PUC

Série: Monografias em Ciência da Computação

- Fascículo Especial -

" SOFTWARE DE APOIO PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO" E " SOLUÇÃO NUMÉRICA DE EQUAÇÕES DIFERENCIAIS" ; RELATÓRIO DE PROJETOS DE PESQUISA: I

editado por

Rosane Teles Lins Castilho

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

- Fascículo Especial -

Editor da Série: Michael F. Challis Setembro, 1977

"SOFTWARE DE APOIO PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO" e " SOLUÇÃO NUMÉRICA DE EQUAÇÕES DIFERENCIAIS"; RELATÓRIO DE PROJETOS DE PESQUISA: I*

editado por

Rosane Teles Lins Castilho

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro

*Projetos parcialmente financiados pelos contratos TC 0060/77 e TC 222.0059/77 do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)

PREFÁCIO

Este é o primeiro relatório dos projetos de pesquisa do Departamento de Informática da PUC/RJ, a saber "METODOLOGIA PARA A SÍNTESE DE SISTEMAS DE INFORMAÇÃO", de responsabilidade do professor Carlos José Pereira de Lucena, e "DESENVOLVIMENTO E IMPLEMENTAÇÃO DE METODOS NUMÉRICOS PARA A SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS E ORDINÁRIAS", de responsabilidade do professor Michael Anthony Stanton.

Os projetos foram iniciados formalmente em abril de 1977, sob o patrocínio do Conselho Nacional de Desenvolvimento Científico e Tecnológico, sob os contratos TC 0060/77 e TC 2222.0059/77.

São apresentados em seguida os relatórios de andamento dos referidos projetos.

R.T.L.C.

S U M Á R I O

1 - SOFTWARE DE APOIO PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

1.1 - Objetivo Geral

Os projetos de pesquisa estão inseridos nos programas do departamento . O denominador comum desses projetos é o fato de que todos são projetos experimentais (envolvem implementações) voltados para o desenvolvimento de sistemas de "Software" que visam facilitar o uso de computadores por parte de engenheiros e especialistas em sistemas de informação.

Os projetos se constituem em implementações sugeridas por propostas constantes de várias pesquisas já desenvolvidas,ou em desenvolvimento, no Departamento. Tais sistemas de "software",uma vez implementados,permitirão a avaliação experimental dos conceitos que eles encerram, recomendando possivelmente a transformação desses protótipos em produtos de uso generalizado.

É responsável pelo projeto o professor Carlos José Pereira de Lucena.

1.2 - Relatórios

De acordo com o objetivo geral do projeto descrito em 1.1, definiram-se oito sub-projetos que se desenvolveram paralelamente e cujos relatórios são apresentados em seguida.

1.2.1 - JACKDAW

Responsável: Prof. Michael F. Challis

1.2.1.1 - Objetivo do projeto

O objetivo deste projeto pode ser resumido em "desenvolvimentos posteriores do pacote do banco de dados JACKDAW que têm a finalidade de prover um sistema adequado para a manutenção de bancos de dados de porte médio e grande, em um ambiente de múltiplos usuários em linha".

Mais especificamente, podem-se considerar separadamente quatro dos seus objetivos, dos quais os três primeiros foram visados durante esse período.

1 - Instalação do JACKDAW no IBM/370-165 da PUC/RJ.

2.- Melhorias da versão corrente (conhecida como JACKDAW I)

3 - Projeto da próxima versão (JACKDAW II)

4 - Implementação do JACKDAW II.

1.2.1.2 - Histórico

Antes de ser instalado na PUC/RJ o pacote do banco de dados JACKDAW [1] vinha sendo utilizado desde julho de 1973, na "University of Cambridge Computing Service" provendo suporte a um banco de dados administrativo. Em outubro de 1975, uma segunda maior aplicação foi iniciada, na qual detalhes das salas e prédios da universidade deveriam ser registrados a fim de se manter uma imagem atualizada das acomodações disponíveis na universidade. Ambas aplicações foram caracterizadas por necessidades totais de armazenamento relativamente pequenas (2 M bytes para a primeira, 4 a 5 M bytes para a segunda aplicação), mas ambas necessitaram de representações de relações complexas entre as várias entidades de informação.

Ganhou-se muita experiência durante a realização desses projetos , e é essa experiência que se pretende utilizar no projeto do JACKDAW II.

Melhorias na versão corrente foram feitas principalmente na área de manipulação de erros e na habilidade de se lidar com volumes maiores de dados.

As outras seções deste relatório descrevem em detalhe os progressos feitos com relação aos três primeiros objetivos identificados acima.

1.2.1.3 - Instalação do JACKDAW

Já foi completada: o sistema está disponível tanto de maneira "on-line" através do sistema Phoenix (em processo de desenvolvimento pelo professor Michael A. Stanton) ou em "batch" utilizando o IBM JCL (Job Control Language) padrão.

As especificações do sistema estão disponíveis em formato para leitura em máquina (cópias na biblioteca) e um pequeno banco de dados exemplo foi projetado para ilustrar as várias facilidades disponíveis.

Esforços substanciais têm sido feitos no projeto de procedimentos para a compilação e construção de vários componentes do pacote, de maneira objetiva, de forma que tornou-se relativamente simples corrigir ou modificar qualquer aspecto do sistema. Isto é particularmente importante em vista do tamanho do sistema (cerca de 25.000 linhas de código fonte). A linguagem Phoenix provou ser de grande utilidade nesta área.

As mesmas técnicas foram utilizadas para gerar um "distribution Job", de forma que tornou-se relativamente fácil instalar e testar o sistema em um computador similar. De fato, este sistema já foi instalado com sucesso em um IBM/370-145, desta maneira.

Os componentes básicos do JACKDAW I são:

- Núcleo: - que consiste de uma biblioteca de procedimentos do BCPL através dos quais pode-se ter acesso e modificar o conteúdo do banco de dados existente por programas de aplicação.

- DBREFORM: - é um utilitário para criar novos bancos de dados ou para alterar a estrutura de algunsjá existentes (ex.: acrescentar no

vos campos ou relações)

- SLANG: - provê uma interface de comando de baixo nível. (Muitas aplicações podem ser feitas com SLANG, não havendo, portanto, a necessidade de se escrever programas em BCPL; particularmente, a maioria dos bancos de dados podem ser carregados utilizando SLANG).

- ODIUM: - que apresenta uma linguagem simples para interrogações e atualizações interativas.

- OBLIST: - um gerador de relatórios

- UTIL: - um programa utilitário que provê serviços de manutenção diversos.

1.2.1.4 - Aperfeiçoamento para o JACKDAW I.

Duas áreas em breve vão exigir atenção:

1 - Tratamento de erros.

No momento, o tratamento "default" de erros consiste simplesmente em imprimir um número identificador do erro: isto deverá ser substituído por mensagens mais informativas. As facilidades já providas para a intercepção de erros por parte do usuário necessitam ser revistas e expandidas.

2 - Processamento mais eficiente de bancos de dados de grandes porte. No momento, várias tabelas são mantidas na memória principal contendo uma entrada para cada bloco do banco de dados: assim, na medida em que o banco de dados cresce, crescem também as necessidades da memória principal. Um novo esquema para a paginação dessas tabelas vem sendo projetado, o qual , quando implementado, permitirá o processamento de bancos de dados maiores, sem aumentar as necessidades da memória. Este

esquema também dará suporte à representação de vários "bancos de dados lógicos" dentro de um arquivo em disco (veja a seção "acesso por múltiplos usuários") e assim o mesmo código será usado na eventual implementação do JACKDAW II.

1.2.1.5 - Projeto do JACKDAW II

O JACKDAW II consistirá somente dos quatro componente relacionados a seguir:

- Núcleo: um conjunto de procedimentos, tão compatíveis quanto possível com aqueles do JACKDAW I.

- DBREFORM: para criar ou modificar a estrutura do banco de dados.

- SLANG 2: para incluir as características correntemente providas por SLANG, ODIUM e DBLIST.

- UTIL: serviços de manutenção diversos.

As facilidades básicas a serem oferecidas pelo núcleo já foram decididas, e a linguagem na qual a estrutura do banco de dados correspondente deverá ser definida (ex.: a entrada para DBREFORM) já foi projetada.

Uma lista das novas características do JACKDAW II é fornecida abaixo juntamente com uma breve explicação:

- Novas características

1 - Generalização do conceito de "chave"

Um campo primitivo de qualquer tipo pode ser especificado como uma "chave" para entradas de uma classe, ou como uma chave para um campo de ligação relacioando. Uma chave pode ser também especificada como consistindo de uma chave "maior" seguida por uma ou mais chaves "menores", todas elas sendo campos primitivos separados.

Ex.:
```
add class building
begin
        int code
        string name
        key is code
end

add class room (
begin
        int floor
        int number
        int area
        no key
end

add link (building,
          Rooms (key is floor, number))
          from room to building
```

Aqui, entradas "prédio"(building) são identificadas por seus códigos, e entradas "sala" (room) não tem identificação. Cada "prédio" é ligado a todas as "salas" naquele prédio, e estas ligações são ordenadas pelo "pavimento" (floor), chave maior, e pelos campos de "número" (number), chave menor, das entradas de "sala".

2 - ÍNDICES SECUNDÁRIOS:

A chave para uma entrada provê o meio principal para o acesso direto a essa entrada, e a ordenação física das entradas é essecialmente determinada por suas chaves.

Um índice secundário permite que se tenha acesso direto a uma entrada citando-se o (s) valor(es) de um ou mais campos primitivos

Ex.: se "nome" (name) for definido como uma chave para um índice secundário para entradas "prédio", pode-se ter acesso as entradas "prédio" diretamente indicando-se seu "nome".

Contrariamente as chaves principais, os valores das chaves secundárias podem ser duplicados e alterados.

3 - CADEIAS DE COMPRIMENTO FIXO:

Um novo tipo de campo primitivo, "byte(n)", será fornecido. Isto também permite uma representação econômica das cadeias curtas de um comprimento fixo.

4 - CAMPOS DE LIGAÇÃO DE ELEMENTOS SIMPLES:

Campos de ligação que sabe-se incluem no máximo um elemento de ligação (ex.: o campo de ligação "prédio" numa entrada de "sala") podem ser especificamente declarados. Isto permitirá aos pacotes produzir uma representação mais compacta destes campos.

5 - CAMPOS GRUPADOS

Um campo grupado é essencialmente um conjunto de campos primitivos que podem aparecer repetidamente dentro de uma entrada.

Ex.:
```
new class building
begin
    int code
    string name
    group rooms
     begin
         int floor
         int number
         int area
         key is floor, number
     end
   key is code
end
```

Da mesma forma que no item 4, esta nova facilidade permite implementações mais eficientes de situações que frequentemente surgem na prática.

Um campo grupado comporta-se sob vários aspectos como um cam-

po de ligação; uma diferença importante é que não é possível a outras entradas ligarem-se aos componentes de um grupo.

6 - CAMPOS DE UNIÃO:

Estas características permitem a definição de um conjunto de entidades similares com uma classe simples.

```
Ex.: new class person
     begin
          string name
          int age
          oneof (professor begin
                             string subject
                             int  sizeofclass
                           end
                 Student   begin
                             bool graduate
                             date yearofenty
                           end
                )
          key is name
     end
```

Cada entrada da classe "pessoa" (person) é ou um "professor" ou um "aluno " (student).

Um campo de ligação de alguma outra classe para "pessoa" pode conter elementos de ligação referindo-se as estradas "professor" ou "aluno"; por outro lado, pode-se insistir que somente entradas "aluno" sejam referenciadas definindo-se o campo de ligação como sendo para a classe "aluno".

- Acesso por múltiplos usuários:

Muita consideração tem sido dada aos problemas inerentes ao acesso simultâneo ao banco de dados por muitos usuários, e alguns resultados

preliminares já foram obtidos. Estes resultados foram baseados na idéia de se representar, ao mesmo tempo, várias ocorrências de um banco de dados em um único arquivo em disco. Cada ocorrência corresponde ao estado do banco de dados naquele momento em particular e para um usuário em particular, e pode ser posteriormente combinada com outra ocorrência, ou simplesmente ser eliminada. Mais detalhes sobre esta abordagem são fornecidos em 1.2.1.4, seção 2.

- SLANG 2

O objetivo do SLANG 2 é prover uma linguagem unificada com a qual um usuário tenha acesso ao banco de dados.

Alguns indícios demonstram, no momento, que é apropriada uma abordagem a dois níveis. O nível mais baixo proverá acesso a quase todos os procedimentos de interface do núcleo, e podem-se imaginar muitos programas de aplicação que utilizem este sub-conjunto. O nível mais alto permitirá interrogações e atualizações interativas diretas, e proverá macro facilidades que possibilitarão a criação de comandos destinados a uma aplicação particular.

Alguns progressos têm sido feitos no projeto do SLANG 2, e alguns exemplos de seu formato tentativo são fornecidos abaixo:

baixo nível:

```
.B←BUILDING Ø97
NAME OF .B  := PHYSICS
FOR . RL ← EACH ROOMS OF .B DO
$C . R ← FOLLOW . RL

     WRITE ("ROOM", NUMBER OF .R,
              "ON FLOOR",FLOOR OF. R
               "HAS AREA", AREA OF . R
     RELEASE .R $)
RELEASE  .B
```

alto nível: TYPE NAME OF BUILDING Ø97
BUILDING Ø99 NAME = PHYSICS
DELETE ALL ROOMS OF BUILDING 172
TYPE (FLOOR,NUMBER) OF
ROOMS FOLLOW WHERE AREA < 100
OF BUILDING 265

1.2.1.6 - Publicações e seminários

Um trabalho sobre o JACKDAW foi apresentado no "4º Seminários sobre o Desenvolvimento de Software e Hardware ", realizado em Belo Horizonte (MG), em julho de 1977, e encontra-se disponível como Relatório Técnico [2].

REFERÊNCIAS:

[1] -CHALLIS,M.F. The JACKDAW data base package. Cambridge (Engl.) University of Cambridge, Computer Laboratory, 1974. TR-1.

[2] -_____. Integrity techniques in the JACKDAW data base package. Rio de Janeiro, PUC, Depto. de Informática, 1977. Monogrs. em Ciência da Computação Nº 9/77

1.2.2 - Implementação da álgebra de relações quocientes para manipulação de bancos de dados.

Responsável: Prof. A. L. Furtado

1.2.2.1 - Descrição do projeto

O projeto visa a implementação da álgebra de relações quocientes [1], como linguagem intermediária para a manipulação de bancos de dados.

O projeto se divide em duas partes:

a. implementação de um pacote básico;
b. implementação da álgebra propriamente dita.

O pacote básico é um conjunto de procedimentos, programados em PL/I, para a manipulação de três estruturas:

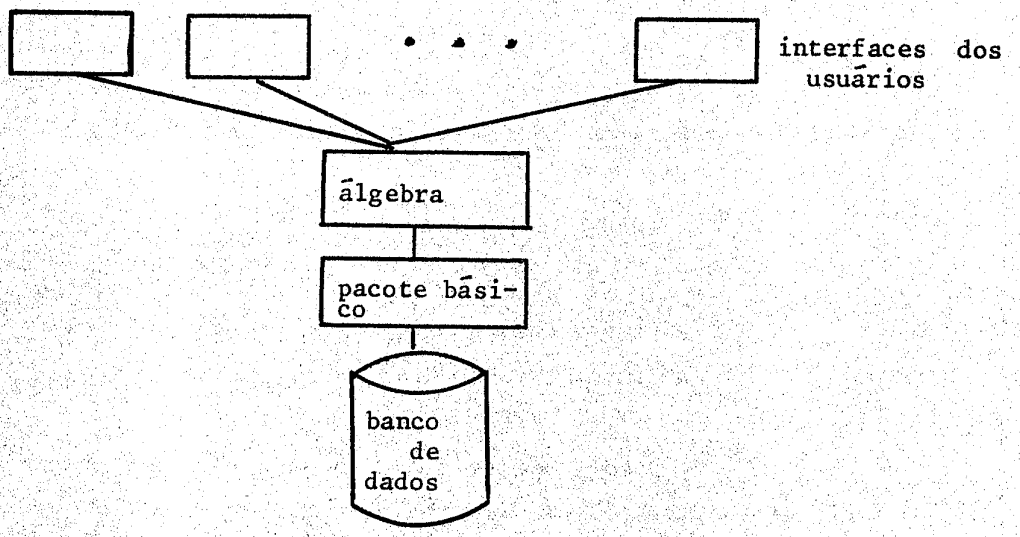
- arquivos de dados;
- arquivos de elos ;
- arquivos de inversões

Tais estruturas residirão em disco magnético. De vez que o pacote está sendo implementado em um IBM/370/165 os procedimentos PL/I interagem com o sistema operacional dessa máquina e utilizam um de seus métodos de acesso (indexado sequencial).

Em princípio o pacote básico poderá servir à implementação de sistemas de bancos de dados que sigam qualquer um dos três principais modelos - relacional, hierárquico ou redes. Entretanto o esforço inicial será dirigido para o modelo relacional, cujas vantagens tem sido reconhecidas por um grande número de autores.

[1] A.L.Furtado e L.Kerschberg - "An Algebra of Quotient Relations" - Proc. SIGMOD(1977).

A álgebra a ser implementada a partir do pacote básico se enquadra no modelo relacional. Foi apresentada no início como linguagem intermediária porque será usada para a elaboração de diversas interfaces, as quais se destinarão a diversas classes de usuários. A figura abaixo representa a arquitetura adotada.



Também a álgebra tomará a forma de procedimento em PL/I , os quais portanto invocarão os procedimentos do pacote básico. Para cada interface de usuário será elaborado um compilador que traduzirá da linguagem da interface para programas em PL/I contendo invocações aos procedimentos da álgebra. O desenho e implementação das interfaces não faz parte do corrente projeto.

Uma vez concluida a implementação do pacote básico e da álgebra estas servirão como ponto de partida para diversos trabalhos sobre bancos de dados.

Não se espera entretanto que o produto final seja adequado para o processamento de rotina de grandes bancos de dados. O projeto visa à criação de um protótipo, a partir do qual se desenvolverão outras pesquisas do Departamento (especialmente as diversas interfaces).

1.2.2.2 - Situação do projeto

Os trabalhos foram iniciados em abril do corrente ano.

O desenho do pacote básico foi concluído e em seguida foi iniciada sua programação. Em paralelo foram esquematizados os algoritmos para as operações da álgebra, de modo a garantir que o pacote básico seja realmente adequado para a implementação da álgebra.

Os procedimentos para a criação das estruturas, para a ativação e desativação do banco de dados, e três módulos auxiliares estão completamente programados e já passaram pela primeira fase de testes.

Os procedimentos de inserção e remoção de elementos nas estruturas estão sendo programados, e alguns já se encontram em teste.

Os procedimentos de seleção de elementos nas estruturas estão todos programados e a fase de testes se iniciará na semana do dia 29/8.

Cerca de dez procedimentos ainda não foram iniciados.

Prevê-se para 15/9 um teste conjunto envolvendo criação, inserção e remoção (e talvez também seleção). Os testes conjuntos são uma fase especial em projetos que adotam a técnica de programação modular: primeiro se testa cada módulo em separado para depois testá-los em conjunto.

A programação dos algoritmos da álgebra ainda não foi iniciada.

Para acelerar o andamento do projeto tentaremos obter mais dois alunos de mestrado, que participariam de tarefas de programação.

Os dois alunos de mestrado que já participam do projeto farão suas teses de mestrado sobre:

- desenho de documentação de pacote básico;
- avaliação do desempenho do pacote básico;

Para a orientação da segunda tese irá colaborar o Professor K. Sevcik, da Universidade de Toronto, cuja chegada está prevista para 10/9.

O aluno de doutorado que participa do projeto está estudando diversos aspectos formais e práticos da álgebra e de sua implementação, a serem relatadas em artigos técnicos.

1.2.3 - UMA LINGUAGEM DE ESPECIFICAÇÃO DE SISTEMAS DE INFORMAÇÃO

Responsável: Carlos José Pereira de Lucena (Coordenador do Programa)

1.2.3.1- Histórico do Projeto

O objetivo deste projeto é a implementação de um sistema de apoio ao desenvolvimento de sistemas de processamento de dados, através de uma linguagem de especificação de sistemas. Dividindo a tarefa de desenvolvimento de um sistema em três etapas, isto é, análise de requisitos, projeto lógico e projeto físico, o sistema a ser implementado se destina ao projeto lógico somente. A análise de requisitos não é parte do escopo deste projeto. O projeto físico poderá beneficiar-se futuramente deste sistema, esperando-se que possam ser implementadas facilidades para detalhar em maior profundidade alguns aspectos do projeto físico. É uma possibilidade, porém, e não um objetivo.

Os três primeiros meses do projeto foram dedicados a uma análise dos sistemas semelhantes existentes, para determinar suas propriedades básicas. Foram estudadas linguagens interpretativas, como PSL, MIL,VDL, e linguagens gráficas como PETRI NETS, bi-graphs e LOGOS. Verificou-se que estas linguagens são equivalentes em vários aspectos, podendo ser em princípio modeladas entre si. Concretamente estudou-se um modelo de PSL em LOGOS. O único ponto que apresentou dificuldades foi a modelagem de estruturas de dados de PSL em LOGOS. Através de um modelo gráfico para estruturas de dados chamado v-graphs, adotado como extensão do LOGOS, foi possível modelá-las.

Após esta fase foi feito um balanço das propriedades e confrontro com os objetivos da lingaugem a projetar. Um deles era a possibilidade proceder a uma prova quase formal do modelo, com base em asserções descritas em uma parte da linguagem. Concluiu-se não haver nas linguagens atuais facilidades para provas.

Finalmente elaborou-se uma lista de requisitos considerados importantes, a partir das propriedades dos sistemas existentes, estudos publicados e discussões. Daí partiu-se para um desenho tentativo da linguagem, o qual está descrito mais adiante.

1.2.3.2 - Requisitos da Linguagem

- Nível Linguístico de sistemas
- Estruturas de dados "genéricas" e operadores associados.
- Linguagem pouco ou não procedural.
- Auto documentável
- Descrição de asserções embutida na linguagem
- Paralelismo
- Referências associativas e estruturas de dados
- Verificável, isto é, possibilidade de obter provas semi formais.

O requisito da verificação é importante em vista do consenso atual de que a propriedade mais importante de um software é a correção. A auto documentação é considerada importante por facilitar a comunicação do trabalho do projetista do sistema tanto entre membros de uma equipe de projeto como entre projetista e usuários não treinados em processamento de dados. A documentação consome também grande parte do tempo dedicado ao projeto, pelos sistemas atuais, e, embora importante, é considerada desinteressante como tarefa. Além, disso, existe o problema de manter uma documentação atualizada durante as várias etapas do projeto, que se pode resolver adequadamente só com a automação da tarefa.

1.2.3.3 - Descrição da Linguagem

A linguagem não possui ainda uma descrição formal, embora alguns detalhes já estejam desenvolvidos. Basicamente é composta de três partes:

- descrição das estruturas de dados
- descrição de processos
- descrição de estados (asserções)

A primeira descreve as estruturas de dados do sistema. Os seguintes tipos de objetos são considerados:

. Conjuntos
. Entidades
. Grupos
. Elementos

Os conjuntos compõe-se de entidades. As entidades compõem-se de grupos e elementos. Os grupos compõem-se de elementos. Os elementos são o nível mais baixo das estruturas e contêm valores. Os grupos servem a duas finalidades: agrupar lógicamente elementos relacionados e indicar ao sistema a possibilidade de várias ocorrências do grupo. As entidades podem ser identificadas com o conceito de registro e os conjuntos, com arquivos . Mas esta identificação não é imposta.

A descrição de processos inclui processos, eventos e condições. Os processos são os transformadores de estruturas de dados, e recebem, atualizam e criam novas estruturas. Os eventos estão relacionados a início ou fim de processos, ou a condições. Podem por sua vez deflagrar execução de processos. As condições são normalmente descritas dentro dos processos e causam eventos, quando verdadeiras.

A descrição de estados faz o papel de asserções sobre o sistema. Cada estado está associado a um ponto no fluxo de controle do sistema exterior aos processos, e neles são descritos certas condições que as estruturas de dados e eventos devem satisfazer. Basicamente descreve-se o que contêm cada elemento, e que certos eventos ocorreram. Como exemplo de uma descrição de estado, vejamos a seguinte:

```
ENTIDADES USADAS pedido, preços
ENTIDADES CRIADAS, fatura
pedido:

   cliente = "a"
   itens:

      item = "x"
      quantidade = "1Ø"
```

```
preços:

   item = "x"

   preço unitário = "2"

Fatura:

   cliente = "a"

   discriminação:

      ( item = "x"

         quantidade = SOMA DE "1Ø"
         preço-unitário = "2"
         preço-total    = "2" x quantidade ) PARA CADA item

   valor-fatura = SOMA DE preço-total
```

Cada elemento, grupo e entidade é descrito uma única vez em cada descrição de estado. A sua definição completa entretanto é feita na descrição das estruturas de dados do sistema. O formato da descrição de um elemento na descrição de estado é:

elemento = expressão

onde a expressão pode ser uma fórmula aritmética, ou um if then else. Todos os elementos descritos são totalmente qualificados pelo nome da última entidade e grupo anteriores à sua descrição. Quando se quizer especificar uma qualificação é necessário fazê-lo explicitamente.

Na descrição do elemento, isto é, na expressão, além de outros elementos podemos usar também valores entre aspas. Estes valores são exemplo dos valores possíveis que o elemento recebe.

Os grupos "itens" e "discriminação" podem ocorrer várias vezes para cada ocorrência de "pedido" e "fatura", respectivamente. Eles correspondem a conjutnos de n-uplas de elementos. As entidades são n-uplas de elementos ou grupos. Cada elemento tem um valor associado a cada ocorrência da entidade no conjunto de que faz parte. Este conjunto de valores associa-

dos podem ter eventualmente valores repetidos. Através de um índice associado a cada valor, podemos distinguir cada valor repetido. Este índice terá sempre valor 1 para valores não repetidos, e 1, 2, 3, ... para valores repetidos. Dessa forma evitamos valores repetidos no conjunto, o que coincide com a noção matemática de conjunto. Os conjuntos serão chamados conjuntos - valores.

As entidades são caracterizadas como USADAS ou CRIADAS. Os elementos das entidades usadas são descritos por exemplos, sempre. Estes exemplos por sua vez são usados nas descrições de elementos de entidades criadas. O conjunto-valor de um elemento de entidade criada é definido como a interseção dos conjuntos-valores dos elementos das entidades usadas em cuja descrição sejam usados os mesmos exemplos. No exemplo de descrição de estado fornecido, o elemento item é descrito por um exemplo, "x", tanto nas entidades usadas (pedidos, preços) como nas criadas (fatura). Isto significa que o conjunto-valor deste elemento em fatura é a interseção dos conjuntos-valores dos elementos item de pedido e item de preços.

O operador PARA CADA elemento-chave atua sobre os conjuntos-valores dos elementos. O efeito é tomar como novo conjunto-valor o conjunto quociente do anterior em relação ao elemento chave. O novo conjunto-valor portanto só terá valores associados a valores distintos do elemento-chave .

O operador SOMA DE é associado ao operador PARA CADA abrangente. Significa que os valores associados aos valores iguais do elemento chave são somados e substituem estes valores. Exemplo:

item	quantidade
x	10
x	20

Após a SOMA DE ... PARA CADA:

item	quantidade
x	30

A especificação dos eventos ocorridos não apresenta problemas na descrição de estado.

Terá basicamente o seguinte formato:

```
PROCESSO nome
EVENTOS DE INICIO lista de eventos
ENTIDADES USADAS lista de entidades
ENTIDADES CRIADAS lista de entidades
EVENTOS DE FIM lista de eventos
TEXTO:
  comandos
FIM DO TEXTO
```

O procediemtno usará uma linguagem "procedural" com operadores sobre os conjuntos de entidades e formulas aritméticas. Possivelmente estruturas de controle como IF THEN ELSE e DO WHILE serão introduzidas.

Através deste resumo procurou-se descrever o estado atual da linguagem de especificação a implementar.

1.2.4 - FERRAMENTAS PARA O PROJETO E CONSTRUÇÃO DE SISTEMAS DE INFORMAÇÃO APOIADOS EM BANCO DE DADOS

Responsável: Rubens Nascimento Melo

1.2.4.1 - INTRODUÇÃO

Pode-se dizer que os estudos e pesquisas na área de Banco de Dados tem relegado a segundo plano a integração do projeto do 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 não 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 estruturas 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 adoação do enfoque de Banco de Dados para o sistema de informação sendo desenvolvido, permite reduzir este problema ao problema de escolher as 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"

1.2.4.2 - Objetivos do Projeto:

Implementar o sistema (semi) automaticamente a partir da sua especificação conceitual pode ser colocado como 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 (estática) 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 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 programática (linguagem de consulta para o Banco de Dados do Sistema).

1.2.4.3 - Andamento do Projeto e Resultados Parciais

O projeto envolve alguns programadores e alunos de mestrado cujas teses saíram/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

têm sido usadas na construção de outras ferramentas. Por exemplo:

a) Processador de Macros de Proposito 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 pré-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 Banco 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 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 sido FORTS. Devido aos comandos DOWHILE, REPEAT, IF-THEN-ELSE, OERFORM, etca programação tem sido bastante facilitada. Os problemas relativos a manipulação de caracteres tem sido resolvidos 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, Rubens N. "On the mapping from the conceptual Specification to the internal model in Data Base Design" (a ser publicado)

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

[4] Melo, Rubens N., Schwabe,D. "Extending the control structures of FORTRAN via a macro generator" TR 1/76 - PUC-RJ

[5] Melo,Rubens N., "PM-Um processador de macros" Tese de mestrado - ITA - 1971

[6] Andrade,G.K., "Especificação de uma subliguagem 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., "Data Base Management System for PSA based on DBTG 71" ISDOS Working Paper Nº 88 Julho 1975

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

1.2.5 - META-ASSEMBLER

Responsável: Prof. Albrecht von Plehwe

O programa "Meta-Assembler" fez parte de um sistema que está sendo desenvolvido principalmente para ajudar em ensino e pesquisa na área de arquitetura de computadores, mas também para servir de sistema de programação no desenvolvimento de computadores reais. Este sistema deverá permitir, com um mínimo de tempo de adaptação, a compilação e execução simulada de programas escritos em linguagem Assembler de computadores reais ou projetados. Ele facilitará, assim, trabalhos tais como:

- comparação de jogos de instruções dependendo do tipo de processamento
- desenvolvimento de jogos de instruções para computadores novos ou micro programáveis
- programação de software básico para estes computadores

Um sistema com este objetivo pode ser implementado de várias maneiras (considerando a aplicação prioritária na área de ensino foi escolhida uma estrutura que modela os três passos de programas no computador: Compilação, Ligação-Edição, Execução. O sistema completo, portanto, será composto de três programas correspondentes: Meta-Assembler, Ligador-Editor, Emulador. O Meta Assembler traduz programas em linguagem Assembler do computador considerado para uma forma intermediária relocável; o Ligador-Editor combina vários programas, traduzidos separadamente, e forma um programa executável; o Emulador, finalmente, "executa" este programa, simulando uma máquina geral, microprogramada para executar as instruções do computador considerado.

Como primeiro programa a ser implementado foi escolhido o Meta-Assembler porque pode ser usado sem os outros dois componentes do sistema, em particular no desenvolvimento de software básico para computadores já existentes.

Todos os compiladores de linguagens Assembler executam muitas tarefas em comum, tais como criar tabelas de símbolos, avaliar expressões aritméticas ou compor informação binária a partir de campos codificados em forma simbólica. A idéia básica do Meta-Assembler é a de fazer um compilador Assembler ca

paz de traduzir a parte comum à maioria das linguagens Assembler e equipá-lo com a capacidade de absorver as regras adicionais da tradução para cada linguagem particular. Uma maneira natural e vantajosa de realizar esta capacidade é através de uma facilidade de macros; maneira natural porque não acrescenta nenhum elemento estranho a linguagem; maneira vantajosa porque representa uma capacidade muito poderosa da linguagem que está a disposição do programador, também, para outras finalidades.

No início do projeto foram estudadas as linguagens Assembler de vários computadores(PDP11, HP 2100, Variam 620, PDP 10, IBM 360/370) e o único Meta-Assembler que ainda existe, o Assembler SLEUTH II nos computadores da série UNIVAC (1100). Ficou claro que com os elementos da linguagem SLEUTH não será possível modelar em todos os detalhes compiladores de linguagem já existentes. Os dois obstáculos principais que foram identificados são:

- a transmissão dos parâmetros de macros por valor e não por substituição
- a maneira pouco sistemática de identificar tipos de dados em expressões, propriedade, aliás, de todas as linguagens Assembler estudadas com exepção do Assembler IBM 360/370.

Depois entrou-se na fase de definição da linguagem a ser usada no Meta-Assembler. Esta fase será concluída em cerca de 2 semanas de documentada em forma de um manual de usuário. A Linguagem oferecerá um conceito de macro bem mais poderoso que as "procedures" da linguagem"SLEUTH". Mesmo assim, continuam as dificuldades com a identificação dos tipos de dados. A idéia de uma modelagem compatível com linguagens já existentes teve que ser abandonada por não ser viável num sistema eficiente. Os tipos de dados serão implementados através de um conceito de "função" parecido com o da linguagem SLEUTH que permite um tratamento mais sistemático e, também, mais poderoso.

1.2.6 - L.G.S. (Linguagem Geral de Simulação)

Responsável: Prof. H.M.Quintella

1.2.6.1- Objetivos: O projeto visa:

a) identificação das características desejáveis numa linguagem geral de simulação e a implementação de protótipos parciais de aplicação com acabamento não industrial:

b) Treinamento de um pequeno núcleo de pessoas com o know-how adequado para projeto e desenvolvimento de modelos computacionais de sistemas gerais e de compiladores de linguagem de simulação.

1.2.6.2- Aplicabilidade: A linguagem de simulação tem grande aplicação em :

a) problemas industriais e comerciais como por exemplo: planejamento de produção, controle de projeto, lay-out de lojas de grande porte, projeto , modelagem e avaliação de sistemas de informação etc...

b) problemas de pesquisa experimental como física das radiações, biofísica, geologia, física e engenharia do meio ambiente etc... Com substanciais vantagens econômicas sobre outros métodos de previsão e avaliação de alternativas.

1.2.6.3- Motivação: Muitos dos packages e compiladores de linguagens de simulação apresentam um ou vários dos seguintes problemas:

a) não são gerais.

b) não são orientados para o usuário

c) são muito caros e/ou produzidos no estrangeiro.

A realização desse projeto teria duas contribuições importantes:

1) tomaria algumas aplicações de simulação independentes de "brain - ware" especializado em informática.

2) estimularia a transferência de know-how de projeto e desenvolvimento de compiladores de linguagens de simulação.

1.2.6.4- Situação do projeto:

O projeto se encontra em fase de exploração de grandes alternativas e avaliação de viabilidade. Nessa fase participaram do mesmo diversos alunos de mestrado desse departamento tendo havido inclusive a colaboração de um professor deste departamento.

Os trabalhos produzidos até o momento são os seguintes:

1 - Fundamentos da Teoria de Cadeias para uso em linguagem de simulação - RT-02-76 Dep. Inf. Quintella H.M. jul. 76

2 - On the modelling and representation of abstractions in Simulation languages, IFIP Working Conference On Modelling of Evironmental Systems Tokyo 76 Lucena C.J., Quintella H.M., Schwabe D.

3 - Estudos Preliminares para especificação e implementação de uma linguagem geral de simulação (LGS) Quintella,H.M., Silva E.O. e Silva V.G. Agosto 76 Relatório Técnico PUC.

4 - Estudo comparativo da aplicação de linguagens de simulação em sistemas de filar , Quintella H.M., Roelsch W.C., Leite M.C. Relatório Técnico a ser publicado.

5 - Formalização de um subconjunto de uma linguagem geral de simulação através de BNF e CBL, Miranda CC julho 77 (a ser publicado)

6 - Considerações sobre a estrutura externa de dados e formato padrão de saída do protótipo de uma linguagem geral de simulação Quintella H.M., Miranda CC. jjunho 77 (a ser publicado)

O primeiro protótipo parcial já foi produzido e o resultado da análise de sua performance estão relatados em diversos dos documentos acima citados.

Atualmente a equipe compreende 3 pessoas:

- Coordenador (professor do Depto. de Informática)
- Um especialista (a ser pago pelo projeto)
- Um especialista (aluno de mestrado) trabalhando para obtenção de créditos (em geral esse papel é desempenhado por diversos alunos trabalhando 30 horas por semana por algumas semanas em épocas distintas do ano.

1.2.7 - ESPECIFICAÇÃO DO PROJETO COMCOM2

Responsável: Prof. Arndt von Staa

1.2.7.1 - Introdução:

Com o projeto COMCOM2 visamos criar um ferramental útil para a especificação de compiladores e de pré-processadores. Este ferramental é fundamental para a condução de experimentos de desenvolvimento de linguagens de acesso a bancos de dados, de linguagens de especificação de programas e de linguagens de processamento simbólio que vêm sendo conduzidos no Departamento de Informática da PUC/RJ.

A especificação atual do sistema COMCOM2 é uma evolução de um compilador de compiladores - COMCOM - escrito em 1970 (v. Staa 70). Esse sistema demonstrou sua utilidade em diversos projetos (Furtado 70, Azeredo 72). O sistema COMCOM original possuía, porém, diversos erros de concepção e tinha restrições que reduziam sua aplicabilidade geral. Pior que isto, o sistema COMCOM original havia sido escrito em Assembler 7044 e tornou-se inútil com a desativação do computador IBM 7044 outrora insatalado no Rio Datacentro da PUC/RJ.

Com base na experiência do sistema COMCOM original, e com base na evolução do estado da arte em ciência da computação, propomos a criação de um novo sistema - COMCOM2 - que seja moderno, flexível e de aplicabilidade geral. Como subproduto deste esforço almejamos, também, experimentar técnicas de desenvolvimento de programas e de técnicas de documentação.

1.2.7.2 - Especificação do Projeto COMCOM2

O sistema COMCOM2 constará de um conjunto de rotinas e sistemas geradores de rotinas que visarão dar apoio à construção de compiladores e/ou pré-processadores. São seguintes os componentes do sistema COMCOM2:

1 - Um sistema para a geração de co-rotinas que perfaçam a filtragem léxica, acrescidas de funções particulares fornecidas pelo usuário. Os filtros léxicos serão gerados a partir de especificações dadas pelo usuário;

2 - Um sistema para a geração de co-rotinas que perfaçam a transdução sintática, baseada em linguagens do tipo SLR(1), acrescidas de rotinas de análise semântica fornecidas pelo usário. Estas co-rotinas produzirão árvores sintáticas reduzidas. Os transdutores serão gerados a partir de especificações dadas pelo usuário;

3 - Um sistema para a geração de co-rotinas que perfaçam a otimização de programas através da manipulação de árvores reduzidas. A otimização será efetuada a partir de especificações dadas pelo usuário;

4 - Um conjunto de rotinas para a geração de código a partir de árvores sintáticas reduzidas, acrescido de rotinas fornecidas pelo usuário;

5 - Um conjunto de rotinas para suporte ao pós-processamento da fase de compilação;

6 - Um conjunto de rotinas para suporte aos programas em tempo de execução;

7 - Um conjunto de rotinas para suporte ao pós-processamento da fase de execução.

Partes do COMCOM2 carecerão de pesquisas e/ou de criação de protótipos para o levantamento de características de comportamento. Os resultados destas pesquisas e/ou criação de protótipos serão parte da documentação do projeto COMCOM2.

1.2.7.3 - Ambiente de Hardware e Software.

O sistema COMCOM2 será desenvolvido tendo por base o computador central instalado no RDC-PUC/RJ, um IBM/370-165. Inicialmente o sistema COMCOM2 será desenvolvido para um ambiente de programação em PL/1F. Futuramente será revisto e possivelmente adaptado a outro ambiente de programação.

Sistemas geradores poderão produzir programas (co-rotinas) em assembler (ou LKED) e poderão estar escritos em linguagens quaisquer. Porém, o ambiente de execução dos programas gerados e das bibliotecas de subrotinas

é o da linguagem PL/IF.

A escolha deste ambiente deve-se ao fato de PL/1 possuir amplas facilidades de manuseio de informação e ser a linguagem utilizada com maior frequência pelo corpo discente do Departamento de Informática. Em se tratando do COMCOM2 um projeto experimental, é de importância maior haver um rápido retorno quanto à sua utilidade do que gerar compiladores altamente eficientes.

1.2.7.4 - Documentação

O desenvolvimento do sistema COMCOM2 gerará diversos documentos. O uso desta documentação será facilitado através de três publicações de suporte:(1) Ree, (2) S = ário e (3) dice.

Além dessas publicações serão produzidos os documentos (4) Planejamento e, para cada subsistema, (5) Especificação do Subsistema, (6) Manual do Programa e (7) Manual do Usuário.

Algumas destas publicações são dinâmicas no sentido de sofrerem alterações e/ou acréscimos à medida que o projeto for evoluindo.

A seguir daremos uma vista geral do conteúdo de cada um dos documentos a serem produzidos.

1. Rede de Documentação do COMCOM2

Este documento tem por objetivo demonstrar o interrelacionamento e a precedência de cada um dos documentos que compõe o resultado do sistema COMCOM2. Serão relacionadas também outras publicações (livros, teses, monografias) que tenham fornecido subsídios diretos e/ou sejam subprodutos do desenvolvimento do sistema COMCOM2.

A rede de documentação é um documento dinâmico, sofrendo revisões sempre que novas publicações forem anexadas ao conjunto documentos do COMCOM2. Ela é organizada sob a forma de um grafo, onde os nodos são os documentos, ou con

juntos de documentos, e as arestas definem o relacionamento de precedência.

2. Sumário do COMCOM2.

O Sumário do COMCOM2 tem por objetivo permitir o discernimento rápido quanto ao conteúdo de cada documento produzido durante a elaboração do COMCOM2. Este documento visa tornar mais rápido o acesso à informação desejada por pessoas não familiarizadas com o projeto como um todo.

A organização deste documento é a "folha solta" e sua atualização será feita sempre que novos documentos forem incorporados à coletânea de documentos do COMCOM2.

Cada "folha solta", possívelmente ocupando mais de uma folha física, conterá o seguinte:

a - código de identificação
b - título
c - autor (es)
d - data da publicação
e - sequência das datas de revisão em ordem cronológica
f - tipo da publicação (tese, monografia, manual, etc.)
g - número de páginas
h - resumo da publicação
i - palavra chave
j - sumário da publicação (tabela de conteúdo)

3. Índice do COMCOM2.

O Índice de COMCOM2 tem por objetivo permitir o acesso rápido a documentos que tratem de assuntos específicos, através de pesquisa por palavras chave.

A organização do Índice do COMCOM2 é de fichário e sua atualização é feita à medida que novos documentos forem incorporados a coletânea.

A determinação das palavras chave será feita à medida que os documentos forem sendo produzidos, não estando definida à priori.

4. Planejamento do COMCOM2

O objetivo do documento "Planejamento do COMCOM2" é o de fornecer informações quanto às tarefas já realizadas e por serem realizadas, permitindo uma rápida verificação do andamento do projeto.

Serão definidas metas intermediárias(pontos de controle) que constarão da apresentação de resultados específicos.

5. Manual de Especificação:

Para cada programa ou subsistema será fornecido um Manual de Especificação. Este manual tem por objetivo definir precisamente o que deverá ser feito pelo programa ou subsistema, sem porém entrar em detalhes quanto à sua implementação (como é feito).

As interfaces entre subsistemas terão sua especificação feita em separado como se fossem subsistemas autônomos.

Cada Manual de Especificação conterá:

a - objetivo da interface, programa ou subsistema
b - teoria de apoio
c - características operacionais, tempos de respostas, volumes e nomes de arquivos.
d - dados de entrada - sintaxe e semântica
e - resultados produzidos - listagens, organização e estrutura dos resultados
f - índice

6 - Manual de Programa

Para cada programa ou subsistema será produzido um

manual de programa. O objetivo deste manual é dar informação sobre como o programa ou subsistema foi implementado.

O manual de programa constará de :

a - objetivo do programa ou subsistema
b - descrição das estruturas de dados internas utilizadas (tipos abstratos)
c - propriedades de desempenho dos algoritmos utilizados (estatísticas, gráficos, simulações)
d - listagem comentada do programa
e - mapas de teste (comprovação do plano de testes)
f - dados e resultados dos casos teste executados.

7- Manual de Operação

Para cada programa ou subsistema será produzido um manual de operação. O objetivo deste manual é de dar informações quanto à implantação e uso do programa ou subsistema.

O manual conterá:

a - objetivo do programa ou subistema
b - pontos de entrada ("entry points")
c - pontos externos referenciados ("external", "common")
d - mensagens de erro produzidas
    - explicação
    - causa
    - consequência
    - remédio
e - arquivos utilizados: formatos, organização, volumes, unidades.
f - JCL necessário para o uso
g - JCL necessário para a incorporação
h - propriedades de desempenho.

REFERÊNCIAS:

Azeredo, P.A. - Otimização de Código Objeto e suas Aplicações em um Compilador para a Linguagem BASIC Usando um Compilador de Compiladores; Tese de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro; 1972

Bauer, F. L.; Eickel, J. - Compiler Construction: An Advanced Course; Springer Verlag; 1975

Furtado,A.L. - PUCMAT - A Programming Module for Arithmetic Pattern Matching Monographs in Computer Science no. 2/71, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro; 1971

Huang, J.C. - "An Approach to Program Testing"; em Computing Surveys Association for Computing Machinery 7(3); September 1975; pp 113 - 128

Staa,A. v. - COMCOM - Compilador de Compiladores; Grupo de Aplicações, Rio Datacentro, Pontifícia Universidade Católica do Rio de Janeiro; 1970

1.2.8 - LINGUAGENS DE CONTROLE E COMANDO

Responsável: Prof. Michael A. Stanton

Com o advento de sistemas operacionais modernos com multiprogramação e acesso por terminal, tornou-se necessária a criação de linguagens em que o usuário possa descrever ao sistema os recursos de hardware e software necessários para execução do seu programa, recursos estes tais como arquivos de dados, equipamentos periféricos etc. Obviamente um programa pode especificar todos estes dados na hora de compilar, mas é bem mais útil especificá-los na hora de execução. Para atender estas necessidades no Sistema OS/360 a IBM introduziu JOB Control Language (JCL) para uso não interativo, e a Time Sharing Option Command Language (TSO-CL) para uso interativo. As linguagens são muito parecidas, fornecendo facilidades para descrever o meio ambiente para a execução de um programa, estruturas primitivas de controle e uma facilidade de criar "procedures" com passagem de parâmetros. É notória a dificuldade de usuários aprenderem aproveitar bem das facilidades do sistema operacional através destas duas linguagens. O objetivo deste projeto é fornecer uma máquina virtual mais cômoda através de uma linguagem de controle de maior sofisticação.

Como primeiro passo para alcançar este objetivo foi estudado a linguagem de comando PHOENIX, que junto com seu sistema interpretativo, foi desenvolvido por B. Landy da Universidade de Cambridge, onde o Sistema roda num 370/165 usando uma versão bastante modificada de OS/MVT. Obtivemos a fonte do Sistema PHOENIX em setembro de 1975, e começou o trabalho de adaptação para funcionar num sistema OS/MVT padrão. Este trabalho agora está quase terminado e PHOENIX foi liberado para uso em maio deste ano.

O que é PHOENIX ? há duas respostas: a primeira é a linguagem de comandos; e segunda , são os programas que incluem o interpretador dos comandos, um editor de texto e vários programas para manuseio de arquivos (utility programs). Estes programas substituem um outro conjunto de programas fornecidos por IBM, e as funções desempenhadas são quase as mesmas, embora de maneira mais conveniente. Notamos aqui o editor de texto e os programas para manipulação de arquivos particioados(XPDS), para fornecer informação sobre os arquivos de um usuário(EXAMINE) e para copiar um arquivo sequencial em qualquer for

mato (FILE). Estes programas foram desenvolvidos em Cambridge para suprir deficiências do Sistema padrão do fabricante.

Mais importante para este projeto é a linguagem PHOENIX que é ao mesmo tempo mais expressiva e mais poderosa de que a TSO-CL. Para citar um executar um programa que acessa quatro arquivos usando TSO-CL é necessário a execução de seis comandos básicos, sendo que 5 destes tratam de alocação dos arquivos,i.é, o estabelecimento de canais de entrada/saída entre o programa e seus arquivos. Em PHOENIX o comando básico consiste da especificação do programa a ser executado, junto com os arquivos usados. Por exemplo, para listar um arquivo usando IEBGENER:

```
TSO-CL.     ALLOC   F(SYBRINT)  DA(*)
            ALLOC   F(SYSIN)  DA('NULLFILE')
            ALLOC   F(SYSUT1)  DA(nome do arquivo)  SHR
            ALLOC   F(SYSUT2)  DA(*)
            CALL    'SYS1.LINKLIB(IBGENER)'
            FREE    F(SYSPRINT SYSIN SYSUT1 SYSUT2)

PHOENIX:    :IEBGENER (SYSPRINT = * SYSIN = $ SYSUT1 = nome-do- arquivo
             SYSUT2=*)
```

A diferença mais importante entre TSO-CL e PHOENIX reside no mecanismo de 'procedures'. Uma 'procedure' é essencialmente uma lista de comandos contida num arquivo. Esta lista pode ser executada por um comando, o chamado "procedure call",com propriedades parecidas às de uma procedure-subrotina em linguagens de programação. Podemos notar uma sofisticação gradativa nas estruturas de controle disponível na passagem de TSO-CL pela PHOENIX para uma linguagem de programação. Em todas estas linguagens temos passagem de parâmetros para procedures, sendo que nas primeiras duas linguagens isto é feito sempre por substituição de parâmetros simbólicos por cadeias de carateres que são posteriormente interpretados como comandos. Chamamos isto macro. Substituição porque é típico da implementação de processadores de macro instruções. Por outro lado, em linguagens de programação, procedures têm variáveis de determinado tipo (inteiro, caráter, etc.) cujos valores são determinados na hora de chamar. O fato dos valores serem de tipo determinado permi-

te a análise sintática da procedure, como também sua compilação antes da sua execução. No caso de macro substituição isto não seria possível pelo fato dos valores dos parâmetros não terem tipo determinada.

Uma segunda graduação de propriedades de procedures se deve ao "SCOPE" de parâmetros e/ou variáveis. Na TSO-CL os valores de todos parâmetros usados, somente podem ser especificados na chamada da procedure. No PHOENIX existem também parâmetros globais cujos valores podem ser alterados através de um comando especial. Estes valores serão usados numa macro substituição se um valor local do mesmo parâmetro não for especificado na chamada da procedure. A extensão natural desta idéia é encontrada nas regras de "SCOPE" de linguagens de programação como ALGOL.

Finalmente quando comparamos as estruturas de controle observamos que TSO-CL (em OS/MVT) somente temos a execução de um comando sendo condicional no resultado do anterior, expresso através de um código numérico. Em PHOENIX diversas condições podem ser testadas: além da última mencionada, também podem ser testadas condições de erro e os valores de parâmetros. Os comandos de desvio são simples, permitindo a transferência de controle a qualquer outro comando da procedure. Notamos que estes desvios não são "estruturados" no sentido moderno do termo, mas oferecem ampla flexibilidade. Contrastamos isto com uma linguagem estruturada de programação onde temos meios de testar qualquer condição imaginável, mas os tipos de desvio permitido são bem disciplinados.

(OBSERVAÇÃO: notamos que versões de TSO-CL para funcionar com OS/VS tem muito mais estruturas de controle para procedures, especialmente parâmetros globais, e desvios e testes de condições menos ampla do que com MVT. Mas parece que ainda oferece menos facilidades do que o PHOENIX.)

O motivo desta comparação era mostrar a tendência de linguagens de controle a se assemelhar a linguagens de programação. (Isto pode ser visto particularmente na WORK FILE LANGUAGE (WFL) do Sistema Burroughs MCP-B6700, que deve muito a ALGOL.) Nosso objetivo é chegar a uma definição de uma linguagem de controle que tenha as propriedades acima descritas de ter variáveis tipadas, e uma melhor estruturação de estruturas de controle. Assim seria possível uma compilação de procedures muito usados, e podemos imaginar um programa de controle sendo interpretado parcialmente da linguagem fonte e par-

cialmente de forma compilada. Notamos porém que esta linguagem deferiria de linguagens como ALGOL devido à possibilidade de criar novas variáveis, a qq. momento, devido a interação do usuário.

Esperamos poder produzir uma implementação da linguagem proposta para rodar no sistema OS/TSO até o final de 1978.

Neste trabalho conto com a colaboração do Sr. Henrique Manela na parte de implementação e documentação. Agradeço a ajuda substancial do Computing Serviçe da Universidade de Cambridge e do seu Diretor, Dr. D.F. Hartley pela doação de software, e particularmente pelas facilidades que gozei durante duas visitas em 1975 e 1976. O progresso deste trabalho deve muito as frequentes e longas conversas que tive com meu colega e amigo Dr. Michael Challis.

REFERÊNCIAS:

1 . IBM. JCL Reference. Form number GC28-6704

2 . IBM. TSO Command Language Reference. Form numbers GC28-6732

3 . University of Cambridge Computing Serviçe. Cambridge 370/165 Users' Reference Manual

2. SOLUÇÃO NUMÉRICA DE EQUAÇÕES DIFERÊNCIAIS

2.1 Objetivo geral

O presente projeto objetiva a implementação de novos métodos numéricos para a solução de equações diferenciais ordinárias e parciais, presentes na maioria dos problemas atuais, principalmente na Engenharia e na Física.

No caso de equações ordinárias serão desenvolvidos métodos especialmente orientados ao tratamento de "stiff equations", cuja solução numérica pelos métodos tradicionais é, em geral, difícil. Esses métodos são baseados no uso cíclico de métodos "multistep". Quanto às equações parciais com condição de contorno, serão estutados, principalmente, métodos que utilizam elementos finitos como discretização.

Também serão focalizados métodos especiais para a solução de sistemas lineares com matrizes esparsas, sistemas esses advindos da discretização das equações parciais.

É responsável por este projeto o professor Michael Anthony Stanton.

2.2 Relatórios

Conforme o objetivo geral do projeto descrito em 2.1, definiram-se dois sub-projetos que se desenvolveram paralelamente. Os relatórios desses sub-projetos são apresentados em seguida.

2.2.1 - Estudo da região de estabilidade de métodos ciclícos.

Responsáveis : Profs. Peter Albrecht e Therezinha C.F. Chaves

2.2.1.1 - Descrição

Este projeto visa ao desenvolvimento de métodos cíclicos especiais , cuja região de estabilidade seja a mais ampla possível.

Os métodos cíclicos têm sido estudados intensivamente nos últimos anos, principalmente depois da publicação, em 1971, do artigo [1] de Donelson e Hansen. Esses métodos têm se mostrado muito eficientes na solução de equações diferenciais ordinárias e são métodos preditor-corretores onde, ao invés de apenas um corretor, são usados vários, ciclicamente. Dessa maneira , verificou-se que a combinação cíclica de fórmulas de passo múltiplo permite um aumento considerável na ordem de consistência do método final, obtendo- se fórmulas estáveis mesmo quando se usam corretores instáveis, superando as dificuldades apontadas por Dahlquist em seu artigo clássico [2].

Vários são os autores que estudam atualmente essas fórmulas, abordando o tratamento de várias maneiras.

O estudo aqui utilizado é baseado no trabalho do Prof. Peter Albre - chet, cujos resultados são encontrados em [3].

Sua abordagem permite a criação e a manipulação de fórmulas de passo múltiplo ("multistep") de uma forma simples e direta, apresentando condi - ções para o aumento da ordem do método combinado e para a garantia da estabilidade final.

Aliando-se esse conceito ao de regiões de estabilidade, procura- se determinar métodos que além de possuirem uma ordem elevada, têm todas as características de métodos especiais para o tratamento de "stiff-equations" .

Um sistema desse tipo é de difícil tratamento com os métodos tradicionalmente usados para solução de equações diferenciais ordinárias,porque torna-se difícil ajustar o passo h de modo a que o sistema tenha globalmente uma solução com a mesma precisão.

Nesses casos os métodos A- estáveis são especialmente indicados , conforme [4].

Esse desenvolvimento faz parte do trabalho de tese de doutorado da prof. Therezinha Chaves, sob a orientação do prof. Peter Albrecht.

Como resultado, espera-se definir e implementar rotinas de solução de equações diferenciais ordinárias do tipo "stiff".

2.2.1.2 Situação Atual

Esse subprojeto foi iniciado em abril do corrente ano. Como primeira etapa fez-se um estudo para a criação de métodos cíclicos de passo 3. Foi proposta como tarefa para esse período a elaboração de rotinas que gerassem automaticamente triplas de corretores com passo 3, determinando a estabilidade do método composto. Esse trabalho não é simples pois é necessário que , para cada uma das centenas de triplas de fórmulas escolhidas se determine as raízes de um polinômio com coeficientes e raízes complexas, e se resolva um sistema não-linear razoàvelmente complicado.

2.2.1.3 Fases

As fases distintas do projeto consumiram, aproximadamente:

- estudo teórico, com a comparação da literatura sobre o assunto: 2 meses;
- implementação e testes, com a elaboração do programa e de exemplos testes: 3 meses;
- documentação provisória, em fase de conclusão: 1 mês.

Nos próximos meses serão generalizados esses resultados, obtendo-se métodos cíclicos com passo 4 e 5, e combinados de modo a se ter um compromisso entre ordem elevada e ampla região de estabilidade.

REFERÊNCIAS:

[1] - J. Donelson and E. Jansen

"Cyclic Composite Multistep Preditor - Corretor Methods" SIAM J NUMER. ANAL., Vol. 8, Nº 1, March 1971

[2] - G. Dahlquist "Convergence and Stability in the Numerical Integration of Ordinary Differential Equations" MATH. SCAND., 4 (1956) pp. 33-53

[3] - P. Albrecht "Explicit, optimed Stability functionals and their application to cyclic discretization methods". a aparecer em "COMPUTING"

[4] - L.Lapidus and J. Seinfeld "Numerical Solution of Ordinary Differential Equations" Academic Press - 1971

2.2.2 - ANÁLISE TEÓRICA E DESENVOLVIMENTO DE TÉCNICAS DE DETERMINAÇÃO EXPERIMENTAL DO ERRO E DA CONVERGÊNCIA MÉTODO DOS ELEMENTOS FINITOS

Responsável: Prof. Vitoriano Ruas.

2.2.2.1- Introdução

Esse projeto visa ao desenvolvimento e aplicação de técnicas de resolução numérica de equações parciais pelo método dos elementos finitos. Paralelamente ao estudo de problemas correlatos como a análise de erro e convergência e à implementação de rotinas para processamento com elementos finitos, servindo também para a análise de convergência experimental, um objetivo de carater teórico foi também fixado para este projeto afim de satisfazer as necessidades do corpo discente e auxiliar de pesquisa. Tratando-se de assunto recentemente desenvolvido com base em aspectos matemáticos específicos e em geral pouco conhecido por engenheiros, que são justamente os que possuem interesse particular nessas técnicas, foi dada ênfase ao aspecto de formação de futuras pesquisadores na área através da elaboração de textos matemáticos especialmente orientados para o assunto.

A descrição mais precisa dos resultados já obtidos e a obter nessas duas linhas de trabalho é apresentada em seguida.

2.2.2.2 Linhas de desenvolvimento.

A primeira lina de caráter eminentemente teórico introduz uma extensão do plano inicialmente apresentado. Sua inclusão, no entanto, se justifica pela necessidade de dar ao corpo discente e auxiliares de pesquisa formação mais sólida em fundamentos matemáticos especificamente orientados para o estudo de aspectos por vezes bastantes sofisticados da convergência e erro do método dos elementos finitos.

Assim sendo, com base nas observações colhidas em um ano de experiências com cursos sobre o assunto foram elaborados dois textos destinados ao usuário cuja formação matemática não seja suficientemente abrangente estando, entretanto, especificamente interessado nesse aspecto da resolução numérica

de equações diferenciais parciais, caso em que enquadramos notadamente os engenheiros.

O 1º texto em fase de conclusão, prevista para fins de setembro próximo, versa sobre Análise Funcional básica (ver referência 1). Além de se introduzirem todos os conceitos fundamentais desse ramo da matemática, tais como os de espaço vetorial, normas, convergência, operadores e funcionais procura-se orientar o texto para as aplicações com a farta apresentação de exemplos. Este texto escrito em colaboração com um auxiliar de pesquisa deverá sair em breve na forma de monografia editado pelo setor de publicações do Departamento, com cerca de 120 páginas.

O 2º texto (ver referência 2), já pronto, versa sobre espaços de funções interessando diretamente ao estudo de equações diferenciais com condições de contorno. Estas são enfim tratadas na forma sob a qual se apresentam para resolução pelo método dos elementos finitos: a forma variacional. Procura-se introduzir da maneira mais direta e simples possível esses assuntos básicos para o desenvolvimento apropriado de resultados de convergência e de estimativas de erro tal como têm sido tratados na literatura. Por meio de exemplos limitados ao caso de equações ordinárias conduz-se o leitor a generalizações naturais a casos análogos de equações diferenciais parciais, intrinsecamente mais complexos, e mais interessantes nas aplicações.

O texto que deverá ser utilizado como referência tanto na PUC / RJ como no CBPF para os cursos programados para a Escola de Matemática Aplicada a funcionar no verão de 1978, consta de 130 páginas. Será editado em breve pela série Cadernos da PUC. O autor pretende ainda submetê-lo para publicação na série do Projeto Euclides, coordenado pelo IMPA.

A segunda Linha é essencialmente prática e diz respeito a tópicos previstos para estudar no projeto inicialmente apresentado.

Foram obtidos resultados particularmente importantes no tocante à geração automática de triângulações para a resolução de problemas de contorno bidimensionais definidas sobre domínios quaisquer, pelo método dos elementos finitos.

O processo instituído pelo prof. Vitoriano Ruas, em 1974, para tais

triangulações e que já havia sido aplicado com sucesso para testar resultados teóricas de convergência que obtivera para a resoluções numérica de proble - mas de placas (ver referências [4] e [5]) foi agora analisado. Ficou provado que sua aplicabilidade é garantida bastando que o domínio bi-dimensional satisfaça a uma condição que ocorre frequentemente na prática: que o domínio seja estrelado (isto é, que todo segmento de reta unindo um certo ponto fixo a qualquer outro ponto do domínio fique inteiramente contido no domínio)

Em seguida foram elaboradas rotinas para a geração automática e numeração dos triângulos, as quais se encontram parcialmente à disposição do usuário.

Foi redigido uma monografia contendo a descrição do processo de geração automática e respectiva análise de aplicabilidade (ver referência 3 abaixo) Constam ainda da mesma expressões para avaliação numérica de constantes que figuram em resultados clássicos de convergência do método dos elementos finitos. Os trabalhos referentes à 1ª. linha tiveram início em março último, sendo o término dos textos previsto para fins de setembro próximo. Nos 4 meses subsequentes senão levadas a efeito a revisão, correção e tiragem dos textos.

Os trabalhos referentes à 2ª. linha tiveram início em maio último, deles só restando a implementação de rotinas para a numeração de graus de liberdade, o que deverá ser feito até o fim de setembro próximos.

REFERÊNCIAS

[1] - Ruas,V. e Nakanishi T. - "Elementos de Análise Funcional Aplicada" (a sair na série de monografias do Depto. de Informática)

[2] - Ruas,V. - "Introdução aos Problemas Variacionais" (a sair na série cadernos da PUC (coordenada pelo DIE/PUC/RJ)

[3] - Automatic Generation of Triangular Meshes for Solving Partial Differential Equations by the Finite Element Method (relatório técnico do DI / PUC submetido para publicação na revista Computer Mathematics - USA).

[4] - Some Results en the Curved Plate Bending Problem Solved with Non- Conforming Finite Element Methods (Calcolo, Itália, a sair)

[5] - Sur L'Application de Quelques Elements Finis Non Conformes à la Resolution du Probleme de La Flexion et des Vibrations des Plaques Minces - These de Doctorat 3ª. Cycle, Université Paris VI (1976)