max5000 e bench2 e bench6 para Max7000 não couberam em nenhum elemento das respectivas famílias. E os circuitos bench1, bench7 e bench8 couberam apenas nos dois maiores elementos da família Max7000 (EPM7000).

Particularmente, os circuitos bench3, bench4 e bench5 tiveram dificuldades de serem compilados pelo ambiente da Altera, devido a densidade de lógica combinacional; o sintetizador lógico não conseguiu realizar seu trabalho.

Para estes *benchs,* realizou-se a compilação do bloco básico dos *benchmarks* e fez-se uma interpolação da ocupação final em relação ao número de repetições, cujo resultado revelou a impossibilidade de implementação. Problema semelhante foi encontrado na compilação do bench4 dentro do ambiente da Xilinx (para a família XC3000), quando se utiliza o software para plataforma PC.

## 6. CONCLUSÕES

Foi proposta uma metodologia de análise da capacidade lógica de PLDs, em relação a contagem de portas lógicas equivalentes utilizadas em *gate arrays*. Alguns projetos de circuitos digitais, já realizados segundo a metodologia *gate array*, para uma matriz de 2500 portas equivalentes, foram implementados em EPLDs da Altera e FPGAs da Xilinx, tendo sido comparada a utilização dos componentes PLDs e a complexidade aparentes destes circuitos.

Os resultados preliminares indicam uma surpreendente discrepância entre a contagem de portas lógicas nas metodologias de projeto *gate array* e PLD, inclusive, não havendo uma proporção entre estes valores.

Circuitos digitais de cerca de 2500 portas em *gate array* podem atingir no ambiente Altera contagens estimadas na faixa de 3113 a 5363 portas equivalentes, correspondendo a um aumento de 25% a 115%. No ambiente Xilinx, obteve-se contagens estimadas na faixa de 3672 a 7938, ou seja, 47% a 217%.

Como trabalhos futuros, pretende-se estender esta análise para comparações de desempenho elétrico, estudos sobre a influência da intervenção do projetista sobre a alocação de CLs e roteamento de interconexões, durante a fase de compilação do circuito.

Outro aspecto importante diz respeito ao caminho inverso de implementação, partindo-se de circuitos em PLDs para sua integração em *gate arrays*. Resultados preliminares, obtidos da análise de circuitos desenvolvidos pelo CPqD/TELEBRÁS para o sitema Trópico, que foram descritos em VHDL e prototipados em FPGAs da Xilinx, são reportados na Tabela 7 abaixo [13] [14].

Claramente, observa-se que a complexidade de tais circuitos, quando implementados num gate array de 2500 portas lógicas equivalentes, sofre uma redução de cerca de 60%. Tais resultados serão mais profundamente analisados em trabalhos futuros.

**Tabela 7** - Comparação de circuitos inicialmente implementados em PLDs.

| Circuito | Complexidade estimada XC3042 (portas equivalentes) | Complexidade de um gate array de 2500 (portas equivalentes) | % de redução |
|---|---|---|---|
| TB45 | 3267 | 1298 | 60,3 |
| TB46 | 2780 | 1190 | 57,2 |

Portanto, conclui-se que, do ponto de vista da engenharia de sistemas, implementar ASICs por meio de *gate arrays* ou PLDs resulta em avaliações de complexidade totalmente distintas e não coerentes entre si. Um sistema cuja complexidade seja avaliada por sua implementação em ambiente FPGA, por exemplo, pode ser mais facilmente realizável em *gate array*, ou mesmo *standard cell*, com benefícios óbvios de custo e desempenho.

Por fim, dado o contexto do Projeto Conjunto de Hardware e Software, tem-se uma avaliação preliminar das ferramentas de implementação de ASICs para prototipagem rápida, as quais serão consideradas como alternativas para a definição de um sistema de projeto HW/SW, ora em fase de investigação.

## AGRADECIMENTOS

*O autor agradece às empresas Hicad Sistemas Ltda e União Digital Com. Rep. Ltda, em especial a Vanderlei Perez Sanches e José Carlos Stagni, pelo empréstimo dos softwares de PLDs, e à colaboração dos engenheiros do CPqD/TELEBRÁS Luiz M. A. Madureira, Carlos G. Kruger e Alexandre Soares Pires, que tornaram possível a realização deste trabalho. Também ao engenheiro Renato Peres Ribas, que conduziu substancial parte dos trabalhos a título de tese de Mestado, sob suporte do Programa RHAE. Agradecemos à Fundação Centro Tecnológica para Informática, FCTI, onde as trabalhos preliminares foram conduzidos, e pelo suporte de laboratórios e infraestrutura, sem os quais não seriam atingidos os objetivos relatados.*

## REFERÊNCIAS BIBLIOGRÁFICAS

[1] Hardware/Software codesin, D & T Roundtable, IEEE Design & Test of Computers, pp. 83-91, Março, 1993.

[2] D. E. THOMAS, J. K. ADAMS, H. SCHMIT, A model and methodology for hardware-software codesign, IEEE Design & Test of Computers, pp. 6-15, Setembro, 1993.

[3] R. K. GUPTA, Program implementation schemes for hardware-software systems, IEEE Computer, pp. 48-55, Janeiro, 1994.

[4] REIS, A. I., GÜNTZEL, J. L. & RIBAS, R. P. Algumas formas de implementação de ASICs. Anais do VII Simpósio Brasileiro de Concepção de Circuitos Integrados, pp. 15-34, Rio de Janeiro - RJ, Setembro, 1992.

[5] Functional flexibility keeps ASICs in demand. Journal of Electronic Engineering - JEE. January, 1993. Vol. 30, Nº 313, Dempa Publications. pp. 66-69.

[6] Data Book da Altera Corp., 1993. San Jose, Califórnia.

[7] The Programmable Logic Data Book, 1993. Xilinx, Inc. San Jose, Califórnia.

[8] SIMÕES, S. A.; CHAVEZ, F.; RIBAS, R. P.; FINCO, S.; CUIN, M. & BEHRENS, F. H. Matriz Gate Array CMOS avançada, configurável por um único nível de metal. Anais do VII Congresso da SBMicro, pp. 281-291, São Paulo - SP, Julho, 1992.

[9] A Systems Perspective on Inter-preting the PREP Benchmarks. Documento de divulgação da Xilinx, Inc.

[10] PREP Benchmarks For Pro-grammable Logic Devices (White Paper). Documento de divulgação da Altera Corp.

[11] Estruturas e Circuitos de Teste do Projeto GA2500. Documento Interno 011/93 LPCI-IM-CTI. 1993.

[12] CHAVEZ, F.; FINCO, S.; ROHWEDDER, J. J. R. & PASQUINI, C. Integrated multichannel voltammetric detection system: design description. Anais do VIII Congresso da SBMicro, pp. XII. 34-36, Campinas - SP, Setembro, 1993.

[13] Documentação de Projeto do TB45. Documento interno do CPqD / TELEBRÁS.

[14] Documentação de Projeto do TB46. Documento interno do CPqD / TELEBRÁS.

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

## NORMAS AOS COLABORADORES

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

2 - Os artigos deverão conter, sequencialmente:
- Título em português;
- Título em inglês;
- Nome(s) do(s) autor(es) (colocados por extenso, o último nome maiúsculo, e seguidos de*, para especificações profissionais do(s) autor(es) no rodapé da primeira página);
- Abstract (máximo de 200 palavras);
- Resumo;
- Introdução (precedida do número 1);
- Corpo do artigo (itens numerados sequencialmente a partir do número 2);
- Conclusões;
- Referências Bibliográficas (conforme utilizado nesta edição);

Desenhos, gráficos e fotografias serão denominados Figuras, numerados sequencialmente (algarismos arábicos) e constantes do corpo do trabalho. As tabelas serão denominadas Tabelas, numeradas sequencialmente (algarismos arábicos) e constantes do corpo do trabalho;

3 - Os trabalhos, digitados com, no máximo, 30 000 caracteres (aproximadamente dez páginas), deverão ser elaborados em WinWord 2.0, WinWord 6.0 ou DOS Word 5.0 (acompanhados de folha de estilos);

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

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

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

7 - Quaisquer outros esclarecimentos poderão ser feitos pelo Conselho Consultivo da Revista;