A LINGUAGEM PASCAL SCRIPT E SUA APLICAÇÃO AO PROPAGAR MOO.

João Soares Viegas Filho[1], Antonio Eduardo Leão Lanna[2] e Adriano Rochedo Conceição[3].

Resumo - O presente trabalho tem por objetivo apresentar a linguagem PASCAL SCRIPT como uma ferramenta auxiliar de grande valia para os modelos computacionais desenvolvidos segundo técnicas de Modelagem Orientada a Objetos e integrantes de Sistemas de Apoio à Decisão aplicados a Sistemas de Recursos Hídricos. O artigo demonstra a concepção básica da linguagem, suas principais características e uma aplicação da mesma ao PROPAGAR - MOO - programa integrante do SAGBAH (Sistema de Apoio ao Gerenciamento de Bacias Hidrográficas), destinado à simulação de propagação de vazões em bacias hidrográficas visando o atendimento de demandas hídricas submetidas a decisões gerenciais e a regras de operação de reservatórios.

Abstract - This paper presents the PASCAL SCRIPTS language with as a valuable tool to Water Management Decision Support System developed with Object Oriented (OO) approach. The basic concepts and language characteristics are presented and a application of a river basin simulation model named PROPAGAR MOO, part of the SAGBAH - River Basin Management Support System, is presented as an example.

Palavras-Chave: PASCAL SCRIPT, Sistemas de Apoio à Decisão, Modelagem Orientada a Objetos, PROPAGAR, SAGBAH.

[1] Engenheiro Civil, Doutor, Professor Adjunto da Faculdade de Engenharia Agrícola - UFPel. Av. República, 440 - Areal - Pelotas - RS : CEP 96077-230 : fone (0532) 28-3060. e-mail: jviegas@conesul.com.br

[2] Engenheiro Civil, Ph.D, Professor Titular do Instituto de Pesquisas Hidráulicas - UFRGS.
Caixa Postal 15029. CEP 91501-970. Porto Alegre, RS.

[3] Bel. em Ciência da Computação, Analista de Sistemas, Instituto de Física e Matemática - UFPel.
Rua Fernando Ferrari, 248 apt 304A : Areal : Pelotas : RS : CEP 96080-090.

INTRODUÇÃO.

A utilização da Modelagem Orientada a Objetos (MOO), bem como de linguagens como o Object Pascal, contido no ambiente de desenvolvimento Borland Delphi, vem, certamente, trazer uma nova dinâmica à modelagem no âmbito dos Sistemas de Recursos Hídricos.

Alguns dos aspectos levantados por diversos autores, tais como Loucks et ll (1985), Goulter (1992), Simonovic (1992) e Porto e Azevedo (1997), relativamente a diferentes tipos de dificuldade que conduzem a existência de um "gap" entre o que se produz nos ambiente de pesquisa e o que se aplica na prática profissional, tais como a aridez das interfaces e a dificuldade de utilização dos modelos por parte dos usuários, ficam em grande parte superados. Além disso, todo um conjunto de ferramentas tais como editores de texto, planilhas eletrônicas e gráficos vêm possibilitar uma poderosa instrumentalização dos diálogos de análise, da apresentação de resultados e, ainda, da construção de relatórios.

Entretanto, como acontece com todo o avanço, a simplificação por um lado, termina por conduzir ao aumento de dificuldades por outro. Embora, como foi visto, a MOO induza a uma prática de modelagem muito mais intuitiva e alicerçada sobre objetos do mundo real, além de uma maior facilidade de abordagem da complexidade e uma maior produtividade, face à ferramentas tais como o Delphi e C++, dentre outras, o que indiscutivelmente vem facilitar a vida de pesquisadores e modeladores, o domínio de tais ferramentas envolve estudo e tempo. Isso exige, para facilitar o modelador, a criação de bibliotecas, de um lado - como é o caso da Biblioteca de Classes PROPAGAR MOO (Viegas Fº, 2000) -, e o estudo de outras, de terceiros - como aquelas que possibilitam o trabalho com editores de texto, planilhas e gráficos.

Dessa forma, isso tudo termina por aumentar a dificuldade de trabalho do usuário final médio que, embora não seja um "expert" em programação, conseguia a partir de um modelo procedural tradicional em FORTRAN, BASIC ou PASCAL, já desenvolvido e com fonte aberto, inserir nele pequenas rotinas destinadas a dar-lhe um caráter particular no trato de um problema específico. Exemplo disso é o PROPAGAR versão DOS, construído em FORTRAN (Lanna, 1997); nele o usuário pode programar, de modo simples, as rotinas PLANEJA e OPERA para atender as necessidades de um dado projeto. Isso feito compila o modelo e faz a sua aplicação.

A disponibilização dessa mesma "facilidade" no âmbito de um modelo concebido segundo o paradigma da MOO, como é o caso do PROPAGAR MOO, e através de um ambiente de desenvolvimento como o C++ e o Delphi, por mais simples que o mesmo seja, faz com que o usuário tenha que lidar com uma gama bastante grande de elementos característicos dessa forma de modelagem, tais como classes, atributos, métodos, rotinas, variáveis, configurações de compilação e de "linkedição" de uma complexidade que dificilmente ele estará disposto a enfrentar. Por outro

lado, manter o aplicativo “fechado”, criando rotinas padrões e alternativas pré-estabelecidas tira uma das grandes vantagens originais desses modelos.

A linguagem PASCAL SCRIPT, comportando-se como um “Pascal interpretado”, de fácil programação, dentro dos moldes “procedurais” tradicionais, permitindo, ao mesmo tempo, isso ser feito em “tempo de execução” e, ainda, possibilitando a manipulação dos atributos e métodos de objetos do modelo concebido e implementado à luz da MOO, vem constituir uma ferramenta de grande valia na construção desses aplicativos. O presente trabalho tem por objetivo apresentar a linguagem Pascal Script, sua concepção básica, as razões e diretrizes que orientaram a sua criação, suas principais características e, finalmente, uma aplicação da mesma.

A LINGUAGEM PASCAL SCRIPT - CONCEPÇÃO BÁSICA, SINTAXE E FUNCIONALIDADE.

A Concepção Básica.

A linguagem PASCAL SCRIPT, conforme já mencionado, é uma linguagem de “scripts”, com uma sintaxe semelhante à do PASCAL - portanto, estruturada - e baseada em objetos.

A idéia de utilizar a sintaxe do PASCAL decorreu do fato da mesma ter sido originalmente criada para ser um “interpretador” PASCAL e, ainda, em virtude de ser inicialmente utilizada em conexão com a Biblioteca de Classes PROPAGAR MOO, implementada em Delphi Object Pascal, como se verá a seguir. Entretanto, é fundamental que se diga que é uma linguagem totalmente independente, na sua concepção básica e estruturação funcional, tanto de ambiente, como de plataforma.

O fato de dizer-se ser a PASCAL SCRIPT baseada em objetos, decorre de que a mesma pode, através de funções especiais, conforme será visto, permitir a criação e a destruição de objetos em tempo de execução, bem como, acessar seus atributos e métodos. Entretanto, é importante que se perceba que não é, verdadeiramente, orientada a objetos, na medida em que não possui mecanismos típicos das linguagens que dão esse tipo de suporte, como é o caso da herança e o do polimorfismo.

Na presente versão, a linguagem PASCAL SCRIPT está encapsulada em um conjunto de Classe implementadas em Delphi Object Pascal, conforme será visto a seguir.

TPascalScript é a principal dentre um conjunto de classes que encapsulam as características de uma linguagem de “script” baseada no Pascal. Nela está incluído um “scanner” que é um leitor e fracionador do código de “alto nível” escrito pelo usuário; um compilador - que interpreta esse código e o transforma em um p-code (pseudo-código), ou seja, em um código intermediário entre o

de alto nível e o código de máquina; um gerenciador de p-code que é quem, de fato, transforma este em código de máquina e, também, uma máquina virtual para execução do código.

Com seu uso é possível fazer-se, dentre outras possibilidades, o seguinte (Conceição, 2000 apud Viegas Fº, 2000): criar "scripts" para controlar processos; criar "scripts" de inicialização de dispositivos; criar "scripts" de instalação de aplicativos; criar uma Linguagem de Macros para uso específico; permitir que usuários personalizem processos e algoritmos; permitir que usuários possam escrever "plugins".

Dentro desse espírito, o mecanismo encapsulado em TPascalScript apresenta as seguintes características: é totalmente orientado a objetos; apresenta bloco de definição de variáveis Var; aceita tipos Boolean, Integer, Real, String e Object; permite a construção de Procedimentos (Procedures) e Funções (Functions); permite o uso de objetos (indiretamente - através de funções especiais); permite o acesso a funções e procedimentos externos (através de funções internas especiais); realiza o controle da correção da compilação através de múltiplas mensagens de erro; permite o salvamento e posterior execução do p-code (executável); possui um código claro e altamente expansível.

Para materializar essas características TPascalScript utiliza seu mecanismo baseado nas seguintes classes: TPascalScript: que inclui o "Scanner" e o Compilador; TFuntionList: que encapsula o mecanismo de geração das bibliotecas básicas e do usuário; TObjectList: que encapsula o mecanismo de gerenciamento das bibliotecas de classes que podem ser criadas pelo "script".

Na verdade, a utilização e manipulação das classes acima é tarefa do modelador que as incorpora ao seu aplicativo construído segundo a MOO para, posteriormente, este disponibilizar ao usuário final a possibilidade de fazer sua programação de rotinas. O usuário final não necessita conhecer as classes acima mencionadas mas somente as características sintáticas e léxicas da linguagem Pascal Script. É o que será examinado a seguir.

A Estrutura da Linguagem e sua sintaxe.

Todo programa em PascalScript subdivide-se em 3 áreas (Quadro 1): (a) Cabeçalho ou Identificação do Programa [opcional]; (b) Bloco de Declarações de Variáveis [opcional] (área de declaração); e, (c) Bloco Principal (corpo do programa).

Na definição padrão da linguagem Pascal, o cabeçalho do programa, onde ocorre a sua identificação - determinado pela palavra-chave *Program* seguida no *nome* do programa -, é obrigatório, no entanto, em Pascal Script ele é opcional. Entretanto, recomenda-se a sua colocação com o propósito de documentação.

A área de declarações, onde localizam-se os blocos de declarações de variáveis, é o lugar onde são definidas aquelas que não tenham sido previamente pré-declaradas na própria linguagem. Caso, no “script”, sejam utilizadas apenas variáveis pré-declaradas, esta área pode ser suprimida. Entretanto, quando o usuário desejar definir novas variáveis dentro do escopo do “script” deverá, obrigatoriamente, fazer a sua declaração utilizando-se desta área. Isso é feito utilizando-se a cláusula **var** para que a alocação de espaço de memória para as variáveis seja feita durante a compilação. A seção **var** no Quadro 1, a seguir,exemplifica isso.

O PascalScript permite a declaração de 5 tipos de variáveis, a saber: Integer, Real, Boolean, String e Object. As variáveis declaradas como Object é que permitem a utilização do mecanismo que dá suporte, através de funções específicas, de acesso aos objetos, através de seus atributos e métodos.

O Pascal Script utiliza o conjunto de caracteres ASCII, incluindo as letras de "A" a "Z" e "a" a "z", os dígitos de "0" a "9" e outros caracteres dentro do mesmo padrão. São ignoradas as diferença entre letras maiúsculas e minúsculas. Os identificadores, que tem por objetivo nomear variáveis, funções e procedimentos obedecem, como já foi mencionado, às mesmas regras do Pascal onde: (1) o primeiro caractere do identificador deverá ser, obrigatoriamente, uma letra ou um símbolo de sublinhado (underscore; "_"); (2) os demais caracteres podem ser letras, dígitos ou sublinhados; (3) não pode ser palavra reservada. Este último caso, inclui os vários identificadores pré-definidos e que não fazem parte da definição padrão da linguagem Pascal Script, mas que consistem em procedimentos (*procedures*) e funções (*functions*), que podem ser utilizados normalmente na construção de programas, como é o caso, por exemplo, de: Beep (emite um som), ShowMessage() (mostra uma mensagem na tela), RenameFile(,) (renomeia um arquivo existente), Tan( ) (retorna a tangente de um arco), VolumeMaximo (obtém o Volume Máximo de um Reservatório), dentre outros possíveis. Não podem, portanto, servirem para nominar variáveis. A última palavra reservada da lista acima, VolumeMaximo, só o é no âmbito de uma aplicação específica desenvolvida a partir do Modelo de Objetos Básico aplicado a Sistemas de Recursos Hídricos apresentado por Viegas Fº (2000).

Os comandos definem as ações dentro do algoritmo em um programa. Comandos simples como, por exemplo, um comando de atribuição ou chamadas de procedimentos e funções podem ser combinados para formarem laços, comandos condicionais e outros comandos estruturados. Múltiplos comandos devem ser separados por um ponto e vírgula. Os comandos básicos apresentam igualmente notação baseada no Pascal como, por exemplo:

- Comandos de atribuição (p.ex.: A:=5);
- Blocos Begin-End (ver Quadro 1);

- Comandos de controle de fluxo e de tomada de decisão: laço While e For; laço For; If-Then-Else, instrução Case (em fase de implementação);
- Operadores +, -, *, / div, mod, and, or, not,>, <, =, <=, >=, dentre outros.

A manipulação de objetos, de funções e de procedimentos.

A linguagem Pascal Script pode acessar todo um conjunto de funções pré-definidas e, também, manipular objetos e acessar seus atributos e métodos.

As funções pré-definidas são diversas e com uma amplitude variada de propósitos, tais como:

- Rotinas de manipulação de arquivos: DeleteFile, RenameFile, ChangeFileExt, CreateDir, etc,;
- Rotinas matemáticas e trigonométricas: SinCos, Power, SQR, Tan, DegToRad, RadToDeg, Log10, Log2, Min, Max, etc.;
- Rotinas para gerenciamento de Strings: Delete, Insert, Pos, Copy, LowerCase, UpperCase, CompareText, TrimLeft, StrToFloat, etc.;

A Pascal Script, conforme já foi dito, é uma linguagem baseada em objetos, e como tal, permite a criação de instâncias de classes, ou seja, permite que "em tempo de execução" o usuário programe, através de funções específicas, a criação e manipulação de objetos. É o caso, por exemplo, da criação de objetos de saída como Editores de Texto, Planilhas e Gráficos. Todos esses dispositivos de saída de dados são tratados como objetos que tem de ter variáveis alocadas para tal, serem criados em memória, manipulados através de seus atributos e métodos - através de funções específicas desenvolvidas para isso - e, depois, quando não mais forem necessários, destruídos (liberados da memória). A criação de objetos se dá através da função *CreateObject('Nome da Classe')* cujo uso é exemplificado através da rotina constante do Quadro 1.

No caso do exemplo, a variável *P* é, primeiramente, declarada como *Object* na seção **var**. Depois, já dentro do corpo do programa (bloco principal), é instanciado um objeto da *Classe TPlanilha* e atribuído à variável *P* - através do comando "P := CreateObject('TPlanilha');". Posteriormente, são utilizados métodos da *Classe TPlanilha*, disponibilizados através de funções especiais em PASCAL SCRIPT para manipular a planilha. O comando "P.SetCellFont(1, 1, 'ARIAL', 14, 0, True, True, False);", faz com que à célula 1, 1, seja preenchida com caracteres da fonte ARIAL, tamanho 14, cor preta, negrito, itálico e não sublinhados; o comando "P.Write(1, 1, 'Planilha Teste');", através do método *Write*, insere na célula 1,1 o conteúdo desejado, no caso "Planilha Teste"; o comando "P.WriteFloatCenter(i, 3, i*0.3);", por sua vez, dentro de um laço do tipo "for", insere nas linhas *i*=5 até *i*=10, na coluna 3, o valor de *i* multiplicado por 0.3, como um número de ponto flutuante (real) e centralizado; "P.Copy;", copia todos os valores existentes na

planilha para a “área de transferência (clipboard)” do Windows, podendo de lá serem colados em planilhas Excel, por exemplo, ou em editores de texto; os comandos seguintes: “P.SaveToFile('D:\temp\ScriptPlanilha.xls', 'XLS');” e “P.SaveToFile('D:\temp\ScriptPlanilha.txt', 'TXT');” salvam a planilha em um arquivo disco com o nome ali indicado e segundo o formato estabelecido no segundo parâmetro (XLS ou TXT); finalmente, o comando “P.Show;”, utiliza o métoso *Show* para apresentar no monitor a planilha através de uma janela apropriada para tal.

Conforme a natureza do aplicativo que utiliza a Classe TPascalScript, funções específicas oriunta de classes existentes nas bibliotecas utilizadas, como já foi mencionado acima, são disponibilizadas através de funções que dão acesso aos seus atributos e métodos. Dentre outras, são exemplos de classes que disponibilizam funções uso com a Pascal Script: (a) dispositivos de entrada/saída: Editor de Textos (TOutPut), Planilha (TPlanilha) e Gráficos (TChartSeries); (b) criação e manipulação de matrizes e vetores (TwsMatrix e TwsVec); (c) Manipulação de objetos de uma Rede Hidrográfica (TprProjeto, TprPCP, etc.).

Quadro 1 - Exemplo de rotina Pascal Script.

```
Program Teste_Planilha;
Var
   i, ii  : Integer;
   S      : String;
   P      : Object;
begin
  P := CreateObject('TPlanilha');

  P.SetCellFont(1, 1, 'ARIAL', 14, 0, True, True, False);
  P.Write(1, 1, 'Planilha Teste');

  for i := 5 to 10 do
    P.WriteFloatCenter(i, 3, i*0.3);

  P.Copy;
  P.SaveToFile('D:\temp\ScriptPlanilha.xls', 'XLS');
  P.SaveToFile('D:\temp\ScriptPlanilha.txt', 'TXT');
  P.Show;
end.
```

A LINGUAGEM PASCAL SCRIPT NO CONTEXTO DO PROPAGAR MOO.

Conforme foi mencionado acima, embora a linguagem PASCAL SCRIPT seja uma linguagem independente, sua origem deveu-se à necessidade de ter-se uma forma de programar, dentre outras, as rotinas PLANEJA e OPERA do PROPAGAR MOO. Desse modo, aqui será utilizada essa aplicação para exemplificar as características e potencialidades dessa nova linguagem.

O PROPAGAR MOO, sua concepção e funcionalidade segundo o paradigma da MOO.

O PROPAGAR MOO é um dos aplicativos que integram o SAGBAH - Sistema de Apoio ao Gerenciamento de Bacias Hidrográficas. O SAGBAH constitui-se por um Sistema de Apoio à Decisão orientado para as atividades de Planejamento e Gestão de Recursos Hídricos, originalmente composto por um conjunto de programas desenvolvidos para a plataforma DOS, em linguagem FORTRAN (Lanna, 1997, Chaves, 1993) e atualmente sendo remodelado através da aplicação de técnicas de MOO, com a utilização da linguagem Object Pascal e do ambiente de desenvolvimento DELPHI (Viegas Fº, 2000; Viegas Fº e Lanna, 1999).

O PROPAGAR, como modelo, busca simular, como já foi dito, a propagação de vazões ao longo de uma bacia hidrográfica, visando o atendimento de demandas hídricas sujeitas a decisões gerenciais e à operação de reservatórios quando existentes. Essa é, portanto, a idéia que fundamenta o domínio do problema , ou seja, a concepção básica de quais característica o PROPAGAR abstrai da Rede Hidrográfica que tem por propósito simular. As principais características que norteiam a abstração do domínio do problema são as seguintes: (1) o modelo está alicerçado sobre uma estrutura de Rede Hidrográfica constituída por Pontos Característicos - na forma de Pontos de Passagem ou Pontos de Armazenamento - ligados entre si por Trechos de Água (Figura 1); (2) cada PC tem ligada a si uma ou mais Sub-bacias que são responsáveis pela transformação chuva-vazão que ocorre localmente; cada Sub-bacia pode estar ligada, por sua vez, a mais de um PC, desde que adjacentes; (3) existirão Demandas Hídricas na Rede as quais poderão estar ligadas a PCs - no caso de serem demandas localizadas - ou a Sub-bacias - no caso de serem demandas difusas; essas Demandas Hídricas agrupadas em Classes de Demandas (aqui, no sentido de categoria e não no sentido utilizado na MOO); (4) o atendimento às Demandas Hídricas, é feito em função da verificação de disponibilidade de água, através de balanço hídrico realizado em cada PC, por ocasião da propagação das vazões em cada intervalo de tempo de simulação, e, também, das decisões gerenciais que são estabelecidas para a Rede como um todo e para cada PC e/ou Demanda em particular.

Na medida em que o presente trabalho tem por propósito a apresentação da linguagem Pascal Script, a descrição do Modelo de Objetos que dá sustentação do PROPAGAR MOO, bem como, a descrição das classes que o compõe e sua dinâmica detalhada levariam a uma grande dispersão de idéias e conceitos. Assim sendo, optou-se por apresentar-se aqui apenas os aspectos importantes para a compreensão do seu funcionamento e da utilização da linguagem Pascal Script no seu contexto. Um estudo detalhado da metodologia de Modelagem Orientada a Objetos aplicada a Sistemas de Recursos Hídricos baseados em Redes Hidrográficas, com enfoque específico no PROPAGAR MOO, pode ser encontrado em Viegas Fº (2000), bem como em Viegas Fº e Lanna (2001a) e Viegas Fº e Lanna (2001b). A Figura 1, a seguir, ilustra a janela principal do PROPAGAR MOO, sobre cuja Área de Projeto está lançada uma Rede Hidrográfica composta por PCs, reservatórios, trechos de água, sub-bacias e demandas hídricas (difusas e localizadas). Para facilitar a visualização, parte da rede está destacada em um detalhe.

A dinâmica da simulação, como já dito, dá-se pela propagação de vazões de montante para jusante, através da bacia, ao longo do tempo, buscando atender as demandas existentes em face das disponibilidades hídricas igualmente existentes. Para tanto, em cada PC, é realizado, a cada intervalo de tempo, um balanço hídrico, considerando o afluxo de água oriundo dos PCs de montante e da suas próprias sub-bacias, bem como as demandas difusas (nas sub-bacias) e localizadas (nos próprios PCs) que ali devem ser supridas.

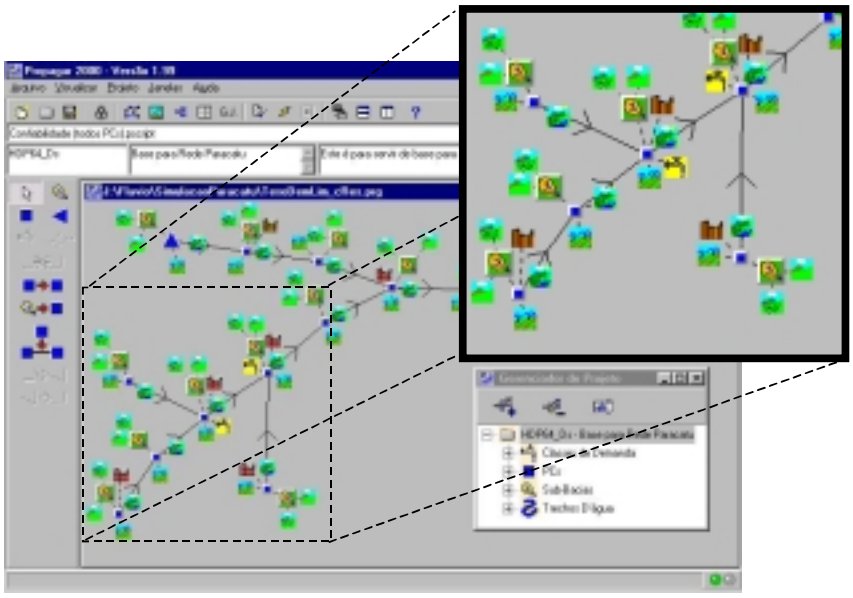

Figura 1 - Janela principal do PROPAGAR com um Rede Hidrográfica sobre a Área de Projeto.

Quando o PC corresponder a um Ponto de Passagem, ou seja, a um nó no qual não existir reservatório, a equação representativa do balanço hídrico é dada por:

$$X(t) = Q(t) - D(t) + R[D(t)] \qquad \text{(Equação 1)}$$

onde X(t) é a descarga defluente do PC; Q(t) é a contribuição afluente total, formada pelas contribuições difusas das bacias laterais mais a contribuição concentrada dos PC's de montante quando esses existirem; D(t) é representação da totalidade das demandas concentradas que devem ser supridas nesse PC e R[D(t)] é o retorno de água que possa lhe corresponder (Lanna, 1997, Viegas Fº, 2000).

Por outro lado, em PCs controlados por reservatórios a equação de propagação será a equação clássica de balanço hídrico em reservatórios:

$$S(t+1) = S(t)+Q(t)-D(t)+R[D(t)]-X(t)-E[S(t+1),S(t)] \qquad \text{(Equação 2)}$$

onde S(t) será o armazenamento no reservatório no início do intervalo (t), E[t] a evaporação que lhe corresponde e S(t+1), o armazenamento ao final do mesmo.

Nessa situação X(t) será uma descarga defluente, destinada ao atendimento dos diferentes tipos de demandas à jusante controlável pela política operacional do reservatório, que será estabelecida pelo usuário, através de uma decisão gerencial.

As decisões gerenciais são introduzidas em duas fases. Na primeira fase, denominada de fase de planejamento estratégico, são estabelecidas as políticas operacionais para todos os PCs em função da água existente na bacia naquele momento. Isso pode ser feito de duas formas: admitindo-se desconhecer a água que afluirá ao PC durante o intervalo de tempo - que é o que mais aproxima da realidade - ou, então, considerando-se esta como conhecida. No primeiro caso, a água corresponde somente à água armazenada nos reservatórios se eles existirem; no segundo, a toda a água disponível durante o intervalo de tempo. Nos cursos de água sem controle de reservatório a política a ser aplicada - e testada pelo modelo - refere-se ao nível de atendimento das demandas hídricas que são supridas no trecho. Isso permite o estabelecimento de racionamentos à montante para atender demandas prioritárias de jusante. Nos cursos de água com reservatório, a estratégia operacional envolve o estabelecimento do nível de atendimento à demanda e a descarga defluente do reservatório (Lanna, 1997).

Na fase seguinte, de operação tática ou em tempo "atual", é feita a verificação se as decisões estratégicas podem ser, de fato, implementadas. Aí são confrontadas restrições e condicionamentos de origem física, como, por exemplo, a existência de água para atendimento a uma demanda ou descarga, ou de origem gerencial, como por exemplo, se haverá de fato racionamento não obstante haver água para suprimento a uma demanda (Lanna, 1997).

Essa forma de ordenação do processo decisório simula a situação real em que a operação é planejada previamente ao conhecimento das afluências. Depois delas conhecidas avalia-se, então, a oportunidade de implementar a operação ou de retificá-la.

A idéia é a de que a estratégia operacional seja introduzida através de um método denominado *Planeja* e a tática operacional através de outro método chamado *Opera*. O primeiro, *Planeja*, está na classe correspondente ao Projeto, que controla o processo de simulação, já que deverá servir para um planejamento geral. O método *Opera*, entretanto, fica ligado à classe correspondente aos Reservatórios.

Durante a execução da simulação, na fase tática, na medida em que for feito o balanço hídrico em cada PC, será detectada a existência ou não de água suficiente para atender as demandas previstas na fase de planejamento. Quando a não existência de água suficiente for detectada, deverá ser utilizado outro método, denominado *Raciona* cujo propóstito é o de promover a distribuição adequada de água. Nos PCs sem reservatório o método promoverá o atendimento obedecendo à prioridade das demandas e a possíveis regras adicionais que sejam estabelecidas pelo usuário. No caso de PCs com reservatório, o método *Raciona* aplicará a regra operacional que lhe for pertinente, também, por escolha do usuário. Aqui, tem-se mais um exemplo de aplicação do mecanismo de polimorfismo (Viegas Fº, 2000).

Cabe, ainda, aqui, distinguir a função dos métodos *Opera* e *Raciona* quando aplicados a Reservatórios: o primeiro, trata da operação tática do Reservatório - como já foi explicado - de modo a que a água existente possa ser distribuída, segundo as regras estabelecidas pelo usuário, visando o atendimento de todas as Demandas atinentes ao mesmo, independentemente, delas pertencerem ou não ao seu próprio PC; por outro lado, o método Raciona deveria tratar apenas do racionamento de água no próprio PC onde se encontra o Reservatório, quando esta não for suficiente para o atendimento total das demandas ali existentes. Na verdade, no caso de Reservatórios, como ambos métodos podem dar acesso a “scripts”programados pelo usuário, sua ação pode vir a ser redundante. Esta é a razão pela qual é sugerida a separação de serviços acima indicada. Em todos esses casos as estratégias são introduzidas através de rotinas escritas na linguagem PASCAL SCRIPT.

Os pontos de inserção de rotinas PASCAL SCRIPT no PROPAGAR MOO e suas finalidades.

As rotinas PASCAL SCRIPTS são constituidas por textos ASCII, cujos arquivos ficarão ligados a pontos específicos dentro do aplicativo. Esses pontos, são na verdade, interfaces para as propriedades (atributos) que em cada classe as deve conter e a partir de onde serão compiladas e executadas. A Figura 2, abaixo, ilustra um Editor Pascal Script que facilita a edição dessas rotinas dentro do PROPAGAR MOO.

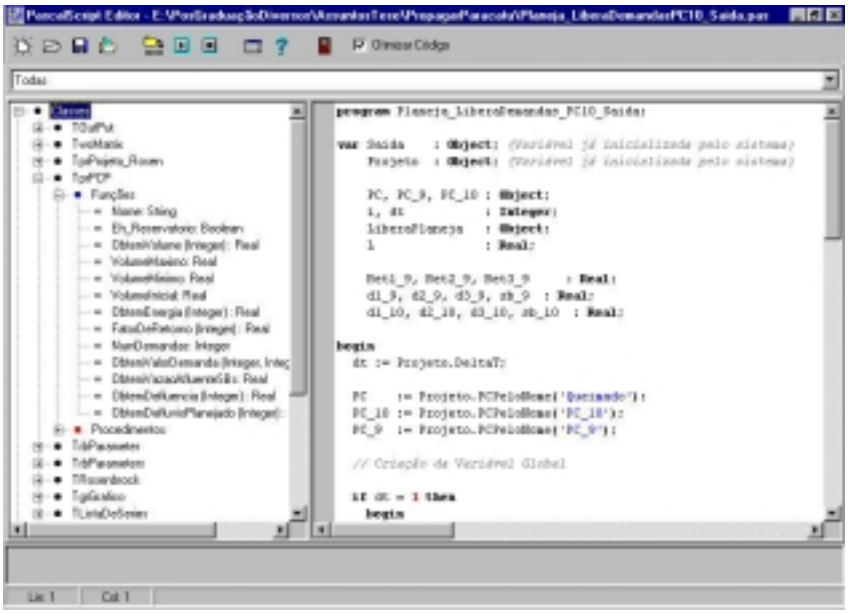

Figura 2 - Editor PascalScript do PROPAGAR MOO.

A Figura 3, a seguir, apresenta os diferentes níveis de inserção de rotinas PASCAL SCRIPT possíveis de serem programadas pelo usuário, no contexto do PROPAGAR MOO.

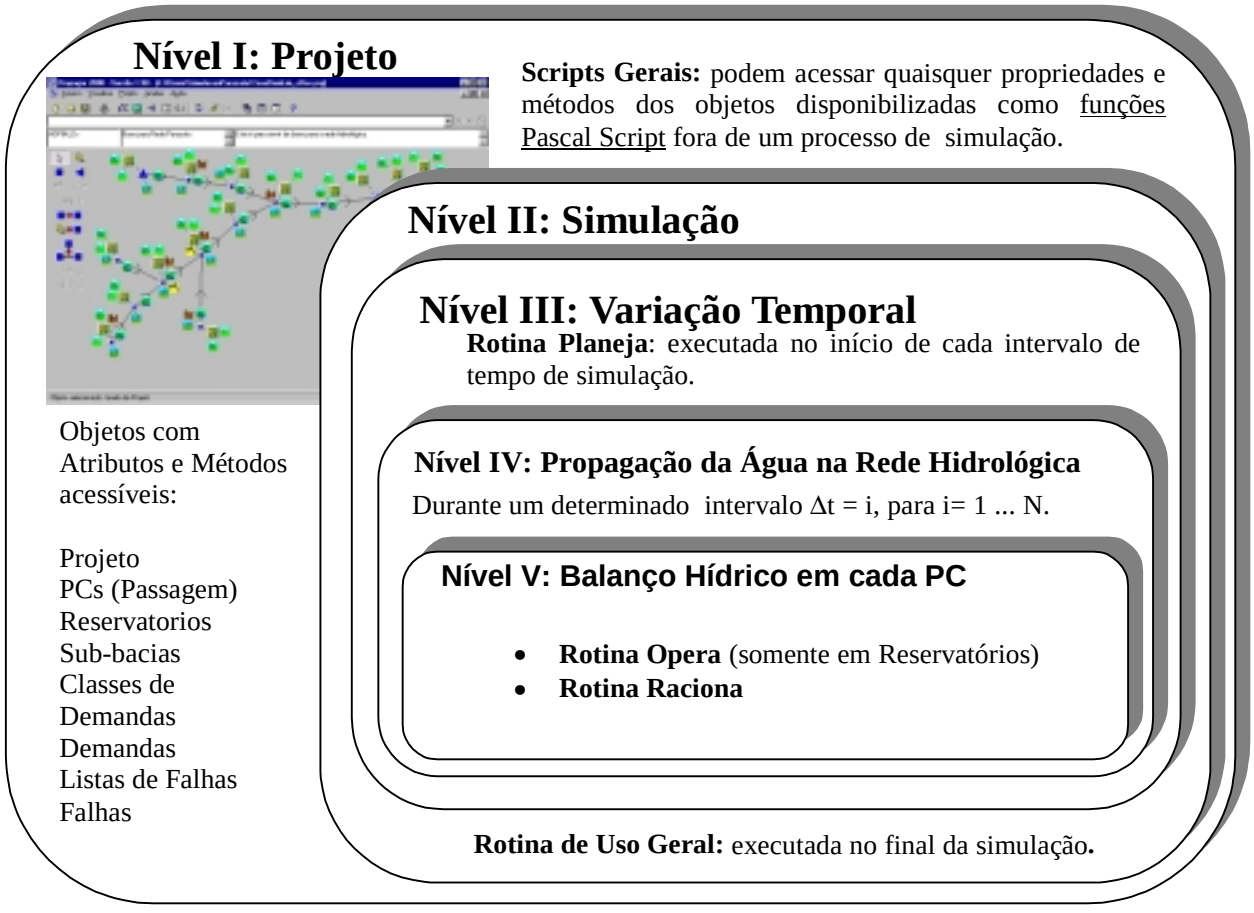

Figura 3 - Níveis de abragência das rotinas PASCAL SCRIPT no PROPAGAR MOO.

O primeiro nível, denominado Nível 1: Projeto, é o mais geral de todos. As rotinas desse nível podem ser executadas a qualquer momento, independentemente da execução da simulação de um determinado projeto. Através das mesmas o usuário pode acessar quaisquer propriedades (atributos) e métodos dos objetos que integram o projeto em questão quando as mesmas estiverem disponibilizadas na forma de funções PascalScript. É importante que se perceba que tanto os dados de cada objeto da Rede Hidrográfica, tais como, volumes máximos e mínimos de reservatórios, valores de demandas, dentre outros, como, também, os resultados decorrentes da execução de uma simulação, no caso, volumes finais em reservatórios, volumes afluentes de sub-bacias, valores correspondentes ao atendimento das demandas, bem como outros, todos referentes a cada intervalo de tempo, ficam disponíveis em cada um de seus respectivos objetos. Dessa forma, o usuário pode, através desse nível de rotinas, realizar a manipulação que desejar sobre os mesmos, tais como cálculos estatísticos, elaboração de relatórios na forma de planilhas e gráficos ilustrativos, dentre outras possibilidades.

A execução das rotinas PASCAL SCRIPT correspondente aos nível seguintes, Nível 2, Nível 3 e Nível 4, já ocorrem durante a execução de uma simulação. A chamada da rotina referente ao Nível 3, por exemplo, é feita no início de cada intervalo de tempo, correspondendo à rotina Planeja do Usuário. Nela o usuário pode testar o planejamento estratégico que desejar. A rotina relativa ao Nível 4, acontece por ocasião da realização do balanço hídrico em cada PC, tanto para os reservatórios através da OPERA e da RACIONA, como para os PCs onde não existirem armazenamento, quando apenas a RACIONA estará disponível. Na OPERA o usuário poderá testar as regras operacionais que desejar e na RACIONA as de racionamento de água. A rotina PASCAL SCRIPT referente ao Nível 2, denominada Rotina de Uso Geral, é chamada ao final da simulação. O usuário pode utilizá-la para quaisquer finalidades que desejar e que devam ser realizadas neste momento. Exemplos disso são a apresentação dos resultados em objetos do tipo Planilha e do tipo Gráfico criados durante a execução da simulação, bem como a liberação da memória utilizada pelos mesmos, através da destruição desses objetos. Além disso, o usuário pode utilizar-se de quasiquer dessas rotinas para realizar toda e qualquer interferência que desejar sobre o projeto, ou, então, para conferir resultados, realizar testes com os próprios algoritmos que estiver criando, e assim por diante.

EXEMPLO DE USO DE ROTINAS PASCAL SCRIPT APLICADAS AO PROPAGAR MOO.

Um exemplo construído sobre a Bacia do Rio Paracatu.

A aplicabilidade de rotinas PASCAL SCRIPT ao PROPAGAR MOO, foi testada em Viegas Fº (2000), através de uma aplicação-exemplo, realizada sobre a Bacia do Rio Paracatu-MG, a qual foi escolhida, principalmente, em virtude de ser essa, uma bacia hidrográfica com uma razoável quantidade de dados disponíveis, tanto no aspecto de disponibilidade hídrica como, também, de demandas hídricas. Não houve, em momento algum, maior preocupação de fazer-se uma análise real sobre as condições da bacia em qualquer sentido que fosse.

A Bacia do Rio Paracatu, localiza-se entre as coordenadas 15º 25’ e 18º 40’ de latitude sul e 45º 03’ e 47º 40’’ de longitude oeste, com uma área de 45.625 km^2, distribuída entre os estados de Minas Gerais (92,3%) e Goiás (4,5%) e o Distrito Federal (3,2%). Integra a sub-bacia do Alto-Médio São Francisco, ficando sua confluência com esse curso de água localizada entre as cidades de Pirapora e São Romão. Apresenta como suas principais sub-bacias, pela margem direita, a do Rio da Prata com 3.750 km^2 e a do Rio do Sono com 5.969 km^2 e, pela margem esquerda, a do Rio Escuro, com 4.347 km^2, a do Rio Preto, com 10.459 km^2, e a do Ribeirão Entre-Ribeiros com 3973 km^2 (Plano Diretor da Bacia do Rio Paracatu, 1999).

A representação da Rede Hidrográfica referente à bacia encontra-se ilustrada na Figura 4, consistindo de 15 PCs, dos quais dois, o PC-8 e o PC-12, são cenarizados como sendo os reservatórios Queimado e Paracatu. Na figura, inclusive, pode-se ver a Rede Hidrográfica lançada sobre uma imagem do mapa da bacia sobre a Área de Projeto.

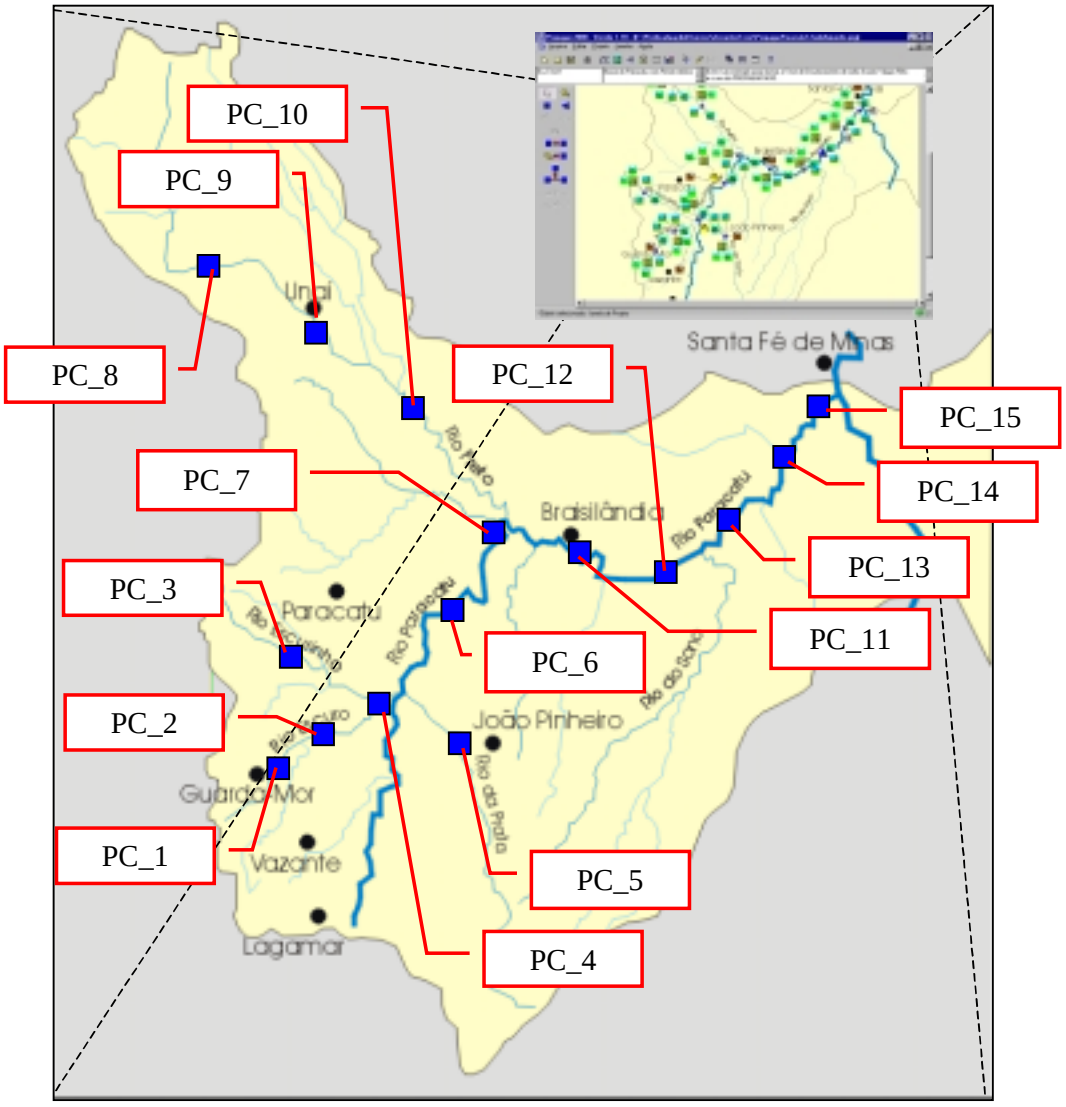

Figura 4 - Mapa da Bacia do Rio Paracatu com localização dos PCs e inserção como figura de fundo da Área de Projeto no PROPAGAR MOO. (Fonte: Viegas Fº, 2000).

As demandas hídricas de caráter difuso consideradas foram aquelas referentes a pecuária e a população rural. Para demandas localizadas foram consideradas as demandas urbanas, a demanda ecológica e as demandas de irrigação por aspersão e, também, por inundação. Uma descrição detalhada das mesmas encontra-se em Viegas Fº (2000).

Um exemplo de rotina PLANEJA aplicada à Bacia do Rio Paracatu.

O exemplo de uso de “scripts” foi feito através da construção de uma rotina Planeja visando, estabelecer uma estratégia de operação de reservatório. Relembrando o que foi dito acima, essa rotina é executada, no Nível 3, da variação temporal, no início de cada intervalo de tempo, antes de ser iniciada a propagação da água através da Rede. No caso presente, a rotina teve por objetivo liberar água no Reservatório Queimado (PC_8) para atender as Demandas Hídricas no PC_10, o qual apresenta falhas para a Demanda Terciária (Demanda de Irrigação) conforme pode-se ver na Tabela 1. Essa tabela, na verdade, apresenta planilhas de falhas da Rede Hidrográfica, segundo três aspectos: primeiro, resultante de uma simulação sem reservatórios, depois, com os reservatórios e, por último, com os reservatórios e a rotina Planeja. Nela as falhas referentes às Demandas Primárias, Secundárias e Terciárias, respectivamente, no caso em estudo, Demandas Urbanas, Demandas Ecológicas e Demandas de Irrigação, são apresentadas nessa ordem em três conjuntos de colunas, por intervalo (INTs), quando críticas (CRITs) e anuais(ANOs). No exemplo, foram consideradas como críticas as falhas de atendimento superiores a 50%. Pode-se ver, na tabela, que, para o PC_10, existem, respectivamente, 3, 2 e 3 falhas em um cenário sem reservatório; 2, 0 e 2 falhas quando com reservatórios mas sem nenhum critério de operação estabelecido; e, finalmente, nenhuma falha quando, com o uso da PLANEJA, existe liberação programada de água para o atendimento das demandas ali existes.

O Quadro 2, acima, ilustra o código da rotina Planeja escrito na linguagem Pascal Script. É importante que se tenha em mente que esse é chamado pelo PROPAGAR MOO, à cada vez, como um programa à parte, que chega, cumpre sua função e é descartado. Logo, o escopo de todas suas variáveis de trabalho é local e, mais do que isso, não existem passagens de parâmetros, como no caso de sub-rotinas. Desse modo, à cada chamada, as variáveis têm de ser, necessariamente, inicializadas. Assim, no caso da rotina Planeja, no início de cada intervalo de tempo, essa chamada é feita, o ‘script’ é executado, realiza as tarefas que lhe estão programadas e é descartado. Isso significa que, em um determinado intervalo de tempo *DeltaT*, ele não “lembrará” o valor das variáveis com as quais trabalhou no intervalo anterior, como, por exemplo, o próprio `dt`. Logo, esse valor e todos os que forem necessários têm de ser inicializados para cada intervalo.

Quadro 2 - Código da rotina Planeja_LiberaDemandas_PC10.

```
Program Planeja_LiberaDemandas_PC10;

var Saida    : Object; {Variável já inicializada pelo sistema}
    Projeto  : Object; {Variável já inicializada pelo sistema}

    PC, PC_9, PC_10 : Object;   // Variáveis para conterem os PCs
    Dt              : Integer;  // DeltaT
    Libera          : Real;     // Liberação de água

    Ret1_9, Ret2_9, Ret3_9      : Real; // Retornos PC_9
    D1_9, d2_9, d3_9, sb_9      : Real; // Demandas PC_9
    D1_10, d2_10, d3_10, sb_10  : Real; // Demandas PC_10

Begin
  // Obtém o intervalo DeltaT atual
  dt := Projeto.DeltaT;

  // Inicializa as variáveis PC, PC_10 e Pc_9 com os
  // PCs que lhe correspondem.
  PC    := Projeto.PCPeloNome('Queimado');
  PC_10 := Projeto.PCPeloNome('PC_10');
  PC_9  := Projeto.PCPeloNome('PC_9');

  // Inicializa as variáveis de retorno do PC_9 com os valores que
  // estão armazenandos nas propriedades.
  Ret1_9 := PC_9.FatorDeRetorno(1);
  Ret2_9 := PC_9.FatorDeRetorno(2);
  Ret3_9 := PC_9.FatorDeRetorno(3);

  // Inicializa as variáveis das Demandas e Vazões Afluentes das
  // Sub-Bacias do PC_9 e PC_10.
  //  1, 2, 3 => indica a prioridade
  //  'T'     => indica Demanda Total
  d1_9 := PC_9.ObtemValorDemanda(dt, 1, 'T');
  d2_9 := PC_9.ObtemValorDemanda(dt, 2, 'T');
  d3_9 := PC_9.ObtemValorDemanda(dt, 3, 'T');
  sb_9 := PC_9.ObtemVazaoAfluenteSBs;

  d1_10 := PC_10.ObtemValorDemanda(dt, 1, 'T');
  d2_10 := PC_10.ObtemValorDemanda(dt, 2, 'T');
```

```
  d3_10 := PC_10.ObtemValorDemanda(dt, 3, 'T');
  sb_10 := PC_10.ObtemVazaoAfluenteSBs;

  // Cálculo do valor a ser liberado
  Libera := (1- Ret1_9)* d1_9 + (1- Ret2_9)* d2_9 + (1- Ret3_9)* d3_9
            - sb_9 + d1_10 + d2_10 + d3_10 - sb_10;
  if Libera < 0 then Libera := 0;

  // Atribui o valor planejado à propriedade AtribuiDefluvioPlanejado.
  PC.AtribuiDefluvioPlanejado( dt, Libera);

end.
```

Dessa forma, é recomendável, embora não obrigatório, que a primeira parte do corpo de um 'script' constitua-se pela inicialização das variáveis de trabalho, garantido que o usuário não se esqueça de fazê-lo. No caso do exemplo presente, a inicialização engloba o intervalo de tempo atual (`dt`), os objetos PC, PC_9, PC_10, bem como, os valores das demandas hídricas e vazões afluentes das sub-bacias para cada um deles e, ainda, os valores dos fatores de retornos do PC_9.

Deve-se observar que a inicialização é feita através de procedimentos ou funções disponíveis na linguagem Pascal Script que se está utilizando, mas que, entretanto, não são nativos da mesma. No caso do exemplo que estamos tratando, essas funções são:

PCPeloNome(Nome: String): Object	Função que acessa a lista de PCs de um Projeto, usando o nome de um PC, e o atribui como objeto a uma váriavel do tipo **Object** criada pelo usuário no script. Ex.: PC := Projeto.PCPeloNome('Queimado');
FatorDeRetorno(Prioridade: Integer): Real	Função da Classe TprPCP que retorna o Fator de Retorno referente às Demandas Totais da prioridade que lhe é passada como parâmetro. Ex.: Ret1_9 := PC_9.FatorDeRetorno(1);
ObtemValorDemanda(dt, p, T) : Real	Função da Classe TprPCP que retorna o valor de

uma Demanda Hídrica em função dos parâmetros:

dt : Integer => intervalo de tempo;

p : Integer => prioridade (1, 2 ou 3);

T : String => tipo (‘T’ = Total;

‘A’ = Atendida;

‘P” = Planejada)

Ex.:

```
d2_10 :=
  PC_10.ObtemValorDemanda(dt, 2, 'T');
```

Depois de feita a inicialização das variáveis é, então, escrito o código correspondente ao que se deseja que a rotina faça. No “script” da Planeja, o propósito é o da liberação de água pelo Reservatório Queimado para atender as Demandas do PC_10. Para tanto, o cálculo feito corresponde à determinação da água necessária para o atendimento de todas as demandas do PC_9 e PC_10, menos a água que aflui das suas próprias sub-bacias e o que retorna ao rio do PC_9. Depois, é feita uma avaliação para verificar se o valor resultante não tornou-se negativo, o que pode acontecer quando a afluência das sub-bacias, por si só, já for suficiente para atender às demandas. Quando isso ocorrer a liberação é tornada nula, caso contrário é feita conforme o valor calculado, a seguir ilustrado:

```
Libera := (1- Ret1_9)* d1_9 + (1- Ret2_9)* d2_9 + (1- Ret3_9)* d3_9
             - sb_9 + d1_10 + d2_10 + d3_10 - sb_10;
if Libera < 0 then Libera := 0;
```

Tabela 1 - Falhas no atendimento das Demanda Hídricas em todos os PCs - simulação com Demandas Hídricas limitadas.

TeseDemLim_sR - Sem Reservatório.									
	INTs	**CRITs**	**ANOs**	**INTs**	**CRITs**	**ANOs**	**INTs**	**CRITs**	**ANOs**
PC_1	3	0	3	3	3	3	6	4	5
PC_2	0	0	0	0	0	0	6	1	5
PC_3	1	0	1	0	0	0	4	2	2
PC_4	0	0	0	0	0	0	1	0	1
PC_5	0	0	0	0	0	0	5	1	3
PC_6	0	0	0	0	0	0	0	0	0

PC_7	0	0	0	0	0	0	7	2	6
Queimado	0	0	0	0	0	0	11	2	8
PC_9	0	0	0	0	0	0	0	0	0
PC_10	0	0	0	0	0	0	3	2	3
PC_11	0	0	0	0	0	0	0	0	0
Paracatu	0	0	0	0	0	0	1	0	1
PC_12	0	0	0	0	0	0	1	0	1
PC_14	0	0	0	0	0	0	0	0	0
PC_15	0	0	0	0	0	0	0	0	0

TeseDemLim_cR - Com Reservatório.

	INTs	CRITs	ANOs	INTs	CRITs	ANOs	INTs	CRITs	ANOs
PC_1	3	0	3	3	3	3	6	4	5
PC_2	0	0	0	0	0	0	6	1	5
PC_3	1	0	1	0	0	0	4	2	2
PC_4	0	0	0	0	0	0	1	0	1
PC_5	0	0	0	0	0	0	5	1	3
PC_6	0	0	0	0	0	0	0	0	0
PC_7	0	0	0	0	0	0	7	2	6
Queimado	0	0	0	0	0	0	0	0	0
PC_9	0	0	0	0	0	0	0	0	0
PC_10	0	0	0	0	0	0	2	0	2
PC_11	0	0	0	0	0	0	0	0	0
PC_12	0	0	0	0	0	0	1	0	1
Paracatu	0	0	0	0	0	0	0	0	0
PC_14	0	0	0	0	0	0	0	0	0
PC_15	0	0	0	0	0	0	0	0	0

TeseDemLim_cR - Com Reservatório e Planeja para PC_10

	INTs	CRITs	ANOs	INTs	CRITs	ANOs	INTs	CRITs	ANOs
Paracatu	0	0	0	0	0	0	0	0	0
PC_1	3	0	3	3	3	3	6	4	5
PC_2	0	0	0	0	0	0	6	1	5
PC_3	1	0	1	0	0	0	4	2	2
PC_4	0	0	0	0	0	0	1	0	1
PC_5	0	0	0	0	0	0	5	1	3
PC_6	0	0	0	0	0	0	0	0	0
PC_7	0	0	0	0	0	0	7	2	6
Queimado	0	0	0	0	0	0	0	0	0
PC_9	0	0	0	0	0	0	0	0	0
PC_10	0	0	0	0	0	0	0	0	0
PC_11	0	0	0	0	0	0	0	0	0
PC_12	0	0	0	0	0	0	0	0	0

PC_14	0	0	0	0	0	0	0	0	0
PC_15	0	0	0	0	0	0	0	0	0

Evidentemente que a quantidade de água necessária no PC_9, menos o que dela é retornado ao rio, tem ser considerada para evitar que, caso isso não seja feito, o mesmo aproprie-se da água destinada ao PC_10. Para simplificar o exemplo, resumiu-se o planejamento apenas na determinação das necessidades de atendimento das demandas hídricas do PC_10. Deixou-se para a rotina BalancoHidrico a verificação da consistência do planejamento quanto à existência de água para satisfazer os valores planejados - o que sempre é feito pelo PROPAGAR MOO; caso não existir água, durante o transcorrer da propagação, esta não será liberada. Regras mais sofisticadas, levando em conta o volume do reservatório no início do período ou quaisquer outras considerações que envolvam a utilização das informações contidas nos objetos da Rede, podem ser construídas. Inclusive, pode ser combinada a ação conjunta e seqüencial de diferentes rotinas como é caso da Opera e da Raciona, já mencionadas.

Conforme pode-se ver, através do exame da Tabela 1, a utilização da rotina PLANEJA determinou o suprimento de água, pelo reservatório Queimado, necessário à eliminação das falhas no atendimento das demandas hídricas no PC_10, demonstrando a funcionalidade da rotina PASCAL SCRIPT utilizada com esse propósito.

CONCLUSÕES E RECOMENDAÇÕES.

A utilização de aplicativos desenvolvidos à luz da Modelagem Orientada a Objetos (MOO), como é o caso do PROPAGAR MOO, construído a partir da Biblioteca de Classes aplicada ao Planejamento de Uso da Água em Sistemas de Recusos Hídricos, baseados em Redes Hidrográficas, vem trazer uma nova dinâmica da modelagem nesse contexto, bem como reduzir o “gap” existente entre o que é produzido nos ambientes de pesquisa e o que se aplica na prática profissional, em virtude da aridez das interfaces e a dificuldade de utilização dos modelos. Entretanto, por outro lado, a dificuldade de o usuário comum lidar com os ambientes de desenvolvimento tais como o C++ e o DELPHI, quando surgir a necessidade de serem programadas rotinas internas a esses modelos, adequando-os às suas necessidades específicas, enfatiza a necessidade de serem utilizadas linguagens mais simples, intermediárias e procedurais, que, “em tempo de execução”, permitam a programação de tais rotinas, como é o caso da PASCAL SCRIPT.

Em virtude do exposto anteriormente, verifica-se que a linguagem PASCAL SCRIPT apresenta-se como um instrumento adequado para permitir aos usuários do PROPAGAR MOO, a programação das rotinas básicas do modelo, PLANEJA, OPERA e RACIONA, bem como outras,

em diferentes níveis abrangência dentro do aplicativo, controlando as diferentes etapas da simulação, analisando resultados e produzindo saídas na forma de textos ASCII, planilhas e gráficos.

Recomenda-se, portanto, o estímulo ao uso da mesma e, também, o desenvolvimento de rotinas de análise estatística, econômica, de controle de simulação, dentre outras, promovendo, inclusive, a criação de um número cada vez maior funções PASCAL SCRIPT com diferentes especificidades.

BIBLIOGRAFIA.

CHAVES, E. M. B. (1993). *Propostas para o Planejamento da Bacia do Rio Mosquito no norte de Minas Gerais*. Dissertação de Mestrado. Curso de Pós-Graduação em Engenharia de Recursos Hídricos e Saneamento Ambiental. da Universidade Federal do Rio Grande do Sul. Porto Alegre. 210p.

LANNA, A. E. *SAGBAH – Sistema de Apoio ao Gerenciamento de Bacias Hidrográficas – versão 97*. IPH-UFRGS. 1997. 25p.

LOUCKS, D.P.; KINDLER, J.; FEDRA, K. (1985) Interactive Water Resource Modeling and Model Use: An Overview. *Water Resouce Research*, Vol. 21, Nº 2, Fevereiro 1985, p. 95-102.

PORTO, R.L., AZEVEDO, L. G. T. (1997). Sistema de Suporte a Decisões Aplicados a Problemas de Recursos Hídricos. In: PORTO, R.L. *Técnicas quantitativas para o gerenciamento de Recursos Hídricos*. Editora da Universidade – UFRGS : ABRH. Porto Alegre. 420p.

SIMONOVIC, S. P. (1992). Reservoir Systems Analysis: Closing Gap between Theory and Practice. *Journal of Water Resource Planning and Management*. Vol. 118, No. 3, Maio/Junho, 1992. ASCE. p. 262-280.

VIEGAS Fº, J. S., (2000). *O Paradigma da Modelagem Orientada a Objetos Aplicada a Sistemas de Apoio à Decisão em Sistemas de Recursos Hídricos*. Porto Alegre: UFRGS. Tese (Doutorado) - Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Saneamento Ambiental, IPH-UFRGS. 547p.

VIEGAS Fº, J. S., (2001a). O Paradigma da Modelagem Orientada a Objetos Aplicado a Sistemas de Recursos Hídricos (I) - Modelo de Objetos Básico para um Rede Hidrográfica. Artigo submetido à Revista Brasileira de Recurosos Hídricos.

VIEGAS Fº, J. S., (2001b). O Paradigma da Modelagem Orientada a Objetos Aplicado a Sistemas de Recursos Hídricos (II) - Modelo de Objetos Aplicado ao Planejamento de Uso da Água - PROPAGAR MOO. Artigo submetido à Revista Brasileira de Recurosos Hídricos.

VIEGAS Fº, J. S., LANNA, A. E. (1999). *SAGBAH 2000 - Manual Básico*. Porto Alegre: IPH-UFRGS e FEA/IFM-UFPel. 17p.