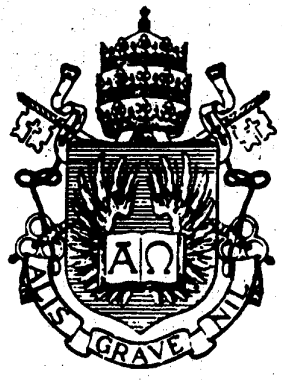


PUC

Série : Monografias em Ciência da Computação
Nº 20/77

PESQUISAS EM ANDAMENTO EM CIÊNCIA DA COMPUTAÇÃO

editado por

ROSANE TELES LINS CASTILHO

- Anais da 1ª Conferência do Departamento de Informática, 9-11 de setembro de 1977, Mendes, R.J.

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 – ZC-19
Rio de Janeiro – Brasil

Série: Monografias em Ciência da Computação

Nº 20/77

Editor da Série: Michael F. Challis Outubro, 1977

"PESQUISAS EM ANDAMENTO EM CIÊNCIA DA COMPUTAÇÃO"

editado por

ROSANE TELES LINS CASTILHO

- Anais da 1ª Conferência do Departamento de Informática, 9-11 de setembro de 1977, Mendes, R-J.
- Patrocínio: PUC/RJ
- Organização: Profa. Therezinha da C. F. Chaves e Prof. Carlos J. Lucena.

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Para obter cópias dirija-se a:

Rosane T.L. Castilho
Chefe, Setor de Documentação e Informação
Deptº Informática - PUC/RJ.
Rua Marquês de São Vicente, 209 - Gávea
20.000 - Rio de Janeiro - RJ - Brasil.

PREFÁCIO

Estes Anais foram baseados nos trabalhos apresentados na 1ª. Conferência do Deptº de Informática, realizada em Mendes, RJ, de 9 a 11 de setembro de 1977.

A Conferência objetivou a divulgação, através de relatos e discussões, de pesquisas ora realizadas por professores e alunos do Departamento de Informática, e, como sub-produto , o intercâmbio de informações e idéias, num ambiente propício ao congrassamento comunitário.

Foram tantos os resultados positivos desse encontro que prevê-se para o próximo ano e anos subsequentes novas realizações do evento.

Cabe aqui agradecer à Pontifícia Universidade Católica do Rio de Janeiro pelo patrocínio e inestimável apoio dado à realização da Conferência.

Carlos José P. de Lucena
Diretor do Deptº de Informática

SUMÁRIO

Métodos Cíclicos para o Tratamento Numérico das Equações Diferenciais.

Prof. Peter Albrecht

A introdução de métodos cíclicos por Donelson e Hansen em 1970 representa o aspecto mais interessante na área do tratamento numérico das equações diferenciais nos últimos anos.

A idéia é simples. Ao invés de usar uma fórmula só para determinar as aproximações $y^*(x_j)$ da solução y da equação diferencial.

$y' = f(x,y)$; $y(a)=\eta_o \in R^n$; $f:[a,b]\times R^n \to R^n$, por exemplo a fórmula:

(1) $y_j^* = y_{j-1}^* + \frac{h}{12}(5\, f_j^* + 8\, f_{j-1}^* - f_{j-2}^*)$; $y_o^* = \eta_o$; $y_1^* = \eta_1(h)$

aplica-se M fórmulas diferentes numa maneira cíclica; para M=2 por exemplo

(2a) $y^*_{2_j} = y^*_{2_{j-1}} + \frac{h}{12}(5f^*_{2_j} + 8f^*_{2_{j-1}} - f^*_{2_{j-2}})$

(2b) $y^*_{2_{j+1}} = y^*_{2_{j-1}} + \frac{h}{3}(f^*_{2_{j+1}} + 4f^*_{2_j} + f^*_{2_{j-1}})$

onde a fórmula (1) é aplicada só para as aproximações com índice par e (2b) para tais de índice ímpar.

Tais métodos cíclicos podem ter ordens mais elevadas do que os métodos de passo k e também melhores características de estabilidade.

Uma teoria completa para métodos deste tipo foi desenvolvida; estão, entretanto, ainda em aberto vários problemas, tais como:

1. existência de tais métodos com ordem máxima.
2. teoria completa sobre as áreas de estabilidade delas.
3. existência de métodos "stiff-estáveis" com ordem alta.
4. relacionamento aos métodos de RUNGE-KUTTA

que podem ser tópicos de teses de mestrado e doutoramento.

The Use of Program Transforms in Program Development.

Prof. Jacques J. Arsac (*)

1. The process of programming generally runs along the following line

speculations → specifications → algorithm → program.

Speculations describe the problem to be solved in a non-formal way. Nothing can be said about their replacement by formal specifications.

In some sense, the program is a representation of the algorithm which solves the problem stated by specifications. There is some redundancy in considering first the algorithm, then the program.

As a mather of fact , it is frequently possible to reach directly a program from the specifications. But the program is a constrained form of the algorithm: it must fit a computer characteristics, as expressed by a programming language. Two of them are very important.

- There is a finite amount of memory
- A program is an ordered set of statements.

Assignment statement is a consequence of the first constraint, and ordering is needed as soon as assigment is used.

u: = u + v, v: = L* u-v; Z: = u * v

is not equivalent to

v: = 2 * u-v; u: = u + v; z: = u * v.

Using an assignment free language is a good way to define an algorithm without entering into programming difficulties. Recursion is another way to do so.

2. Whatever the way in which a program has been obtained, it will very likely have to be modified: maybe for greater efficiency (as it is too frequently pointed out in literature),

(*) Institut de Programation, Univ. Paris VI.

but certainly for greater clarity, simplicity, readability. The normal programming scheme is:

1) - Write a correct program
2) - Improve it, mainly in order to simplify it and in some cases for greater efficiency.

The idea of a methodology based on program transforms has been more or less advocated for the first time by Knuth (Structured programming with go to statements, ACM surveys, Dec. 1974), then developed by Suzan Yerhardt, Loveman, Standish, Burstall, Donlington.

I have considered this method, looking for a set of primitive transforms which should be easily remembered and worked by hand. Their use must be safe, so that a program is valid because its first form has been proved valid, and only valid transforms have been made on it.

Synthetic transforms are defined as those transforms which do not change the history of computation. They do not depend on program meaning or program property , and hold even if there are side effects in procedures (for instance...) There are 5 primitive transforms which form a complete catalog (Guy Cousincan, thése d'Etat, Paris 1977) .

Local semantic transform depend only on local program properties. For instance.

IF t THEN IF t THEN j ELSE g FI
ELSE h FI =>
IF t THEN j ELSE h FI

if there is no side effect in t (successive computations of t give the same result)

A(I) : = 0; A(J) : = 1 →
A(J) : = 1; A(I) : = 0
if I ≠ J.

Having made a lot of experiments, we are convinced that a small number of very obvious local semantic transforms

are needed for practical use.

We have developped in PUC an interactive system for program manipulation. The program is entered at the terminal in RE∞ control structures (loops are represented by DO ... EXIT(t) ... OD, EXIT(t) meaning: go out of t loops). Then the user indicates which transform has to be done, and on what point of the program. Various kinds of transforms may be asked for with various levels of security:

syntactic transforms, entirely checked by the system.

program equations manipulation, which is another way to perform syntactic transforms, also entirely checked.

local semantic transforms partly checked. Their validity rely definitlly on proper use, even if in some cases the system detects an impossibility.

Editing facilities without any check.

Recursion removal. Works on parameter recursive procedures and is very safe, if properly used.

There are 52 transforms implemented. The system has been used to drive more or less sophisticated ransforms. The experiment has not been performed for a long enough period of time so that conclusions can be given about the use of these tools in programming. The only thing which can be asserted is that it eliminates completely clerical errors or inadvertances in the process of program development. In at least one case, a program has been derived which whould probably not have been constructed directly by hand, due to the amount of transformations involved.

But this cannot be taken as a definitive conclusion.

Análise de Sistemas - uma Abordagem Sistêmica (proposta de tese de mestrado)

Solange de L. Asteggiano

1 - Descrição Sumária do Problema

A análise de sistemas compreende a coleta, organização e avaliação de fatos sobre um sistema do mundo real (Sistemas Objeto), com a finalidade de determinar, de forma completa e precisa, os requisitos do sistema de informação a ser projetado para servir a este sistema objeto. Entendendo-se por requisitos as necessidades ou usos de informações no Sistema Objeto.

Um fato largamente observado é que a análise de sistemas tem, inúmeras vezes, apresentado um alto índice de erros na determinação de requisitos, comprometendo desta maneira o desempenho do sistema de informação resultante , em muitos casos de forma irremediável.

Este problema assume proporções mais alarmantes quando observamos que suas consequências não se restringem apenas ao desempenho do sistema de informação mas afetam também os custos dos mesmos. Segundo estudo apresentado por B.W. Boehm, na Segunda Conferência Internacional sobre Engenharia de Software, 12% dos custos de desenvolvimento ocorrem na fase de projeto, 6% na fase de implementação, 12% nos testes e 70% na manutenção, sendo que a maior parte desta manutenção é causada pela má determinação dos requisitos do sistema.

Estes fatos demonstram a importância de procurarmos substituir os métodos heurísticos, que caracterizam atualmente o processo de análise de sistemas, por métodos sistemáticos , alicerçados em uma sólida fundamentação conceitual. Esta é a motivação maior do presente trabalho.

2 - Objetivo da Tese

- Apresentar corpo de conceitos sobre sistemas de informação, visando fornecer um arcabouço teórico para o desen

volvimento do processo de análise de sistemas, em bases menos empíricas.

Esses conceitos serão fundamentados, basicamente, em :

* Teoria Geral de Sistemas
* Teoria de Sistemas de Informação, de Langefors
* Modelo de informação e conceito de linguagem, apresentados por J. Arsac em seu livro "LA SCIENCE INFORMATIQUE".

- Com base nos conceitos formulados e no enfoque de Langefors sobre análise de sistemas:

* determinar uma sequência de passos a ser seguida no processo de análise de sistemas (visando sistematizá-lo).
* definir uma maneira sistemática de representar o produto dos passos identificados no processo de análise de sistemas.

3 - Contribuição Esperada

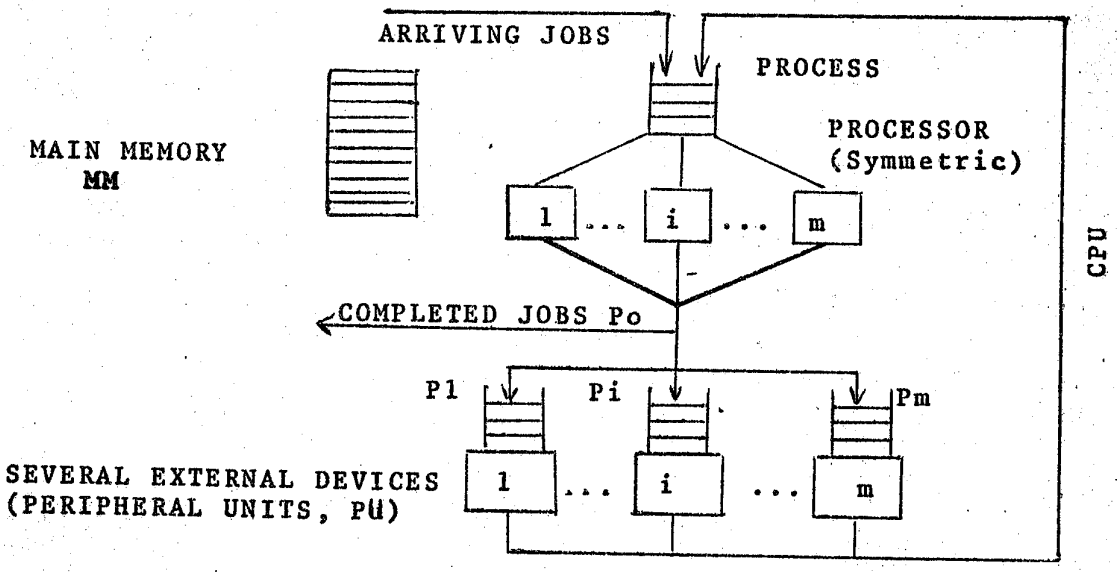
- através dos conceitos apresentados propiciar uma melhor visualização dos problemas habituais de análise de sistemas.

- através do método e da sistemática apresentados contribuir para uma maior sistematização do processo de análise de sistemas.

An Analytical Model of Symmetric Multi-Processor-Systems.

Prof. Gunter Bolch

1. The System



A job is processed in this system as follows:
As soon as there is enough space in the MM an arriving job enters the MM and joins the queue in front of the CPU's. After the processing in one of the CPU's the job leaves the system (and the MM) with the probability Po or needs a PU with the probability Pi. After being serviced in one of the PU's the job joins again the processor queue etc., until it leaves the system and the MM.

We need to know:

1. transition probabilities Pi
2. Capacity of the MM.
3. Distribution of the service times in the CPU's and the PU's.
4. Distribution of the program length of a job
5. Mean of the whole service time of a job.

We are interested in:

1. Utilization of the CPU's and the PU's.
2. System time of a job.
3. Queue length and waiting times
4. Through put

2. Model of the Main Memory.

Using the capacity of the MM and the distribution of the program length we are able to calculate the probability Qn of leaving n jobs in the MM. Each job in the MM is either being processed in one of the CPU's or is waiting in a queue.

3. Exact Model

If we assume that the number of jobs in the system is constant N we can use the well known method of Gordon / Newell [1] to calculate our interesting variables, which are conditioned on N and therefore must be unconditioned using the probability Qn of chapter 2. But the expense is very important [4]. Therefore we tried to find an approximate model.

4. An Approximate Model.

To find an approximate model we assume that the rate of arriving customers at the processor queue is known and then we are able to use the Jackson method [2], which is much less expensive than the Gordon / Newell method if we make some approximations. It can be shown by comparing with simulation results and the exact model that these approximations are close to the exact values.

5. Extension to Jobs with two Priority Classes

The model can be extended to jobs with two priority classes and we get all interesting variables for these two classes separately. To get our extended model we need the results of Coffman [3] and an extension of the conservation low.

References

1. Gordon and Newell: Closed Queueing Systems with Exponential Serves, Oper. Research , 15 (1967) p. 254-65

2. Jackson: Nekworks of Waiting Series, Oper. Research, 5 (1967) p. 518-21

3. Coffman:Priority Assignment in Waiting Line Problems, Oper. Research, 2 and 3 p. 70-76 and p 547.

4. Brartsch and Bolch: Ein analytisches Modell für symmetrische mehrprocessorcenlagen, Informatik-Fachberichte 9 Springerverlag S. 93-108.

The JACKDAW Database Project

Prof. Michael F. Challis

Status

JACKDAW is a database system already implemented on the 370/165 at PUC-RJ. The system includes the following components:

Nucleus - a library of interface procedures which are available to BCPL applications programs.

DBREFORM- for defining a new database or modifying an existing one.

SLANG - a low-level command interface (most commands are in 1-1 correspondence with the interface procedures of the nucleus).

ODIUM - an enquiry/update language designed for interactive use.

DBLIST - a report generator.

Overview

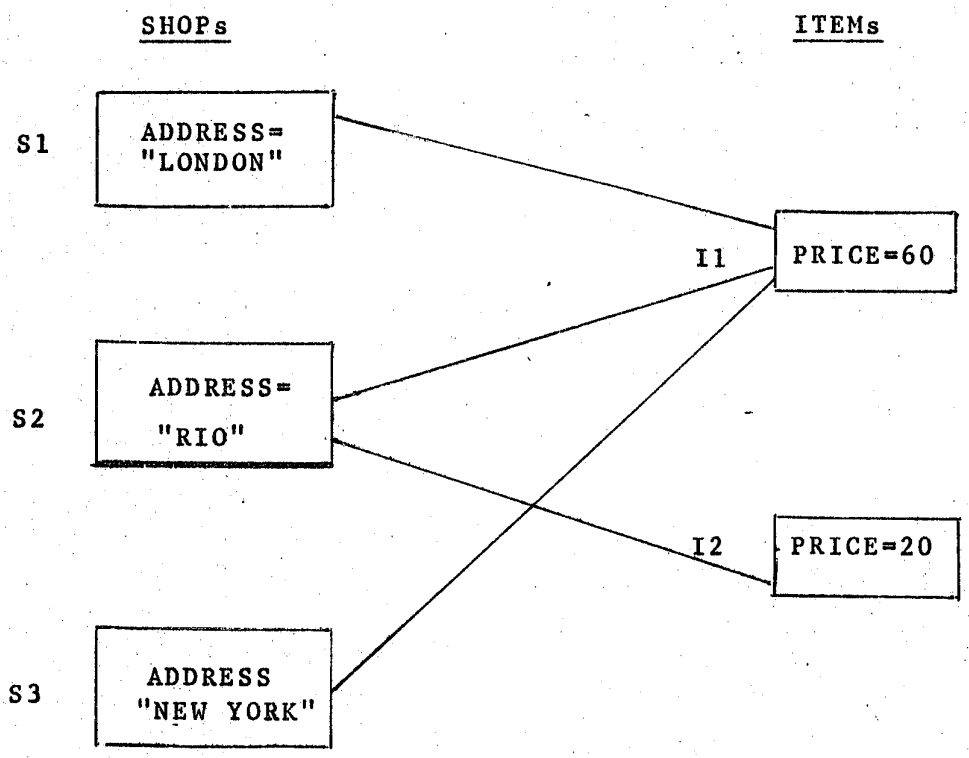
The structure of a database is defined in terms of the "classes" of information it contains, and the "links" between these classes.

eg.
```
NEW CLASS SHOP
BEGIN
   STRING ADDRESS
END

NEW CLASS ITEM
BEGIN
   INT PRICE
END
ADD LINK (STOCK, SHOPS) FROM SHOP TO ITEM
```

An actual database with this structure might contain,

say, 3 SHOP entries and 2 ITEM entries:



Note that each entry in a class is uniquely known by its identifier (S1,I2 etc).

All shops sell item I1; only the shop in Rio sells item I2.

All links are automatically bi-directional.

To illustrate a few of the ways by which data may be accessed and modified, here are some examples of commands in the ODIUM language;

Command	Effect
TYPE SHOP S1	Types information about the SHOP entry S1. eg. SHOP S1 ADDRESS = "LONDON" STOCK= I1
TYPE SHOP S2 STOCK	Types STOCK of S2: STOCK = I1 I2
ITEM I1 PRICE=90	updates the price of item I1
SHOP S3 ADD STOCK I2	Adds a new link between S3 and I2
ITEM I2 ADD SHOPS S3	Exactly the same effect as as the previous command.

Current areas of research

1) Extension of the structural facilities of the package.
   Current ideas include:

   i) "Group" primitive fields.

```
eg      NEW CLASS ROOM
        BEGIN
           INT AREA
             ...
           GROUP OCCUPANTS
           BEGIN
               STRING NAME
               BOOL   STUDENT
           END
           ...
        END
```

   The items in a group may be repeated within an entry. The differences between the properties of group and link fields are interesting.

   ii) Classes with "alternatives" (cf "unions" in Algol 68) .

```
eg.  NEW CLASS PERSON
     BEGIN
        STRING NAME
            ...
        ONEOF (
           PROFESSOR BEGIN STRING SUBJECT; ... END
           STUDENT BEGIN INT AGE ; ... END
           SECRETARY BEGIN INT TELEPHONE; ... END
              )
     END
```

Links may be defined to PERSONS (ie. from a ROOM entry), or just to, say, STUDENTS (ie. from a COURSE entry).

We may also have links defined between the alternatives:

eg ADD LINK (SEC,PROFS) FROM PROFESSOR TO SECRETARY.

2) Design of a general-purpose maintenance language to replace SLANG, ODIUM and DBLIST.

Examples:

i) (at lowest level):

```
.E1  ←  SHOP S1
 ADDRESS OF .E1 := "LONDRES"
 RELEASE .E1
```

ii) (at higher level):

```
TYPE STOCK FOLLOW WHERE PRICE < 100
     OF SHOPS (S1, S2, S3)
```

3) Problems of concurrent access by many users.

The principal idea here is that of maintaining several similar versions of a database within a single datafile.

For example, a read-only program which wishes to be unaffected by changes made by other concurrent update programs may operate on a 'copy' of the database. This copy is created when the program is entered and released when the program

terminates.

The 'creation' and 'release' of such copies is economical and efficient in the implementation of Jackdaw; for more details see [2].

[1] Challis, M.F., 1974. "The JACKDAW Database Package", TR1, Computer Laboratory, University of Cambridge, Cambridge, England.

[2] Challis, M.F., 1977. "Integrity techniques in the JACKDAW Database Package", Monografia em Ciência da Computação nº 9/77, PUC-RJ, Rio.

Situação da Pesquisa em Bancos de Dados

Prof. A. L. Furtado

São observadas atualmente duas tendências aparentemente conflitantes, na orientação da pesquisa em bancos de dados:

- permitir ao usuário grande liberdade na escolha do modelo sob o qual verá o banco de dados e dos métodos de acesso e manipulação;
- padronizar os sistemas de bancos de dados, sendo a mais conhecida a tentativa do grupo ANSI/X3/SPARC nos Estados Unidos, mas havendo outros esforços importantes em diversos países.

Na verdade procura-se conciliar os dois objetivos. Quanto ao primeiro, reconhece-se não haver nada mais esterilizante do que a adoção (ou pior ainda, a imposição) de um modelo inadequado às condições e interesses de cada grupo específico.

Novos modelos vem sendo propostos e mesmo os três modelos básicos - hierárquico, de redes e relacional - tem evoluído drasticamente. No modelo de redes as estruturas de bancos de dados, mostradas por diagramas, revelam a introdução de uma série de modificações desde sua forma gráfica até a seleção, dado o novo conceito de papéis desempenhados qualificando as entidades e condicionando os mecanismos de acesso.

Na implantação do modelo relacional procura-se oferecer ao usuário numerosas estruturas de acesso, tais como elos e inversões, e se reconhece que o sistema, para cada tipo de informação, levando em conta a eficiência, deve procurar seu próprio caminho de inserção.

Outra fraqueza do modelo relacional vem merecendo atenção: os aspectos semânticos. Por exemplo , o conceito puramente matemático de relações é considerado insuficiente, tendo-se feito a distinção entre características e asociações; por representarem interdependências entre entidades concluiu-se que as associações devem ser analisadas com muito cuidado, tanto mais que à interdependência vem agregar-se uma outra característica

fundamental da técnica: a variedade. Agregações e generalizações representam tentativas de definir relações como tipos de dados.

Na padronização ANSI/X3/SPARC ressalta a distinção entre os esquemas conceitual (a cargo do administrador empresarial), interno (administrador do banco de dados) e externos (diversos administradores de aplicações). Enquanto de um ponto de vista estático a distinção entre os esquemas parece razoavelmente clara, a questão mais atual formulada pelos pesquisadores é: Funcionamento do esquema? dados os inúmeros problemas de conciliar a atuação dos diversos usuários através de seus esquemas externos, que depende de um jogo de interesses a ser disciplinado pelo administrador empresarial.

Identificação de Atividades Típicas do Desenvolvimento de Sistemas de Informação (proposta de tese de mestrado)

Maria Cristina L. Gomes.

1 - Situação do Trabalho

O trabalho em desenvolvimento situa-se na área de sistemas.

Uma definição de sistemas é a de um conjunto de partes coordenadas, que concorrem para a realização de um conjunto de objetivos.

Um sistema de informação é qualquer sistema usado para prover informação, qualquer que seja sua finalidade.

Entende-se então sistema de informação como um grupo de pessoas, um conjunto de documentos, equipamentos para processamento de dados, que reduzam decisões ambiguas e que forneçam dados precisos no mais eficiente espaço de tempo e no mais rentável custo.

2 - Motivação do trabalho

Durante o desenvolvimento de um projeto de sistema de informação, são efetuadas diversas atividades. Até então os objetivos de cada atividade eram atendidos isoladamente, não se dando maior importância às duplicidades de processos e dados.

Motivados pela tecnologia e mão de obra especializada, a tendência atual é a de desenvolver sistemas integrados. Portanto as atividades de um projeto de sistema de informação foram grupadas em fases, constituindo cada uma, uma etapa do ciclo de vida do sistema de informação.

A literatura só faz referência às fases que compõem o desenvolvimento de um sistema de informação. Cada autor adota seu ponto de vista, arbitrando os objetivos de cada fase, desenvolvendo e criando atividades concordantes com estes objetivos.

Se analisarmos os"objetivos reais" de um sistema, nem sempre são atendidos pelos resultados apresentados. Requer cautela , a definição destes "objetivos declarados" e as atividades a eles associadas durante o projeto de desenvolvimento.

Nossa expectativa é de que possamos, através de um estudo nesta área, redescobrir as atividades comuns, indispensáveis ao sucesso de um desenvolvimento de sistema de informação .

3 - Elaboração do Trabalho

Será feita uma análise da literatura existente, procurando identificar as atividades específicas do, desenvolvimento de sistemas de informação.

Reteremos maior atenção às atividades relativas a um sistema de processamento de dados, sistema este responsável pelo armazenamento, processamento e recuperação das informações necessárias ao funcionamento do sistema usando equipamento de processamento de dados.

Todo sistema de informação tem um ciclo de vida bem definido, isto é, todos são "concebidos", "desenvolvidos" e "operacionalizados", cabendo ao grupo de projetistas decompô-lo em etapas que reunam atividades afins.

Será também objeto de estudo este "desmembramento", fazendo uma crítica quanto a validade dos resultados obtidos durante a execução das diversas atividades propostas na literatura.

Partiremos para a elaboração deste trabalho, da versaão do fluxo de desenvolvimento apresentado por Neuhold o qual se baseia em um sistema de processamento de dados cujo ciclo de vida é decomposto nas seguintes fases como mostra figura a seguir.

Novos Resultados em "Hashing"

Prof. Gaston H. Gonnet

Nesta palestra foram apresentados vários resultados relativos aos algoritmos de pesquisa por "hashing" que são resumidos na tabela abaixo. Os trabalhos que contêm as derivações dos resultados por extenso são:

Gonnet & Munro, "The Analysis of an Improved Hashing Technique".

Gonnet, "Expected Maximum Probesequence in Hash code Searching".

Gonnet, "Interpolation and Interpolation-Hash Searching".

(c) - resultados conjeturados

(e) - resultados experimentados obtidos por simulação

Algorithm	Average Number of Accesses		Average Longest Probe Sequence	
	Full table	$\alpha = n/m$	Full table	$\alpha = n/m$
Open Address. (uniform probing) [Knu 6.4.D]	$ln(m) - 1 + \gamma + o(1)$	$-\alpha^{-1}\, ln(1-\alpha)$	$0.63158\ldots \times m + O(1)$	$-log(m) + O(log_\alpha(-log_\alpha(m)))$
Brent's reordering scheme [Bre]	$2.49\ \ldots$	*complicated*	$O(\sqrt{m})$ (c)	?
Gonnet & Munro reordering scheme [Gon 2]	$2.13\ \ldots$	*complicated*	$O(ln\ m)$ (c)	?
Optimal reordering to minimize average [Gon 2]	$\geq 1.668\ldots$ [Gon 1] ~ 1.83 (e)	$\geq 2 - \frac{1-e^{-\alpha}}{\alpha}$	$O(ln\ m)$ (e)	?
Optimal reor. to minimize worst case [Gon 2]	~ 1.83 (e)	?	$ln(m) + 1.077\ldots o(1)$	$\lceil -\alpha^{-1} ln(1-\alpha) \rceil$
Separate chaining [Knu 6.4]	1.5	$1 + \alpha/2$	$\sim \Gamma^{-1}(m)$	$\sim \Gamma^{-1}(m)$

Macro e Micro Programação de um Monitor de Entrada e Saída (proposta de tese de mestrado).

Isis D. Larangeira

1 - Introdução

Pretende-se apresentar aqui, a arquitetura do MICRO-02, microcomputador projetado e desenvolvido pelo Departamento de Engenharia Elétrica da PUC/RJ, bem como alguns aspectos do software e firmware que estão sendo desenvolvidos.

2 - Configuração do Sistema Micro-02

- Unidade Central de Processamento, INTEL 8008
- Memória Principal com 4K bytes.
- Processador de Entrada/Saída composto de:
    - Memória de Controle, tipo ler somente, com 1K palavras de 16 bits.
    - Circuitos de Controle que geram e distribuem sinais de controle para os demais componentes do sistema e para os periféricos.
- Dispositivos periféricos:
    - Teletipo
    - Unidade transportadora de fita cassete
- Registros de Endereçamento à Memória Principal (REME e REND) que podem ser carregados pelo Processador de E/S ou pela UCP, quando um desses dois processadores deseja fazer acesso a memória.
- Registro de Endereçamento a Periféricos (REP) através do qual a microprogramação seleciona o periférico com o qual deseja se comunicar. O registro está ligado à Via de Endereços, à qual estão acoplados os periféricos.
- Registros Auxiliares (RAUX e CAUX), acessíveis pela micro programação utilizados para manter informações temporariamente.

- Painel, controlado pela microprogramação, através do qual pode-se realizar várias funções no microprocessador (ex: obter e modificar informações na Memória Principal, realizar o IPL no sistema, causar interrupções na UCP)
- Via Principal (bidirecional) pela qual se faz a troca de informações entre os componentes do sistema e entre o microcomputador e os periféricos.

3 - Compartilhamento da Via Principal

O MICRO-02 é um sistema com multiprocessamento onde não somente há compartilhamento da Memória Principal, como da Via Principal de Comunicação.

A micro-programação usa a Via Principal sem limitações e, quando não se faz mais necessário o acesso à Via permite, explicitamente, o seu uso pela UCP. A micro-programação pode retomar o controle da Via, quando a UCP entrar no estado de WAIT. Este estado é decorrente de uma interrupção ou, da execução na UCP, de uma instrução de E/S.

Através da microprogramação pode-se parar a UCP num acesso à Memória Principal, e suprir no lugar da memória, informações à UCP.

A execução de uma micro-instrução é da ordem de 15 vezes mais rápida que a execução de uma instrução na UCP. A UCP irá sentir pouco a interferência do Processador de ELS se as micro-rotinas não forem muito grandes.

4 - Um Monitor de ELS para o Micro-02

O Monitor que se pretende desenvolver tem como objetivo possibilitar a transferência de blocos de informação entre o microcomputador e os periféricos, ao invés de uma transferência caráter a caráter, de modo que o usuário do sistema interaja o mínimo possível com os aspectos físicos que envolvem uma operação de entrada/saída.

O Monitor será constituído de um Macro-Monitor que executará na UCP e de um Micro-Monitor que executará no Processador de ELS. O problema básico que surge é: como dividir as tarefas

entre o Macro e Micro-Monitor?

Os seguintes aspectos devem ser considerados:

- A dificuldade no tratamento de interrupções por parte do INTEL 8008
- As limitações da micro-programação:
  - -Memória de controle limitada
  - -Número reduzido de registros no Processador de E/S para armazenamento de dados.
  - -Carência de micro-instruções que realizem operações aritméticas.

Dentro dessas limitações, vai se tentar realizar o máximo possível das tarefas em micro-rotinas.

Propõe-se que uma operação de E/S seja realizada da seguinte forma:

- O Macro-Monitor requisita ao Micro-Monitor uma operação de E/S. Esta requisição é feita através da execução de uma instrução de E/S na UCP.
- O Micro-Monitor, ativado pela execução da instrução de E/S na UCP, inicializa e controla a transferência de um bloco de informação. A informação necessária para a realização da operação é obtida através de Blocos de Controle, previamente preparados pelo Macro-Monitor, em posições fixas na Memória Principal. Os Blocos de Controle contêm informações como a Operação a ser realizada, o Status atual da operação, o Número de Bytes a serem transferidos, o Endereço na Memória Principal para onde ou de onde deve ser transferido o bloco de informações.
- A cada interrupção doperiférico, a UCP entra em WAIT, o Micro-Monitor faz o atendimento da interrupção, atualiza informação no Bloco de Controle na Memória Principal (por exemplo, atualiza o número de bytes já transferidos) e força a UCP sair de WAIT. Apenas quando todo o bloco de informação tiver sido transferido, o Micro-Monitor interrompe o programa em execução na UCP e endereça a rotina de atendimento no Macro-Monitor.

Do ponto de vista da UCP, o Processador de E/S é um canal inteligente. A UCP prepara um programa de canal, na forma de Blocos de Controle e inicializa o canal, que realiza toda a transferência do bloco de informação independente da UCP. Com isso, tem-se diminuído essencialmente o número de interrupções na UCP.

5 - Implementação de uma Pilha. Um novo "Branch & Link"

Pretende-se implementar uma Pilha na Memória Principal, manipulada por micro-rotinas que serão ativadas pela execução de instruções de E/S na UCP. O objetivo da pilha é facilitar a troca de contexto no caso de interrupções, além do armazenamento temporário de informação.

Ainda, usando-se as capacidades da microprogramação, pretende-se simular a execução de instruções "Branch & Link" e instruções de E/S na UCP, (O endereço de retorno será mantido na Pilha na Memória Principal).

On the Specification and Verification of Distributed Systems

Prof. Peter E. Lauer (*)

We will be concerned with the problem of the adequacy of concurrent systems consisting of processes and resources. Examples of adequacy properties are absence of deadlock (starvation), observation of capacity bounds, protection of resources (critical sections), etc. With regard to deadlock, one can approach the problem from the point of view of deadlock recovery or deadlock prevention. In either case one must be able to recognize deadlock when it is present. We will also attempt to determine the complexity (cost) of algorithms for the detection of deadlock at, so to speak compile time.

Our approach is language oriented in the sense that we will develop our concepts by means of a system specification language called Path notation akin to a conventional programming notation. Synchronization statements may be incorporated into a language by associating them with processes or resources (system objects). We choose the latter approach and state the coordination of processes as the permissible order of execution of operations on shared system objects as part of the object definitions.

Because we want to detect adequacy at compile time or program generation time we need to discover structural or static or syntactic criteria rather than dynamic criteria which may require extensive simulation of the systems. The discovery of structural criteria requires the formalization of the semantics of our programming notation by means of a formalism capable of expressing a high degree of concurrency explicitly. Such a formalism has been developed by Carl Adam Petri and is called general net theory. This theory is a mathematical systems theory. In the theory a number of formal analogues of adequacy properties, for example liveness and safeness, have been defined. Structural criteria of adequacy are known for certain subclasses of nets. Furthermore, the complexity of the adequacy recognition problem has been investigated.

(*) University of Newcastle upon Tyne, Inglaterra.

The semantics of the path notation is defined by associating, one-to-one, a transformation rule with every production rule of the grammar of the path notation. The iterative application of the transformation rules to a program in the notation will produce a simulating net of the program. The definition of path notation in terms of nets permits the transferral of structural criteria from nets to the syntax of the path notation so as to make recognition of adequacy at compile time possible. It also permits the estimation of the complexity (cost) of the algorithms to be built into the compiler.

A large number of example programs will be presented and analysed and students will be invited to devise novel programs and to contribute to the solution of the problem of verifying adequacy of the programs.

Pesquisa em Especificação, Verificação e Síntese de Sistemas de Programação

Prof. Carlos J. Lucena

As motivações econômicas e sociais de nossa área de pesquisa estão ligadas ao já conhecido valor econômico da computação, em particular do software, em nosso país (1,5 por cento do PNB: taxa de crescimento superior ao da economia). O custo se origina principalmente da incapacidade do pessoal técnico existente em identificar problemas relevantes e resolvê-los através de computadores.

A motivação científica de nossa pesquisa reside no desafio de se caracterizar precisamente objetos tais como programas e linguagens de programação e de se formalizar o processo de solução de problemas através de programas.

Nossa área de interesse pode ser genericamente designada como engenharia de software e cobre especificamente as subáreas de:

1. projeto de linguagens (em particular, atualmente, a área de linguagens de especificação);
2. verificação de programas (técnicas para teste de programas e métodos para o estabelecimento da correção de programas, com ênfase na correção de estruturas de dados);
3. síntese de programas (derivação sistemática de programas a partir de uma especificação formal).

1 - Projeto de Linguagens

Nosso esforço nesta área tem-se concentrado no projeto de linguagens de especificação a nível de sistema, em particular em experiências associadas à linguagem PSL (três teses de mestrado em 1976).

Atualmente estamos projetando uma linguagem que venha a competir com o PSL do ponto de vista de expressão de uma especificação a nível de sistema e que supere o PSL do ponto de vis

ta da análise automática da consistência de uma especificação [1]. Usamos os conceitos de descrição de estados no estilo de "queries by examples" e execução simbólica da especificação.

O confronto desse resultado com os produzidos pelo PSA será possibilitado por um trabalho que visa avaliar em detalhe a capacidade analítica do PSA e sugerir melhorias para o sistema existente [2].

Procuramos ainda obter maiores informações sobre que parâmetros de sistema (volume, dimensão, etc) devem constar do nível de especificação através de um trabalho que visa projetar fisicamente um sistema de programação a partir de uma especificação PSL anotada por requisitos [3].

2 - Verificação de Programas

Tendo trabalhado durante algum tempo em métodos formais para a verificação de programas que manipulam estruturas de dados, estamos atualmente concentrando esforços na re-avaliação de métodos para teste de programas. Acreditamos que o esforço colocado na tentativa de aproximar resultados da área de correção através de métodos de teste produz frequentemente idéias objetivas sobre como retomar o trabalho na área de correção.

Estamos estudando um método de teste baseado na estrutura interna de programas, que utilizando interpretação simbólica de uma maneira "forward", possibilita a deteção prematura de "pares impossíveis" na estrutura de programas [4].

Outro enfoque consiste no desenvolvimento de uma metodologia que visa gerar dados de teste a partir da especificação de um programa e de informações retiradas de sua estrutura interna [5]. Esperamos que ambos os trabalhos venham a ser apoiados em sistemas de software de aplicação bastante generalizada.

3 - Síntese de Programas

Estamos empreendendo esforços preliminares para formular um método para a derivação sistemática de programas a partir de uma especificação formal. O método trata o processo de derivação dos pontos de vista de produção de um esquema de programa e da produção de uma representação de dados a partir do

tipo de dados em que se baseia o problema [6].

Desejamos confrontar o enfoque exposto acima com a noção de derivação de programas a partir de transformações. Um estudo sobre o problema se encontra em desenvolvimento [7].

[1] "Projeto de uma Linguagem de Especificação de Sistemas", ; Gustavo Joppert; Tese de mestrado em desenvolvimento.

[2] "Análise Automática de Especificação Formal de Sistemas de Informação "; Laura Maria M. Cavaciocchi; Tese de mestrado em desenvolvimento.

[3] "Análise de Requisitos de Sistemas de Informação"; Elizabeth Maragno; Tese de mestrado em desenvolvimento.

[4] "Análise de Aspectos Estruturais de Programas para Auxílio na Geração de Dados de Teste"; Antonio M. da Silveira ; Tese de mestrado em desenvolvimento.

[5] "Uma Metodologia de Teste Baseada na Especificação e Estrutura de Programas"; Omar de Abreu Lopes; Tese de mestrado em desenvolvimento.

[6] "Assigning Programs to Meanings: A Case Study"; C.J. Lucena e Tarcisio Pequeno; estudos preliminares para uma tese de doutorado.

[7] "Estudos sobre Transformações de Programas"; Ascendino Rodrigues de Araujo; Tese de mestrado em desenvolvimento.

Análise de Requisitos de Sistemas de Informação (proposta de tese de mestrado)

Elizabeth de J. Maragno

1 - Descrição Sumária do Problema

Nos últimos anos, o computador tornou-se uma ferramenta utilizada para dar suporte aos sistemas de informação em muitas organizações, fornecendo informações necessárias para decisões racionais e executando diferentes funções desta organização.

Com o crescente aumento do uso do computador em aplicações relacionadas a Sistemas de Informação e dado que os métodos disponíveis para a construção desses sistemas contêm numerosas deficiências devidas principalmente à interaçãohumana,vários pesquisadores têm-se preocupado com o desenvolvimento de ferramentas automatizadas para a produção de Software de aplicação.

Analisando-se os problemas da área de Sistemas de Informação, vê-se que existe a necessidade de técnicas para a sistematização da expressão dos requisitos de informação dentro de uma organização, tais como: definição das informações necessárias; volume dessas informações; tempo requerido para produzí-las; dados a partir dos quais podem ser produzidos os resulta - dos, etc.

É de grande importância, tanto para o processamento automatizado de sistemas de informação como para sistema de processamento de dados em geral, que durante o estabelecimento dos requisitos do sistema, os mesmos sejam cuidadosamente analisados.

Baseados na importância dessa análise interativa surgiram as primeiras técnicas semi-formais de especificação e análise de requisitos de sistemas. Dentre as técnicas existentes colocamos ênfase no PSL/PSA (Problem Statement Language / Problem Statement Analyzer) projetado na Universidade de Michigan pela equipe de Daniel Teichroew, que especifica analisa e fornece documentação estruturada de sistemas de informação, com assistência do Computador.

O objetivo último do PSL/PSA é o de conduzir a construção de software através de processos totalmente automatizados. Nunamaker pretende tal resultado através do sistema Soda que é uma metodologia para automatização das funções de projeto de sistemas a partir de comandos de requisitos de processamento de seu sistema. A partir desses comandos e de alguma interação com o projetista, gera-se um conjunto de problemas e arquivos de dados [3].

Segundo o enfoque de Nunamaker [1], o PSL/PSA é utilizado para a definição dos requisitos do sistema. A definição do problema é então analisada pelo PSA que determina a consistência e completeza da mesma, fornecendo aos projetistas "feedback" para modificações. Quando a definição do problema está "completa" e "consistente", o Banco de Dados do PSA conterá informações suficientes para iniciar a fase do projeto Físico que consiste em 2 etapas: Especificações de Módulos do Programa e Organização dos Dados.

O estudo proposto para a tese se concentrará em uma metodologia não necessariamente automatizável (ou seja, com maior flexibilidade para o tratamento de informações sobre requisitos) para consecução das 2 etapas descritas acima. O resultado do trabalho será um conjunto de procedimentos para auxiliar o planejamento da fase posterior: implementação.

Esse estudo foi motivado pela importância de uma avaliação, ou seja, uma análise de performance antes de se implementar um determinado sistema. A metodologia proposta agrupará processos e arquivos de maneira a obter uma solução eficiente para a implementação por uma determinada máquina. Espera-se produzir assim um sistema que tenha um nível de performance realístico, dado os objetivos originais e os recursos disponíveis.

2 - Objetivo da Tese

O objetivo da tese é a consecução de uma metodologia para elaboração do Projeto Físico de um Sistema de Informação: Especificação de Módulos do Programa e Organização de Dados, para uma especificação funcional expressa em PSL.

O trabalho consiste em descrever uma metodologia para agrupamento de Processos e Arquivos e indicar procedimentos a serem seguidos, afim de que a implementação a ser construída tenha como resultado um projeto eficiente com relação à uma dada máquina.

Supõe-se neste estudo, que se tem um problema bem definido e que o conjunto de processos e arquivos possa ser descrito por meio de um grafo dirigido. Deve-se ainda conhecer outras informações tais como frequência e volume, ou seja, os parâmetros do sistema que expressam os requisitos da especificação.

A orientação do trabalho será baseada no enfoque adotado por Nunamker [2]. Nele começa-se por uma representação baseada em conceitos da Teoria dos Grafos que expressa a interação entre processos e arquivos. Depois passa-se para a construção de matrizes, a partir do estudo de Lanfefors [4] sobre Algebra de Matrizes.

O objetivo da fase de Especificação de Módulos do Programa é a determinação de uma modularização efetiva, ou seja, determinar os agrupamentos possíveis e a avaliação da economia de volume de transporte resultante desses agrupamentos; visto que, em geral, para que se tenha projeto eficiente é necessário que se reduza o volume de transporte de tal forma que o tempo de processamento seja reduzido proporcionalmente.

Depois que os módulos do programa são determinados, os arquivos são consolidados, com o propósito de reduzir o número requerido de Entradas/Saídas a partir dos arquivos para melhor utilização da forma de armazenamento. O objetivo da fase de Organização de Dados será o de determinar as estruturas de dados e estruturas de armazenamento de tal forma que se conseguir minimizar o volume de transporte.

3 - Contribuição Esperada

A contribuição esperada é a proposta de uma metodologia não necessariamente automatizável, que possibilite a explicação do Projeto Físico de tal forma que a fase posterior, a implementação, satisfaça tanto a definição do problema (especi-

ficação) quanto os requisitos previstos para a mesma.

Com a metodologia a ser apresentada espera-se conseguir a proposição de um projeto eficiente para uma dada máquina específica, diminuindo-se assim a necessidade de se submeter a implementação a uma avaliação de performance experimental, uma vez que a mesma estará implícita na metodologia a ser usada.

4 - BIBLIOGRAFIA

[1] - Nunamaker, J.F., Konsynski, B., "Progress Report on Automatic Code Generation From PSL" - Management Information Systems, University of Arizona.

[2] - Nunamaker, J.F., Nylin, W. C. Konsynski, B., "Processing Systems Optimization Through Automatic Design and Reorganization of Program Modules" - Department of Computer Sciences, Purdue University

[3] - Couger, J.D, Kn app, R. W., "System Analysis Techiques" Wiley 1974.

[4] - Langefors, B., "Theoretical Analysis of Information System"

[5] - Teichroew, D., Hershey III,E.A, Bastaroche M. J., "An introduction do PLS/PSA" - ISDOS Working Paper Nº 86 March 1974.

[6] - Teichroew, D. Berg, D.L.R, Hershey III, E.A., Berg D.L.R, "An example of the use of PSL using top-down analysis " - ISDOS Working Paper nº 74, April 1974

[7] - Teichroew, D., Bastarache, M.J., "PSL user's manual" - ISDOS Working Paper nº 98, March 1975

[8] - Teichroew, D., Hershey. III, E.A. "PSL/PSA: A Computer-aided Technique for Structured Documentation and Analysis of Implementation Processing Systems" - IEEE Transaction on Software engineering, vol. 3 nº 1, January 1977.

[9] - Grahom, R.M., Clancy Jr., G.J., Devaney, D.B., "A Software Design and Evaluation System - Communications of the ACM, Vol. 16, Nº 2, February 1973

[10]- Nunamaker, J.F., Konsynski, B.R. Ho, T., Singer, C., "Computer-Ardel Analysis and Design of Information Systems -

Communications of the ACM, Vol. 19, Nº 12, Decemeber 1976.

[11]- SODA/1, "Systems Optimization and Design Algorithm / Version 2" - ISDOS Working Paper nº 36, June 1970.

"Ferramentas para o Projeto e Construção de Sistemas de Informação Apoiados em Banco de Dados"

Prof.: Rubens Nascimento Mello

1 - Introdução

Pode-se dizer que os estudos e pesquisas na área de banco de dados tem relegado a segundo plano a integração do projeto de Banco de Dados com os outros aspectos importantes do sistema de informação envolvente. Por outro lado a maioria das pesquisas sobre o processo de construção de sistemas de Informação adotam ainda o enfoque tradicional onde os dados são centralizados em um Banco de Dados e cada processo tem seus próprios arquivos.

Neste projeto procuramos contribuir para o projeto e construção de sistemas de informações baseados em Banco de Dados.

Nas várias fases do projeto consideramos os processos (aplicações) e recursos (dados) do sistema em vários níveis de abstração.

No nível da especificação conceitual [1] os fenômenos relevantes do sistema real são completamente (o mais possível) descritos independentes de software ou estrutura fixas de dados.

No nível seguinte, os processos e dados conceitualmente especificados na fase anterior, são mapeados adequadamente para as estruturas lógicas de dados e suas operações. Este nível de descrição do sistema, em geral já determina um certo tipo de software de Banco de Dados comprometido com um certo modelo de dados.

No próximo nível os processos e dados devem ser descritos em termos de estruturas de armazenamento e suas operações correspondentes. A adoção do enfoque de Banco

de Dados para o sistema de informação sendo desenvolvido , permite reduzir este problema ao problema de escolher estruturas de armazenamento do Banco de Dados para o conjunto de aplicações (processos) descritas no nível lógico.

Alguns softwares de Banco de Dados, assim como o que pretendemos desenvolver permitem escolha de estruturas de armazenamento e neste caso podemos tentar otimizar esta escolha para o conjunto de aplicações do sistema. Após escolhida a estrutura de armazenamento de Banco de Dados e tendo as aplicações descritas neste nível teremos praticamente implementado o "Sistema de Informação Apoiado em Banco de Dados".

2 - Objetivos da Pesquisa

Implementar o sistema (semi) automaticamente a partir da sua especificação conceitual pode ser colocado como o objetivo a longo prazo. Neste projeto, entretanto, pretendemos contribuir para este objetivo com algumas ferramentas para a especificação conceitual de sistemas de informação e para a construção do sistema usando software de Banco de Dados.

Basicamente procuramos agora os seguintes resultados

a) Definição e Implementação (parcial) de uma linguagem para a especificação conceitual de sistemas de informação apoiados em Banco de Dados.

- Como a especificação conceitual se compõe de:
  - Descrição do modelo conceitual do Banco de Dados
  - Descrição da evolução do Banco de Dados no sistema

esta linguagem deverá conter:

a.1- Uma sublinguagem para a descrição do modelo conceitual do Banco de Dados que envolve:

. Descrição conceitual (estatica) dos dados

. Restrições de integridade

a.2) Uma sublinguagem para a descrição dos processos que definem a evolução dos dados no sistema.

b) Definição e implementação de ferramentas para os mapeamentos descritos na seção anterior, incluindo:

b.1) Definição e implementação da linguagem da especificação de estruturas de armazenamento de Banco de Dados.

b.2) Mecanismos que permitam mapear os dados e processos descritos no nível lógico independentes de estruturas de armazenamento em esquema interno [2,3] de Banco de Dados e programas em linguagem de programação comum como FORTRAN (extendida).

b.3) Pacotes de rotinas para a manipulação de arquivos generalizados e ligações entre eles. As rotinas devem ter chamadas uniformes independentes das estruturas de armazenamento adotadas para os arquivos.

Havendo alternativas para a escolha das estruturas dos arquivos pode-se procurar a sintonização das aplicações do sistema com as estruturas de armazenamento para melhor eficiência.

b.4) Extensão da linguagem de propósito geral através de pré-processador para simplificar a programação dos processos do sistema [4].

b.5) Definição e implementação (parcial) de linguagens para interação não programatica (linguagem de consulta para o Banco de Dados do Sistema).

3 - Andamento da Pesquisa e Resultados Parciais

O projeto envolve alguns programadores e alunos de mestrado cujas teses sairam/sairão das implementações ou estudo mais detalhado das ferramentas mencionadas acima.

Podemos considerar que algumas ferramentas já foram desenvolvidas, contribuindo para os objetivos globais do projeto. Estas ferramentas tem sido usadas na construção de outras ferramentas. Por exemplo:

a) Processador de Macros de Propósito Geral[5]

Este processador de macros pode ser usado como um programa independente (processador de símbolos) ou como um pré-processador para extensão de linguagens de programação.

b) FORTRAN Estruturado (FORTS-PUC) [4]

Uma extensão das estruturas de controles de FORTRAN para facilitar programação estruturada. Esta extensão foi feita via macros e como se trata de pre processamento através do programa mencionado acima , sua característica de processador de macros foi preservada.

c) Extensão de Linguagem para Manipulação de Banco de Dados [6]

Este trabalho resultou numa tese de mestrado onde foi feita a especificação de uma extensão via macros para incluir em FORTRAN comandos de Manipulação de Bancos de Dados, segundo a proposta da CODASYL[7] e mapeá-los para subrotinas de manipulação do Banco de Dados do PSA[8]

d) Normas para Documentação do Projeto de Sistemas

Este trabalho está praticamente terminado e

e define as normas de documentação tanto das atividades como dos programas do nosso projeto.

e) Rotinas Auxiliares para Manipulação de Caracteres em FORTRAN

A linguagem base para a construção dos programas tem siFORTS. Devido aos comandos DOWHILE , REPEAT, IF-THEN-ELSE, PERFORM, etc,a programação tem sido bastante facilitada. Os problemas relativos a manipulação de caracteres tem sido resolvido através de rotinas auxiliares (em ASSEMBLER apenas por questões de eficiência) que movem, comparam e acham substrings em string de caracteres definidos sobre vetores. O armazenamento dos strings numa área só para facilitar mais a sua manipulação já foi estudada [9], porém não incorporada ainda ao FORTS.

Algumas teses de mestrados estão em andamento tratando da obtenção das ferramentas mencionadas na Seção 2. assim como dos estudos relativos aos mapeamentos e escolha de estruturas de armazenamento mencionados na Seção 1.

Referências:

1 - Delboni, E.G. "Especificação Conceitual de Sistemas de Informação apoiados em Banco de Dados" Tese de Mestrado - PUC RJ (1977)

2 - Melo,R.N. "On the mapping from the conceptual Specification to the internal model in Data Base design" (a ser publicado).

3 - _____, Lellis,L. "O mapeamento da especificação conceitual para um esquema interno usando o DBMS TOTAL" Relatório Técnico - PUC-RJ (a ser publicado)

4 - _____, Schwabe,D. "Extending the control structures of FORTRAN via a macro generator" TR 1/76 - PUC-RJ

5 - _____, "PM-Um processador de macros" Tese de Mestrado - ITA - 1971

6 - Andrada,G.K., "Especificação de uma sublinguagem de manipulação de dados" Tese de Mestrado - PUC-RJ (1976)

7 - CODASYL Data Base Task Group Report - April 1971

8 - Hershey,E.A., Messink,P.W., "A data base Management System for PSA based on DBTG 71" ISDOS Working Paper No. 88 julho 1975

9 - Melo,R.N., "Implementing Character strings in FORTRAN TR /76 PUC-RJ

"Análise de um Algoritmo que Implementa uma Tabela de Símbolos Para Tradutores com Grandes Volumes de Símbolos" (Proposta de tese de Mestrado)

Mário Martins

1 - Descrição da Área:

1.1 - Tabela de Símbolos

Uma tabela de símbolos é constituida por dois mapeamentos que mapeiam cadeias de caracteres - símbolos - sobre numerais naturais e números naturais sobre o símbolo definido correspondente ou então indefinido.

Tabelas de símbolos são algumas das estruturas de dados usadas em tradutores. A sua finalidade é única e exclusivamente a conversão de cadeias de caracteres em números de tamanho fixo que identifiquem estas cadeias e vice-versa. Não faz parte da tabela de símbolos a tabela de atributos (tipos, estruturas, etc.) dos nomes (variáveis) utilizada nos programas.

1.2 - Tabela de Símbolos para Tradutores

Uma tabela de símbolos para tradutores é caracterizada por um alto índice de instalação de símbolos. A cada diferente símbolo encontrada será associado um outro número. Portanto, a capacidade da tabela, em caracteres, deve ser virtualmente ilimitada.

Os símbolos encontrados em programas são quaisquer cadeias de caracteres de zero ou mais caracteres não havendo limite superior de tamanho da cadeia de caracteres. Os símbolos têm, portanto, tamanhos extensamente variáveis e imprevisíveis. Na prática, podemos limitar o tamanho a um valor razoavelmente grande, por exemplo 5000 caracteres.

1.3 - Características dos Algoritmos a Serem Usados

Desejamos utilizar algoritmos eficientes, que implementem tabelas com capacidade virtualmente ilimitada, que permitam instalar-se símbolos de tamanhos virtualmente ilimitados, e que otimizam programas escritos por usuários convencionais, isto é, programas normais com poucos símbolos.

Visto que os símbolos poderão ser de tamanhos virtualmente ilimitados, deveremos, portanto, particioná-los em blocos de tamanho fixo. Através do encadeamento desses blocos poderemos, então, armazenar símbolos de qualquer tamanho. E de maneira que a tabela tenha capacidade virtualmente ilimitada, deveremos fazer uso de memória auxiliar.

2 - Objetivo:

Vamos, neste trabalho, fazer um estudo, do ponto de vista probabilístico (teórico), do comportamento de um algoritmo que implementa uma tabela de símbolos para tradutores com grandes volumes de símbolos.

Para isto, vamos determinar curvas de comportamento relativas à variações de parâmetros, para que possamos otimizar a tabela de símbolos.

Através de experimentos sucessivos, deveremos comprovar estes resultados teóricos estimados.

3 - Contribuição Esperada:

Esperamos poder determinar parâmetros de comportamento de uma tabela de símbolos para tradutores permitindo um melhor dimensionamento de tabelas em ambientes específicos.

"An Illustration of Some Ideas on the Derivation of Programs

Prof. Tarcísio H. C. Pequeno

The state of the art in the area of program verification is now reaching a stage in which many of the existing results about the analysis of programs are starting to be transferred to practice through no software engineering. Efforts have now turned to the goal of providing a methodology for the systematic (possibly semi-automatic) synthesis of programs. In fact, many people are presently working on the problem of deriving a program from a given program specification. Dijkstra has been exploring recently the idea of predicate transformers [1,2] and proposes a methodology for the derivation of programs form their post- conditions and pre-conditions (specifications).

While these efforts are taking place, an overwhelming majority of programmers spend their time writing programs in ALGOL-like languages with very little understanding about the objects they are producing. A very important task to be undertaken at the present, consists of explaining to these programmers some of the more established ideas on the nature of the program synthesis process.

Gries [3] contributed to this purpose by illustrating some of Dijkstra's ideas while applying them to a reasonably typical programming example. Our attention was called to the fact that the current practice of separating the algorithm and the data aspects of a program in the program development process (program= algorithm + data [4]) was not taken into consideration in Gries ' example.

In [5] we discuss an alternative solution to the example proposed by Gries in [3], by dealing separately with the algorithm and data aspects of the program. For the establishment of this separation we have used the concept of a cluster which is instrumental for providing a programming mechanism for the encoding of the data representation. The same effect could be obtained by mechanisms such as classes and forms.

In deriving a program we have gone through three distinct phases: derivation of a program schema from a formalized version of the problem definition; derivation of the problem data type ultimately in terms of a formally well known and more primtive type; derivation of a programmed version of the data type definition (synthesis of the cluster). We not only defined the problem data type but also checked its correctness. The checking procedure differs slightly from Hoare's [6] since we do not start from a completely defined type and a completely defined representation and try to define a mapping function connecting them. Instead, we express the model constructively in terms of the representation and then verify if it is in fact a model of the theory.

In the example presented in [5] we have explicity used the type axioms for the derivation of programs. We are presently trying to expand and formalize this concept in search for a better understanding of the program derivation process. We are also investigating the idea of dealing with the problem of program transformations (in Gehardt's sencs [7])viewing these transformations along the two axie dealt with in this paper: algorithm and data .

References

[1] Dijkstra, E., "Guarded Comands, Non-determinacy and a Calculus for the derivations of Programs, "Proceedings of the International Conference on Reliable Software. 1975.

[2] _____, "A Discipline of Programming", Englewod Cliffs , N.J., Prentice Hall, 1976

[3] Gries,D., "An Illustration of Current Ideas on the Derivation of Correctness Proofs and Correct Programs". IEEE Transactions on Software Engineering Vol. S.E. 2, nº 4, 1976.

[4] Wirth,N., "Algorithms + Data Structures = Programs", Prentice Hall, 1976.

[5] Lucena,C.J.P. e Pequeno,T.H.C. "A View of the Program Derivation Process Based on Incompletely Defined Data Types: A Case study". To appear .

[6] Hoare,C.A.R. "Proof of Correctuens of data Representations"; Acta Informatical 1, 1972.

[7] Gerhart,S.L. "Knowledge About Programs: A Model and a Case Study". Proceeding of the International Conference on Reliable Software, Los Angeles, 1975

"Um Sistema Operacional Interativo"

Prof. Albrecht von Plehwe

1 - Introdução

O sistema operacional, descrito aqui, foi projetado para controlar um sistema de desenvolvimento e execução de programas, numa linguagem similar ao BASIC, junto com um sistema de ensino para essa linguagem. O sistema se destina ao uso em escolas secundárias e pode controlar até quatorze terminais de usuários. O diálogo é a forma dominante de uso mas existem também, fases de uso intensivo da unidade central de processamento (ucp) , em que o usuário executa um programa gerado.

2 - Algumas características do hardware

O G-10 é um minicomputador de 16 bits, desenvolvido pela universidade de São Paulo, com um desempenho comparável ao LSI 11, menor computador de linha PDP11, mas possui estados de supervisor e usuário e facilidades de relocações dinâmica. Cada processo é composto de dois segmentos, código e dados, relocados e limitados independentemente pelo hardware. A máquina dispõe de um disco de braço móvel com capacidade de 10 M bytes.

3 - Arquivos

Um arquivo se apresenta como um espaço de armazenamento de acesso direto, endereçado de zero até o endereço máximo que corresponde ao tamanho de arquivo. O acesso pode ser feito em blocos de palavras de tamanho variável, a partir de qualquer endereço inicial. Fisicamente um arquivo é dividido em volumes, cada volume reside ou na memória principal ou inteiramente no mesmo desvio.

4 - Segmentação

Cada processo é composto de dois segmentos, um segmento de código e um de dados. Todos os segmentos podem ser residentes na memória principal ou podem ser transientes. Segmentos transientes são carregados na memória de superposição para a execução do processo.

4.1 - Segmentos de código

Segmentos de código são protegidos pelo hardware contra modificações. Cada segmento de código corresponde a uma determinada tarefa de processamento. Um método de 'code chaining' é usado, i é., cada segmento pode transferir o controle para outros segmentos de código. Cada segmento é compartilhado por todos os processos de usuário ligados a ele naquele momento. Erros formais e tentativas de violações de um mecanismo de proteção envocam um segmento de recuperação. Cada processo de usuário pode definir e mudar dinamicamente o seu segmento de recuperação. Os segmentos de códigos são definidos na operação do sistema junto com as características e os direitos de cada um, tais como o seu tamanho, a sua parcela de tempo, os segmentos que ele pode chamar ,etc.

4.2 - Segmentos Dados

Cada processo de usuário dispõe de um segmento de dados que o acompanha durante toda a sua vida. O segmento de dados é o volume inicial de um arquivo de 'área de dados', particular a cada processo de usuário, uma 'janela' neste arquivo que é diretamente acessível. Todo o arquivo, também , é acessível pela entrada/saída de arquivos. O processo pode mudar dinamicamente o tamanho desta 'janela'.

A cada processo de usuário é alocada permanentemente uma área da memória principal em que o segmento de dados fica armazenado se isto é possível. Caso contrário, o segmento é feito transiente e a área na memória é usada como 'buffer' de entrada/saída.

5 - Compartilhamento de recursos entre processos de usuários

5.1 - Memória de Superposição

A memória de superposição é dividida em duas partições com uma fronteira flutuante entre elas. Cada partição pode armazenar um segmento transiente. O carregamento de segmentos de código é dirigido por uma lista que contém cada segmento pelo menos uma vez. Em um ciclo cada segmento que tem um usuário ativo ligado a ele é carregado na ordem dada pela lista. No fim um segmento transiente de dados e o correspondente segmento de código são carregados e o ciclo começa novamente.

5.2 - Controle da unidade central de processamento

A alocação da ucp depende do estado dos segmentos dos processos, dando prioridade a processos com ambos os segmentos residentes na memória. A fila da ucp está dividida em três sub-filas:

Na primeira fila, de mais alta prioridade, se encontram os processos com ambos os segmentos residentes na memória. Esta fila é ordenada de acordo com a prioridade de de código.

Na segunda fila estão os processos com pelo menos um segmento transiente.

Na terceira fila, de menor prioridade, são colocados os processos que gastaram uma parcela inteira de tempo numa das duas primeiras filas sem entrar num estado de espera. Esta fila é organizada em forma circular.

6 - O uso das facilidades do sistema operacional pelo sistema BASIC.

Cada terminal é representado internamente por um

processo de usuário. Os processos de usuários são tratados sem preferências, porque os prioridades são atribuídas as tarefas de processamento, representados pelos segmentos do código, e mudam dinamicamente com estes segmentos. Cada processo de usuário dispõe de dois arquivos particulares e compartilha vários arquivos do sistema em modo 'ler somente' . O sistema, entretanto, pode ser colocado em 'modo único' , em que ele permite somente um usuário. Neste modo todos os arquivos compartilhados são acessíveis sem restrições, permitindo assim a sua manutenção.

A área permanentemente alocada a cada processo de usuário na memória foi dimensionada tal que a memória das tarefas para ser executada com segmentos de dados residentes. Segmentos de código normalmente são transientes porque eles são compartilhados e não precisam ser reescritos no disco.

As parcelas de tempo, atribuídas os segmentos de códigos , são suficientes para terminar a maioria das tarefas em uma única parcela.

"Análise de Aspectos Estruturais de Programas para Auxílio na Geração de Dados de Testes" (proposta de tese de mestrado)

Antonio M. Silveira

1 - Descrição Sumária do Problema

Um dos grandes problemas encontrados no processo de desenvolvimento de software é garantir até que ponto um determinado produto é confiável. Cêrca de 50% dos custos totais dispendidos no desenvolvimento de software's são gastos em processos de teste e verificação, o que mostra a urgente necessidade em pesquisar-se novos métodos que atenuem este percentual.

Várias pesquisas já foram desenvolvidas na área de teste e, continuam-se pesquisar novos métodos em busca de uma perfeição, cada vez maior da solução. Das pesquisas já realizadas, os pesquisadores tem seguido duas diferentes linhas de pesquisa; uma que busca a solução do problema a partir dos aspectos estruturais do programa e, outra que tenta solucionar o problema a partir da especificação. A grande vantagem da primeira é que além de sua operacionalidade manual devida exclusivamente ao usuário, pode também ser automatizada e, consequentemente ter menos interação com o usuário, o que é muito importante no processo de teste. O trabalho proposto se enquadra na primeira linha de pesquisa,como uma alternativa em teste de programas, solucionando alguns dos inúmeros problemas encontrados ao se buscar uma nova metodologia de geração de dados de teste

2 - Objetivo da Tese

O objetivo deste trabalho é apresentar uma nova metodologia de geração de dados de teste, a partir dos aspectos estruturais de programas. Inicialmente estes aspectos

são analizados e, o resultado desta análise é utilizado como subsídio para auxiliar na geração de dados de teste. Este trabalho pode ser resumido da seguinte forma:

- Dado um programa codificado em uma linguagem de programação a ele é associado um grafo dirigido, que na maioria dos casos é cíclico. A este grafo cíclico é associado um grafo acíclico equivalente.

- O usuário então escolhe qual caminho do programa deseja testar.

- O caminho é então executado simbolicamente por substituição FORWARD e, à medida em que vai sendo executado, vai gerando um conjunto de restrições que devem ser satisfeitos para que o caminho seja exercitado. A execução simbólica pára quando detectamos alguma inconsistência num par de restrições, o que é ocasionado quando executamos simbolicamente um caminho não atravessável. Caso o caminho seja atravessável, no final da execução do mesmo teremos um conjunto de restrições (inequalidades) que solucionadas nos fornecerão um conjunto de dados de teste que exercitarão o caminho.

3 - Contribuição Esperada:

Com este trabalho, esperamos ter contribuído com uma alternativa para testarmos softwares, apresentando solução de problemas que contribuem para a complexidade da geração de dados de teste tais como:

- Caminho não atravessáveis em programas
- Manipulação de arrays

A metodologia apresentada tem a vantagem de deixar a critério do usuário a escolha do caminho a ser testado e, de detectar caminhos não atravessáveis antes de que todo o caminho seja executado simbolicamente. A detecção ocorre no momento em que aparece uma combinação impossível de restrições durante a execução simbólica de um caminho.

"Atividades de Pesquisa e Desenvolvimento"

Prof. Arndt von Staa

1 - Introdução

Este documento apresenta um pequeno relato das atividades de pesquisa e desenvolvimento sob a minha orientação.

Para todas as atividades estão previstos laboratórios , ou, pelo menos, mecanismos que permitam verificar-se os resultados na prática.

2 - Projeto de Sistemas

Neste programa de pesquisa são estudados os problemas relacionados com o desenvolvimento de grandes sistemas de programação, particularmente sistemas de processamento de dados.

As atividades compreendem determinar metodologias para:

- especificação da proposta de desenvolvimento
- especificação do plano do projeto de desenvolvimento
- documentação
- aceitação do produto final durante o ciclo de desenvolvimento
- aceitação de um sistema de programação

Dentro deste programa de pesquisa estão sendo desenvolvidas as seguintes teses:

- João Dias de Queiróz - Análise e Definição de Requisitos, Roteiro para Criação da Proposta deDesenvolvimento;
- João Victor de Lellis - Sistemas de Informação Distribuidos;
- Maria Cristina Leite Gomes - Ciclo de Vida de um Projeto de Desenvolvimento;

- Paulo Sérgio Vilches Fresneda - Gerência de de Desenvolvimento;

- Sóstenes Apolos da Silva - Metodologia de Documentação para permitir a agregação da documentação conforme as necessidades do leitor.

Sob supervisão de outros orientadores estão sendo desenvolvidas as seguintes teses:

- Solange Lima Asteggiano - orientada por Luiz Carlos Sá Carvalho (D Adm) - Análise de Sistemas, com vistas a determinar os requisitos de informação;

- Anselmo Frizera Junior - orientado por Jayme Goldstein (NSI) - Decomposição de um sistema integrado em diversos Sistemas Integráveis;

- Elizabeth de Jesus Maragno - orientada por Carlos José de Lucena - Análise de Requisitos;

- Gustavo Henrique Joppert - orientado por Carlos José de Lucena; Linguagens de Especificação de Sistemas de Informação.

Este programa de pesquisa é virtualmente ilimitado. Entre possíveis campos de estudo podemos citar:

- especificação do plano do projeto de desenvolvimento ;
- ferramentas para a especificação de requisitos e do plano;
- ferramentas para a gerência de projetos;
- ferramentas para auxílio de desenvolvimento;
- especificação de um ciclo de vida que permita um controle rigoroso de qualidade do produto durante o desenvolvimento.

3 - COMCOM 2

Este programa de pesquisa e desenvolvimento visa criar fer-

ramentas para o auxílio no desenvolvimento de compiladores e pré-processadores.

As atividades compreendem:

- desenvolver um gerador de tradutores léxicos;
- desenvolver um gerador de transdutores sintáticos e semânticos;
- desenvolver um gerador de geradores de código;
- desenvolver rotinas de suporte à compilação e a execução.

Dentro deste programa de pesquisa em desenvolvimento está sendo produzido o gerador de filtros léxicos.

Está sendo desenvolvida uma Tese de Mestrado por Mário Martins sob orientação de Gaston Gonnet visando determinar os parâmetros que tornem eficientes tabelas de símbolos para grandes volumes de símbolos.

Foi desenvolvido como trabalho individual por Antonio Rodrigues Neto, um simulador de tabelas de símbolo para grandes volumes de símbolos.

A continuação do projeto COMCOM 2 requer diversas atividades de pesquisa:

- transdutor sintático e semântico integrado e que tenha um mecanismo de recuperação de erros embutido;
- gerador de geradores de código.

Requer também atividades de desenvolvimento:

- rotinas de suporte à compilação;
- rotinas de suporte à execução.

4 - Redes de Microprocessadores

Neste programa de pesquisa são estudadas arquiteturas de computadores baseados em redes de microprocessadores funcionalmente distribuidos.

As atividades compreendem:

- desenvolver simuladores;
- desenvolver "cross" linguagens;
- desenvolver diversos modelos;
- comprovação dos modelos através da montagem de computadores;
- estudos teóricos relacionados à otimização, distribuição e inibição de bloqueios;
- desenvolvimento de sistemas operacionais.

Dentro deste programa de pesquisa está sendo desenvolvida a tese Simulador de Redes de Microprocessadores por Ivo Maurício R. de Magalhães.

Este programa de pesquisa encontra-se em fase inicial e, portanto, está totalmente inexplorado dentro do Departamento de Informática. Como campo de pesquisa, apresenta ainda grandes aberturas, uma vez que tem sido pouco explorado internacionalmente.

"Linguagens de Controle"

Prof. Michael A. Stanton

Estamos interessados no interface externo de um programa .

De nosso ponto de vista um programa é um "procedure" com parâmetros que retorna um valor, isto é uma função. Os parâmetros são arquivos e uma cadeia (parameter string).
O valor será um código de condição.

Por exemplo:

Seja "copiar" um programa que copia um arquivo sequencial, talvez desempenhando algumas funções primitivas de edição, especificadas no "parameter string". Copiar usa os nomes simbólicos (parâmetros formais) "original", "copia" e "mensagens" para indicar os arquivos utilizados, e o nome "opções" para indicar a cadeia de parâmetros. O nome "código de condição " representa o valor retornado.

Então, podemos descrever o programa:

```
proc  copiar (file original, copia, mensagens;
      string(opções) int codigo de condição  :
      ( < corpo do programa > ).
```

Uso do programa:

O procedure "copiar" tem que ser chamado de um nível mais alto.

Idealmente:

```
int x; file ARQUIVO.UM
  :
 x:= copiar (ARQUIVO.UM, "terminal" , "terminal");
  :
```

poderia ser usado para listar o arquivo ARQUIVO.UM num terminal.

Linguagens reais no 370 como OS/MVT

1. JCL

```
//STEP1        EXEC PGM=COPIAR,PARM= '<opções>'
//ORIGINAL     DD   DSN=ARQUIVO.UM,DISP=SHR
//COPIA        DD   SYSOUT=A
//MENSAGENS    DD   SYSOUT=A
//STEPLIB      DD   DNS=SYS10.PROGS,DIS=SHR
//STEP2        EXEC PGM=xxx,COND=(0,LT,STEP1)
//   :
```

2. TSO - Command Language

```
ALLOC    F(ORIGINAL) DA('ARQUIVO.UM')  SHR
ALLOC    F(COPIA)  DA(*)
ALLOC    F(MENSAGENS) DA(*)
CALL     'SYS10.PROGS(COPIAR)'  '<opções>'
FREE     F(ORIGINAL COPIA MENSAGENS)
WHEN     (RC  LT 1) ...
```

3. Phoenix

```
SYS10.PROGS(COPIAR)  (ORIGINAL=ARQUIVO.UM  COPIA=*
     MENSAGENS = *) '<opeções>'
UNLESS CC  ...
```

Procedures nas linguagens de controle

Todas estas 3 linguagens tem a facilidade de criar procedures, isto é uma lista de comandos que será executada ao ser

chamada. Na chamada podem ser inicializados parâmetros locais ao procedure. Por exemplo:

```
JCL //  EXEC  FORTGCLG, PARM.FORT = 'NOSOURCE',
    //  LOADSET = 'MODULO.OBJETO',REGION.GO=200K
TSO  EXEC    'NOSSO.PROCLIB(LISTAR)'  'ORIGINAL(ARQUIVO.UM)
                    OPÇÕES(COM NUMERAÇÃO)'
Phx  C       QUALQUER.ARQUIVO(BACKUP)   TAPE=FT0123
             ACCNT = PBTX,0001
```

Antes de executar o procedure todas ocorrências de parâmetros simbólicos são substituidos pelo texto do valor atribuido na chamada. Chamo isto de macro.substituição. Difere de atribuição de valores e variáveis formais na chamada de um procedure numa linguagem de programação. Observamos que, nesta última, as variáveis normalmente tem tipos. Isto permite:

1. que erros na especificação de parâmetros sejam detetados pelo sintaxe.
2. que procedures sejam compilados antes de uso.

Parâmetros globais.

Estas podem ser valores atribuídos e estados fora de procedures, como tb. dentro, e permite a comunicação entre procedures.

São inexistentes em JCL e TSO CL (versão OS/MVT). Em Phoenix existem e seus valores serão usados dentro de procedures se não existir um parâmetro local homônimo.

Estruturas de controle:

JCL. execução de um programa ou de um procedure pode ser evitada ou forçada de acordo com os valores de códigos de retorno da execução de programas anteriores.

TSO.CL (OS.MVT) quase a mesma coisa.

Phoenix. a. existem comandos condicionais que podem testar:

1. resultados ou erros na execução de comandos anteriores.
2. os valores de parâmetros

b. existem comandos de desvio, de subrotina

Outras facilidades de Phoenix

a. "here" datasets - semelhante a DD * em JCL

b. documentos correntes - um conjunto de até 5 arquivos temporários que normalmente contêm o produto de comandos recentes.

Phoenix na PUC.

Originalmente o sistema rodava na Universidade de Cambridge num 370/165 com OS/MVT muito modificado. Até produzir uma versão adequada para uso geral levou 18 meses. Esta foi lançada em maio 77 e consiste de:

1. a linguagem e seu interpretador
2. um editor
3. programas para manipulação e manutenção de arquivos em disco e fita (FILE,EXAMINE,XPOS,TLS)
4. bibliotecas e procedures de 2 tipos
   (a) compilação de programas
   (b) submissão de jobs p/ executar programas do item 3.

Um exemplo é o procedure BACKUP que submete um job para copiar a uma determinada fita todos os arquivos em disco de um dado usuário.

5. Um interface p/ a chamada de "TSO Command Processos" (IBM)

Dificuldade da implementação:

- uso de linguagem assembler
- sistemas operacionais diferentes - a diferença principal reside na maneira de alocar arquivos dinâmicamente, o que é uma das grandes modificações feitas ao sistema em Cambridge. O uso da facilidade original tem a consequência de restrigir o uso do TSO. Em Cambridge Phoenix é usado tb em batch.
- existe um erro ainda não corrigido na facilidade de alocação dinâmica da IBM, que, quando acionado, derruba o sistema operacional.
  (peço desculpas pelo tempo que levou para identificar o problema).

Os próximos passos

- o desenvolvimento de uma linguagem de controle com facilidades pelo menos tão amplas quanto do Phoenix. Para facilitar o uso, esta deve se assemelhar a uma linguagem de programação. Em particular queremos ver:

  1. procedures com variáveis locais tipadas, que só aceitariam parâmetros com valores de tipo especificado.
  2. estruturas de controle estruturadas.

Observamos que não podemos insistir na aplicação de regras de "scope" de linguagens como ALGOL. Isto se deve à flexibilidade, necessário para permitir o usuário num terminal criar novas variáveis a qualquer hora.

Imaginamos que os procedures poderiam ser pré-compilados, e o que seria executada seria uma mistura de procedures compilados, e procedures e comandos interpretados.

como desenvolvimento natural, embora ambicioso, postulamos a integração de linguagens de controle e de programação com o resultado específico de generalizar o interface externo de programas permitindo que este passem estruturas complexas de dados entre eles.

"Situação Atual do Sistema de Documentação do Projeto de Pesquisa"

Pedro L.Steinbruch

É desnecessário comentar a necessidade de uma documentação adequada para o desenvolvimento de um projeto e posterior utilização do produto gerado.

Neste sentido, julgamos necessário padronizar certos procedimentos de programação e atividades complementares objetivando gerar documentos que permitam que se analise, a qualquer momento, a situação atual do projeto, ou, o "significado" de cada programa produzido.

Deste trabalho resultou um manual de procesimentos assim composto:

1. Normas de Programação
2. Normas de Documentação
   - 2.1. De Sistemas
   - 2.2 De Atividades
   - 2.3 De Programas
     - 2.3.1 - Interna
       - 2.3.1.1. Identificação de Programa
       - 2.3.1.2. Identificação de Instruções
       - 2.3.1.3. Identificação de Variáveis
     - 2.3.2 - Externa
       - 2.3.2.1. Manual de Especificação
       - 2.3.2.2. Manual de Implementação
       - 2.3.2.3. Manual de Utilização

A parte relativa a documentação de programas (2.3) já está concluída.

No que diz respeito a documentação de atividades (2.), já foram desenhados todos os formulários necessários, mas ainda não foram concluídas as instruções de preenchimento de cada um

deles.

Com relação a documentação de sistema (2.1), pesquisas ainda estão sendo feitas pois temos interesse que esta parte seja adequada a sistemas de banco de dados e com um nível de operacionalidade bem alto.

As normas de programação (1), foram parcialmente escritas e dentro em breve estarão concluídas.

Acreditamos que o manual de procedimentos poderá ser entregue para teste em fins de dezembro.

Em seguida pode ser encontrada a bibliografia consultada até o momento.

Bibliografia:

1 . LONDON,KEITH R. - Documentation Standards - AUERBACH- 1974
2 . VAN DUYN, J. - Documentation Manual - AUERBACH - 1972
3 . CLARK, FRANK J. ET ALLI - Business Systems and Data Processing Procedures - Prentice-Hall - 1972
4 . WALSH, DOROTHY A. _ A guide for software documention -Mc Graw-Hill - 1969
5 . GLEIM, GEORGE A. _ Eletronic data processing systems and procedures-Prentice-Hall - 1971
6 . BRANDON, DICK e GRAY, MAX - Project control Standards - Fetrocelli Charter - 1970
7 . Hed, Sven R. - Project control manual
8 . IBM - HIPO - A design aid and documentation technique - GC 20-1851
9 . LUCENA, CARLOS J. - Manual para o projeto e documentação de sistemas de programação - DI/PUC-RJ - 1976
10. MIKELSONS,M e WLADAWSKY,I - On the formal documentation of programs IBM T.J. Watson Research Center - 1976
11. TEICHROEW,D. - Computer aided documentation overview - Ann Arbor, Univ. of Michigan, 1975
12. LLOYD, L. e GERBRANDT, L. - UBC documentation : a guide to documentation procedured by UBC computing center-Vancouver, Univ. of British

"Data Structure Design"

Prof. Frank Tompa*

The design of data structures is central to the area of computer programming and therefore deserves much attention. In parallel with recent development in programming methodology , there has been some study into frameworks for data base design [Senko 73, Sundgren 75, Tsichritzis 77]. One framework which is particularly useful for the design of data structures consists of five levels [Tompa 77]: the data reality, that is the informa - tion as it actually exists; the data abstraction, the model of reality containing relevant data only; the information structure, the specification of the data to be represented explicitly and the algorithms for recovering the implicit data; the storage structure, the representation for the explicit relationships; and the structure enconding the machine representation for indivisi - ble units of data.

Because the transition from a data reality to a suitable data abstraction is extremely application dependent (perhaps falling within the realm of business administration or systems analysis), the task of data structure design can be stated as follows: given a data abstraction representing a particular application, implement it (that is, design suitable information structure, storage structure, and structure enconding). This task can be decomposed into two areas of research: the development of languages for representation independent specifications and the development of tools for the subsequent choice of representations.

Within the first area there are two primary targets for specification language: the data abstraction and the information structure. Whereas the transition from data reality to abstrac - tion is assumed to have been done by others, the final description

* University of Waterloo

of the data abstraction must be made to be suitable for further design of the data structure. Research is needed to find appropriate models for such description, perhaps based on graph theory, logic, or algebra. The model must also be capable of describing how the data is to be used (for example, which operations are to be performed and with what frequency), as well as merely serving as a static description.

The description of an information structure involves the specification of data types to represent each explicit data entity or relationship. The first step is to compile a (large) set of common data types such as tuples, sets, sequences, and arrays [Hoare 72] that can serve as buliding blocks for describing information structures. In addition, techniques must still be developed and improved for describing other (abstract) data types, that is, those that are not adequately describable in terms of a given fixed set of types [Liskov 75, Guttag 77] . As an example, most conventional programming languages include facilities for describing a node from a tree of integers as follows:

type node = (integer VALUE; reference(node) LEFT,RIGHT);

but not for describing what is meant by a "tree" (It is interesting to observe that the above definition could as easily be for a node of a doubly-linked list rather than for a tree.) Research into abstract data types must be accompanied by further research into related topics in programming languages, for example data type conversion [Geschke 77] and control structures design [Shaw 77] .

As stated above, the second area of research is concerned with the actual process of data structure design, rather than with data structure description.

The first level for design is the transition from data abstraction to information structure. This involves the separation of data relationships into those which are to be made explicit from those to be made implicit. It may require the development of methodologies for creating new types or for discovering appropriate superpositions of given types [Lucena 77], and it will require the design to be

made in terms of algorithmic efficiency [Aho 74].

The other level is the design of a storage structure (the choice of structure encoding is often fixed for an application). This can be made on the bais of efficiency [Tompa 76] or on the basis of reliability [Taylor 77]. The storage structure can be viewed as a superposition of representations for the data types composing the information structure, and the components can therefore be chosen from a library of potential representations. The mechanisms necessary for creating and maintaining such a library form a further related area for research.

[Aho 74] - A.V.Aho, J.E. Hopcroft,J.D.Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

[Geschke 77] C.M. Geschke,J.H. Morris,Jr., E.H.Satterthwaite, "Early experience with Mesa", presented at Conf. on Lang. Des. for Reliable Software, March 1977

[Guttag 77] J. Guttag, "Abstract data types and the development of data structure, Comm ACM 20,6 (Fune 1977), pp. 396-404.

[Hoare 72] C.A.R. Hoare, "Notes on data structuring", in Structured Programming (Dahl.Dijkstra.Hoare,ed.), Academic Press 1972.

[Liskov 75] B. Liskov, S. Zilles, "Specification Techniques for abstract data types", IEEE Trans. on Soft. Eng. 1, 1 (March 1975), pp. 7-19

[Lucena 77] C.J.Lucena, T.Pequeno, "Assigning programs to meaning a case study," in preparation.

[Shaw 77] M.Shaw, W.A.Wulf,R.L.London, "Abstraction and verification in Alphard: defining ans specifying iteration and generators", presented at Lang.Des. for Reliable Software, March 1977.

[Sundgree 75] B. Sundgren, Theory of Data Bases, Mason and Charters, 1975.

[Taylor 77] D.J. Taylor, "Enhancing software reliabity through the use of robust data structures, "Ph.D. thesis, Dept of Comp. Sci., Univ. of Waterloo, 1977

[Tompa 76] F.W. Tompa, "Choosing on efficient internal schema", Systems for Large Data Bases (Lockeman and Newhold, ed.) North Holland, (1976), pp. 65-77

[Tompa 77] F.W. Tompa, "Data structure design", Data Structures in Pattern Recognition and Computer Graphics (Klinger, ed.) Academic Press., 1977.

[Tsichritzis77] D.Tsichritzis, A. Klug, (ed.), "The ANSI/X3/SPARC DBMS Framework", CSRG Tech. Note 12, Univ. of Toronto (July 1977).

[Senko 73] - M.E. Senko, E.B. Altman, M.M. Astrahan, P.L. Fehder, "Data structures and accessing in data base systems - DIAM", IBM Sys. J. 12, 1 (1973) pp. 30-93

"Tipos Abstratos em Correção de Programas: uma visão algébrica"

Prof. Paulo A. Veloso

O uso de tipos abstratos de dados pode ser uma importante ferramenta no desenvolvimento sistemático de programas confiáveis. Pois, pode-se formular um programa de alto nível operando com tipos abstratos, deixando sua implementação para uma etapa posterior quando ficar mais claro que estruturas devem ser empregadas para tal. Assim, idéias de programação estruturada refinamentos sucessivos, etc. podem ser aplicadas não só a parte do controle como também a de dados.

A situação poderia ser descrita assim:

1. escreve-se um programa P operando com o tipo abstrato $A = < A_i, f_j >$;
2. escolhe-se uma representação mais "concreta" $C =< C_k, g_\ell >$;
3. cada f_j é então descrito por uma rotina R_j sobre C.

O correspondente problema de correção neste caso é naturalmente dividido em três subproblemas:

(a) - correção do programa P sobre A;
(b) - correção da representação de A por C;
(c) - correção de cada rotina R_j

Os problemas (a) e (c) são análogos ao original, embora mais simples. A parte (b) depende da especificação dos tipos empregados. Uma maneira de especificar um tipo abstrato de dados consiste em dar as operações permissíveis e axiomas aos quais elas devem satisfazer. Frequentemente, tais axiomas podem ser postos na forma de equações, sendo conveniente encarar o tipo como uma álgebra (heterogênea). Um tipo abstrato de dados será então uma álgebra inicial na classe das álgebras com estas operações satisfazendo aos axiomas. O problema de correção da representação consiste em mostrar que a versão concreta C e a abstrata A

são isomorfas. É justamente aí que métodos e conceitos algébricos são úteis, sugerindo diversos caminhos para este objetivo.

Uma variante da situação descrita consiste em se começar com tipos em que nem todas as especificações do problema podem ser expressas. A medida que os tipos vão sendo refinados por descrições mais concretas, são adicionadas mais especificações, até a obtenção de programas e tipos básicos satisfazendo a todas as epecificações do problema. Um exemplo disto é o apresentado na palestra de Tarcísio Pequeno. Atualmente estamos trabalhando em alguns aspectos algébricos deste método que parecem ser úteis para programação .