Um metodo para a programacao de software adaptativo

Salvador Ramos Bernardino da Silva e Joao Jose Neto

Escola Politécnica da Universidade de Sao Paulo
sramosbs@usp.br,joao.jose@poli.usp.br
http://www.poli.usp.br

Resumo Os métodos (frequentemente chamados de metodologias) de programaçao costumam compreender um conjunto de recomendaçoes e procedimentos de uso para os recursos disponibilizados por uma linguagem de programaçao, com o objetivo de disciplinar o seu emprego na produçao de programas para uso em diversas áreas de aplicaçao. O presente trabalho é dedicado à programaçao adaptativa, e propoe um método para o desenvolvimento desse tipo de software, embasado na equivalencia que pode ser observada entre estruturas de controle de diversas naturezas, que tem sido empregadas tradicionalmente em programaçao. Para isso, propoe-se aqui uma possível forma de se converter, para o estilo adaptativo, programas imperativos que se utilizam dessas estruturas tradicionais. Um exemplo ilustrativo encerra este trabalho, mostrando a aplicaçao do método proposto para a implementaçao de um projeto simples, mas relativamente abrangente, de software adaptativo.

Keywords: Adaptatividade, programas automodificáveis, método de programaçao adaptativa

1 Introducao e historico

Os primeiros programadores, em meados do seculo passado, utilizavam a automodi cacao de codigo como forma de otimizar os escassos recursos das maquinas entao dispon veis. A introducao, na decada de 1970, da pratica da formalizacao dos metodos de desenvolvimento de programas, levou ao surgimento da Engenharia de Software, a qual praticamente extinguiu, na pratica, o uso da automodi cacao de codigo do processo de desenvolvimento de programas [1].

No entanto, observa-se que, nos ultimos anos, vem novamente sendo reintroduzido o uso dessa tecnica, a qual tem revelado a sua possibilidade (e conveniencia) de aplicacao em uma ampla gama de tipos de problemas, tais como: protecao de programas [2], melhora das tecnicas de obtencao de checksum [3], tratamento de linguagens naturais, reconhecimento de linguagens formais e diversas outras possibilidades [4].

As maquinas antigas nao dispunham de muitos recursos, o que exigia dos seus programadores certa habilidade para o desenvolvimento de programas que fossem capazes de executar as tarefas necessarias sem extrapolar em suas exigencias

de recursos computacionais. Em particular, a memoria se mostrava um recurso escasso, e os programas cresciam cada vez mais, exigindo que o programador executasse muitas manobras e artif cios em seus programas com a nalidade de burlar essa limitacao, que lhe impunha restricoes crescentes.

A utilizacao de linguagens de baixo n vel (linguagens de maquina, linguagens de montagem e mesmo linguagens de macro-montagem), juntamente com a pratica da automodi cacao de codigo, foram, entao, excelentes formas que encontraram os programadores para conviverem com a escassez de memoria, enquanto desenvolviam programas que dela faziam exigencias cada vez maiores.

A entao notoria escassez de fundamentos teoricos e de metodos de desenvolvimento de programas automodi caveis muito contribuiu para que tais programas se revelassem ao menos, do ponto de vista de usuarios que buscavam a solidez de programas cujo comportamento gostariam de poder antecipar cada vez menos atraentes para uso extensivo, devido as situacoes de vulnerabilidade exibida por tais programas em decorrencia da uidez e imprevisibilidade acarretadas pela modi cacao espontanea de tais programas, e da decorrente di culdade de projeto, de manutencao, de compreensao, e de rastreamento que exibiam programas assim constru dos.

Visando a criacao de um metodo adequado para a construcao de programas automodi caveis de boa qualidade, o presente trabalho propoe uma serie de procedimentos que buscam guiar o desenvolvimento de programas dessa natureza, tambem denominados programas adaptativos.

Este artigo contem uma proposta de sistematizacao da forma de produzir programas adaptativos usando para tanto, como ferramenta, uma linguagem de programacao especialmente projetada para essa nalidade, acompanhada por um metodo aderente ao mesmo tipo de estrutura cuja utilizacao e propiciada pela linguagem.

Essa aderencia entre a linguagem e o metodo aqui apresentado e viabilizada devido a adocao, em ambas, de uma mesma arquitetura para o software adaptativo a que se referem, estimulando assim o programador a utilizar-se, para os problemas que estiver resolvendo, de uma modelagem apropriada para o uso das ferramentas em questao.

2 Elementos da Programacao Adaptativa

Um metodo de programacao adaptativa e um conjunto de recomendacoes que orientam a pratica de construcao de programas adaptativos de boa qualidade, fornecendo ao programador recursos para evitar a usual vulnerabilidade, imprevisibilidade e di culdade de leitura, de projeto e de manutencao em decorrencia dos quais por tanto tempo os programas com codigo dinamico foram evitados na pratica.

Nos programas adaptativos podem sempre ser identi cados os dois conceitos que caracterizam os dispositivos adaptativos: um dispositivo subjacente inicial nao-adaptativo e uma camada adaptativa [5]. Neste estudo, o dispositivo subjacente e um programa de comportamento estatico, que representa o programa

adaptativo ao in cio da execucao (suas instrucoes, de maquina real ou virtual, correspondem as regras dos dispositivos adaptativos teoricos), e a camada adaptativa e o conjunto das acoes adaptativas (chamadas de funcoes especiais, ligadas as instrucoes do programa subjacente) que impoem modi cacoes ao programa, quando da sua execucao.

2.1 Formas convencionais de representacao de fenomenos repetitivos

Das situacoes usuais em que e evidente a possibilidade de uso da adaptatividade, destaca-se a representacao de fenomenos que se repetem, dentro de um programa, sendo frequentes ao longo de todas as fases do desenvolvimento de um programa: estudo do problema, modelagem, codi cacao, compilacao e execucao do programa.

Na codi cacao de modelos usando linguagens de programacao, a partir de abstracoes representando componentes do programa, surgem naturalmente as declaracoes de estruturas de dados, de funcoes, subrotinas e macros, possivelmente parametricos.

Algumas repeticoes sao entao resolvidas manualmente pelo programador, que se incumbe de criar comandos, no programa, que re itam atraves da simples copia f sica dos comandos que realizam as repeticoes conceituais pretendidas das abstracoes a serem repetidas no programa. Isso so deve ser feito, todavia, em situacoes muito especiais.

Uma alternativa melhor, adotada por muitas linguagens de programacao, oferece comandos e declaracoes espec cas para a codi cacao de estruturas repetitivas, e automaticamente traduzidas, pelo compilador, para um codigo executavel equivalente, cabendo ao compilador determinar sua forma de realizacao, em linguagem de maquina, como procedimentos abertos (macros, expandidos em tempo de compilacao, resultando programas rapidos e longos) ou fechados (subprogramas, chamados em tempo de execucao, resultando programas compactos porem mais lentos).

Pouco tem sido explorado o uso da adaptatividade como variante para a utilizacao de macros, cuja expansao e postergada para a epoca da execucao nos programas adaptativos. Neste artigo, mostra-se que situacoes usualmente representadas atraves de tecnicas tradicionais podem ser alternativamente codi cadas em forma adaptativa, caso essa opcao se mostre proveitosa na particular situacao do programador.

2.2 Modelo de camadas para software adaptativo

Em [6], descreve-se a sintaxe de uma linguagem para programacao adaptativa, e o funcionamento da camada adaptativa dos programas nela desenvolvidos.

Nela, guram os elementos necessarios para representar programas adaptativos, mas, se conveniente, pode ser usada tambem para codi car uxogramas nao adaptativos.

Pode-se sintetizar brevemente a descricao da arquitetura de um programa adaptativo concebido de acordo com o modelo em que se baseia essa linguagem, considerando suas quatro camadas mutuamente ortogonais: (i) a primeira camada, povoada por um conjunto de blocos basicos, representando operacoes monol ticas com uma so entrada e uma so sa da, nesse n vel de abstracao; (ii) a segunda, que abriga um conjunto de decisores, cada qual acoplado diretamente a sa da de um bloco basico, criando sa das separadas para cada caso poss vel, considerada a variedade existente de condicoes de sa da produzidas pelo bloco basico associado; (iii) a terceira, associada a um conjunto de acoes adaptativas (chamadas parametricas de funcoes adaptativas), responsaveis pela automodi cacao propriamente dita do programa, posicionadas nas extremidades dos conectores, entre uma sa da de algum decisor e um conector (acao adaptativa anterior), ou entre um conector e a entrada de um bloco basico (acao adaptativa posterior); (iv) a ultima, compreendendo um conjunto de conectores, que estabelecem as ligacoes existentes entre as sa das dos diversos decisores e as entradas dos blocos basicos.

A representacao gra ca de programas adaptativos, constru dos em conformidade com o modelo em questao, apresenta o aspecto mostrado na Figura 1, no qual se pode observar a independencia entre as camadas. E facil observar que,

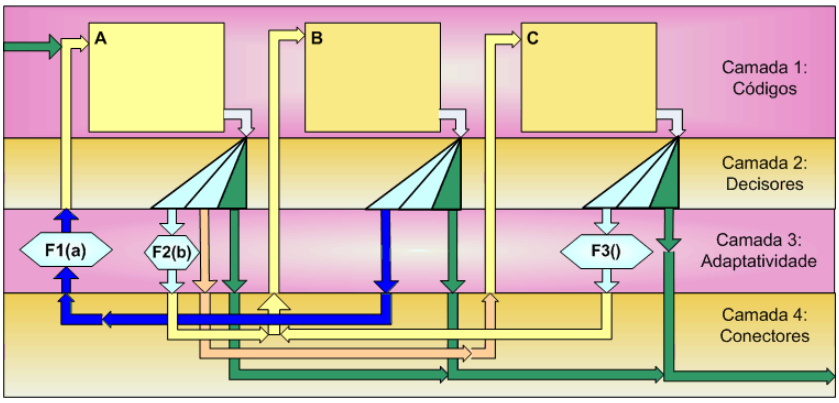

Figura 1. Gráfico de programa adaptativo, idealizado segundo o modelo de camadas.

removendo-se desse diagrama as acoes adaptativas, o programa representado se reduz a um uxograma sequencial, portanto, essa notacao se aplica igualmente a representacao de logicas estaticas e de codigos automodi caveis.

2.3 Implementacao adaptativa de fenomenos repetitivos.

Conforme foi discutido, elementos repetitivos de um programa costumam ter implementacao atraves do uso de tecnicas iterativas ou recursivas, e atraves de abstracoes representadas por procedimentos abertos ou fechados.

As acoes associadas ao tratamento de tais elementos repetitivos ocorrem em epoca de projeto, de compilacao (repeticoes expandidas pelo compilador, a partir das chamadas de macros empregadas no programa) ou de codi cacao (usando-se repeticao expl cita).

Em programas de codigo estatico, usualmente sao processados, em tempo de execucao, os comandos repetitivos disponibilizados pelas linguagens de programacao, bem como as chamadas de procedimentos, acionadas pelo programa ao longo da sua execucao, enquanto em programas automodi caveis, explora-se a modi cacao dinamica do programa durante a sua execucao, promovida pela execucao de acoes adaptativas.

Em particular, repeticoes implementadas adaptativamente se resumem a um processo analogo ao da expansao de macros, porem efetuada nao durante a compilacao, como em programas de codigo estatico, mas apenas em tempo de execucao, e apenas o numero de vezes que for estritamente necessario para a particular atividade que estiver sendo executada em cada caso.

3 Programacao Adaptativa

Na linguagem para programacao adaptativa que foi proposta, podem ser identi cadas construcoes sintaticas que re etem a estrutura dos programas automodi caveis desenvolvidos com a arquitetura acima resumida: (i)Uma linguagem hospedeira nao especi cada pode ser empregada para exprimir a logica do programa nao-adaptativo subjacente, em particular a dos blocos basicos. A logica global do programa pode ser complementada usando-se, adicionalmente, todos os demais recursos da linguagem, exceto as acoes adaptativas. Como nao foram feitas restricoes para os blocos basicos, nada impede que cada um deles possa ser, por sua vez, outro programa automodi cavel. Isso proporciona a possibilidade de um projeto com arquitetura hierarquica, abrindo assim possibilidades metodologicas adicionais. (ii) A camada adaptativa oferecida pela linguagem poe a disposicao do programador, alem das declaracoes e das chamadas das funcoes adaptativas, a possibilidade de implementar seu programa adaptativo a partir de uma concepcao baseada em um modelo em camadas aderente a arquitetura que foi apresentada anteriormente.

Detalhes sobre a linguagem proposta podem ser encontrados em [6]. Especi camente, nessa linguagem, dispoe-se das seguintes construcoes sintaticas: (i) Declaracao dos blocos basicos: efetuada atraves dos comandos `code`; (ii) declaracao das funcoes adaptativas, que descrevem a logica parametrizada, representada pelas acoes adaptativas mencionadas no programa: `adaptive function`; (iii) As acoes adaptativas, referentes as chamadas de funcoes adaptativas declaradas no programa (indicando se a aplicacao da acao adaptativa deve ser anterior ou posterior a transferencia de controle entre os blocos basicos de origem e de destino da conexao a qual estao associadas): `perform ... before/after`; (iv) Declaracao dos decisores, indicando para a sa da de um bloco basico, qual deve ser a entrada do proximo bloco basico a ser executado, em funcao do valor de sa da calculado pelo bloco basico ao qual o decisor estiver associado: especi ca-

dos atraves dos comandos de multipla escolha `case`; (v) Declaracao das conexoes condicionais (representadas por um decisor) existentes entre a sa da de um bloco basico e a entrada de outro: `connect ... to`

4 Metodo Proposto

As recomendacoes aqui descritas visam conduzir a estruturacao de um programa que resulte em um modelo aderente ao mencionado no item 2.2. Este metodo foi idealizado considerando a existencia de uma linguagem compat vel, como e o caso da linguagem citada no item 3, sendo portanto vantajoso, embora nao obrigatorio, que ambos sejam utilizados em conjunto.

Conforme mencionado, estruturas e comportamentos abstratos que se repetem sao muito frequentes em programacao, e podem ser representados nos programas, de maneira convencional, atraves de subprogramas, de forma aberta (por exemplo, como repeticoes expl citas de codigo, ou como macros) ou fechada (por exemplo, na forma de funcoes, subrotinas, procedimentos, metodos etc.).

Em programas que se alteram dinamicamente, identi cam-se igualmente abstracoes dessa natureza, que convivem com outras partes do programa, especi - camente utilizadas para promoverem a automodi cacao do codigo.

Partindo desse fato, e poss vel iniciar-se o desenvolvimento de um programa automodi cavel pela estruturacao de sua logica sem considerar, em um primeiro momento, as alteracoes dinamicas que se pretende para o mesmo. Para isso, e poss vel fazer uso dos recursos e metodos usuais de projeto oferecidos pela Engenharia de Software, originando um programa nao adaptativo subjacente.

O proximo passo consiste em identi car no programa todas as regioes sobre as quais se preve que possa vir a acontecer, durante a execucao do mesmo, quaisquer tipo de modi cacao. Em cada regiao assim caracterizada, determinar exatamente todos os pontos onde ocorrerao as alteracoes, e quais serao essas alteracoes, as quais podem acontecer na forma de retirada ou de insercao de trechos de codigo. A seguir, secciona-se o programa nesses pontos, destacando do restante do programa os pontos de potencial insercao e os trechos a serem potencialmente removidos.

Caso seja observado que alguma estrutura de controle do programa em analise foi seccionada, deve-se reescrever a mesma utilizando estruturas mais simples.

Dessa forma, obtem-se, ao nal, uma particao do programa, em que cada trecho de codigo assim obtido devera ser nomeado e formara, na estrutura proposta, um bloco basico (na linguagem adotada, os blocos basicos sao representados pela estrutura `code`).

A primeira secao do programa adaptativo, em sua versao nal, sera formada pela declaracao de todos os `code`s. Para tanto, a logica dos `code`s deve ser expressa na forma de um trecho de programa, com uma so entrada e uma so sa da. Na linguagem proposta, isso e feito usando-se os comandos de uma linguagem hospedeira.

O uxo de execucao do programa pode agora ser estabelecido, interconectando-se os blocos basicos identi cados anteriormente. A conexao de um bloco basico

aos outros depende dos valores assumidos a sa da do bloco basico de origem, os quais determinam o bloco basico sucessor, naquela ocasiao.

Deve-se assegurar que cada bloco basico tenha apenas uma entrada e uma sa da. A sa da de cada bloco basico deve ser associado um valor numerico inteiro, para que possa ser determinado corretamente o bloco basico que o devera suceder, uma vez terminada a sua execucao.

Como isso ocorre em tempo de execucao, e necessario que o programa preveja todas as condicoes e o rumo a tomar em cada uma delas. Por essa razao, cada bloco basico ca associado a um decisor, o qual determina, para cada valor de sa da associado ao bloco basico de origem, qual dos blocos basicos do programa sera seu sucessor naquela ocasiao.

Para tanto, utilizam-se conectores, que interligam as sa das do decisor associado aos blocos basicos de origem as entradas dos blocos basicos que os sucederao na sequencia de execucao do programa.

Finalmente, estabelecidos todos os valores de sa da dos blocos basicos e tambem os seus blocos basicos sucessores em cada caso, resta apenas utilizar os conectores para interliga-los, estabelecendo assim a topologia inicial do grafo da logica do programa.

A declaracao das conexoes formara a segunda secao do programa.

Uma vez desenvolvida a logica do programa estatico, a fase seguinte consiste em incorporar a ele as chamadas de funcoes adaptativas, bem como as declaracoes das respectivas funcoes adaptativas.

Sao essas funcoes que especi cam as acoes adaptativas elementares a serem executadas, responsaveis pelas modi cacoes atomicas do codigo.

Todas as alteracoes de logica, a serem efetuadas durante a execucao do programa, deverao nessa ocasiao ser especi cadas, na forma de chamadas de funcoes adaptativas.

Na arquitetura adotada, a aplicacao de acoes adaptativas ao programa podem ocorrer em apenas duas situacoes: imediatamente a sa da de um bloco basico, antes que o controle do programa seja transferido ao bloco basico que o sucede (acao adaptativa anterior), ou entao, na situacao inversa (acao adaptativa posterior), apos a transferencia de controle ao bloco sucessor, mas imediatamente antes de este assumir o controle da execucao do programa.

A especi cacao dessas acoes adaptativas se faz acoplando-se as chamadas das respectivas funcoes adaptativas as extremidades das conexoes entre blocos basicos, que estabelecem a topologia do uxograma do programa: acoes anteriores sao acopladas a extremidade de origem da conexao, enquanto acoes adaptativas posteriores se associam a extremidade de destino da conexao.

Na linguagem proposta, quando existir uma chamada de funcao adaptativa, ela sera especi cada ao ser descrita a conexao que a suporta.

Na linguagem proposta, sempre que a execucao de uma acao adaptativa promover a remocao ou adicao de uma conexao, e da responsabilidade do programador cuidar para que seja mantida a conectividade do grafo que representa a logica do programa.

A declaracao das funcoes adaptativas encerra a codi cacao do programa.

5 Exemplo Ilustrativo

Para exempli car o emprego do metodo de desenvolvimento de programas automodi caveis proposto neste artigo, apresenta-se aqui o esboco do desenvolvimento de uma implementacao adaptativa do nucleo de um simulador dirigido por eventos, similar ao apresentado no antigo trabalho de McDougall [7]. A simulacao dirigida por eventos e uma tecnica simples, ao mesmo tempo em que se mostra tecnologicamente importante, em virtude da vasta gama de aplicacoes em que pode ser empregada. A classe de problemas que este nucleo de simulacao resolve e a da reacao de um modelo, quando estimulado por uma sequencia de eventos.

No simulador, tal sequencia funciona como uma especie de programa a ser executado, cujas instrucoes sao os eventos, a serem interpretados pelo simulador.

Essa programacao dos eventos a serem simulados e especi cada na forma de uma estrutura de dados organizada como lista, ordenada segundo os instantes relativos previstos para a ocorrencia temporal dos eventos que a compoem.

Os eventos sao modelados na forma de triplas de informacoes, especi candose para cada qual o tipo de evento, a atividade simulada a qual o evento se refere, e o instante programado para a sua ocorrencia. Eventos podem ser independentes ou dependentes, conforme seja sua ocorrencia especi cada externamente, ou decorrente de acoes provocadas por outros eventos.

Naturalmente, a ocorrencia de eventos dependentes e determinada pelo proprio processo de simulacao a medida que este progride, portanto novos eventos podem ir sendo dinamicamente agregados a lista de eventos, razao porque a lista de eventos pode com naturalidade ser considerada como um script de programa adaptativo, e o simulador, como sendo o seu interpretador.

A simulacao tem in cio quando se fornece ao simulador um conjunto de eventos independentes, ou seja, um conjunto de ocorrencias desejadas, Assim, cada elemento dessa lista representa e especi ca um particular evento a ser simulado, contendo para isso as informacoes mencionadas acerca do evento, que viabilizem sua simulacao (tipo de evento, atividade a que se refere, instante de ocorrencia), servindo o instante de ocorrencia como chave de ordenacao da lista.

E interessante notar que, com pequenas modi cacoes, essa estrutura de dados pode se adequar com facilidade a representacao de uma grande variedade de fenomenos, que podem ser objeto de simulacoes estruturadas de forma similar. Entre tantos exemplos poss veis, e particularmente aderente a este exemplo a simulacao de arquiteturas de interfaces reativas, como por exemplo, as de maquinas de calcular, ou ainda de processadores em geral, guiada por uma programacao apresentada na forma de uma lista de operacoes a realizar (teclas acionadas ou instrucoes especi cadas). Ilustra essa situacao a interpretacao de programas expressos atraves de uma linguagem de script, ou de alguma linguagem de maquina, real ou virtual.

Um importante elemento de uma simulacao dessa natureza e, naturalmente, o tempo, visto que se trata de uma simulacao de fenomenos temporais. A simulacao de ocorrencias temporais raras vezes pode ser efetuada na mesma escala de tempo que os fenomenos simulados, ou seja, para um observador simultaneo do processo

de simulacao e do processo simulado, o tempo f sico gasto pelo simulador para simular um dado fenomeno real pode ser muito diferente (maior ou menor) que o correspondente tempo f sico gasto pelo fenomeno real. Adicionalmente, raras vezes e poss vel criar simuladores cujos intervalos tempo de simulacao sejam sequer proporcionais aos correspondentes intervalos de tempo dos fenomenos simulados. Por essa razao, opta-se por um tempo simulado que ui de forma discreta, nao cont nua, limitando-se aos instantes em que haja alguma atividade em curso, e saltando intervalos inativos, economizando dessa maneira tempo de processamento.

Paralelismo e concorrencia podem ser facilmente simulados em uma arquitetura como esta. Eventos que ocorrem simultaneamente ocupam posicoes diferentes na lista de eventos, apesar de estes estarem ordenados segundo o instante de sua ocorrencia. Assim, a simulacao (sequencial) de eventos que estejam programados para ocorrerem simultaneamente nao sera obrigatoriamente simultanea, mas o instante de tempo simulado nao deve progredir enquanto houver eventos programados para ocorrerem num mesmo instante.

Considerando-se esse fato, e tomando-se apenas alguns cuidados com o tratamento dos inevitaveis con itos que surgem em situacoes dessa natureza, a simulacao de eventos simultaneos podera ser feita sem grandes di culdades, praticamente da mesma forma que a dos demais eventos.

Esboca-se a seguir em pseudo-codigo, a logica do nucleo de um simulador dirigido por eventos de proposito geral, que pode ser convenientemente adequado para a implementacao de aplicacoes como as sugeridas anteriormente.

1. Obter o conjunto de eventos independentes especi cado pelo usuario, e com ele construir a lista de eventos inicial. Ordenar essa lista segundo os instantes de ocorrencia programados para os diversos eventos. Posicionar o relogio simulado (TIME) no instante inicial de simulacao, usualmente, o instante relativo zero. Posicionar o simulador para a extracao do primeiro evento da lista.
2. Preparar o simulador para extrair, identi car e interpretar o proximo evento da lista.
3. Nao restando eventos na lista, emitir relatorio dos resultados coletados durante a simulacao, e terminar o processamento. Caso contrario, prosseguir.
4. Retirar o primeiro evento (e, t, j) da lista de eventos, cujo instante de ocorrencia (t) e o mais baixo dentre os eventos programados.
5. Registrar o instante corrente de simulacao para este evento: a) caso o relogio simulado TIME acuse um instante futuro em relacao ao instante (t) previsto para a ocorrencia do evento, o instante de simulacao daquele evento sera aquele indicado pelo relogio (TIME), o qual se mantem inalterado; b) caso contrario, TIME devera ser adiantado para coincidir com o instante (t) previsto para a ocorrencia do evento.
6. Determinar o tipo (e) do evento extra do e a identi cacao do fenomeno simulado ao qual o evento se refere (j).
7. De acordo com o tipo (e) do evento, e preciso acionar uma rotina de interpretacao apropriada. A logica espec ca dessa rotina pode ser criada adaptativamente, neste ponto, para ser executada logo a seguir, em substituicao

a um eventual outro bloco que estiver presente abaixo, no item 8. (Em casos muito simples, essa logica poderia ter sido programada antecipadamente, de forma estatica.)

8. Bloco de interpretacao do evento corrente, criado dinamicamente.
9. Registrar os resultados da simulacao do evento, para o relatorio nal de simulacao; Inserir, na lista de eventos, de acordo com as consequencias programadas para o particular evento simulado, os eventos dependentes adicionais que forem necessarios, programando-os para que seu (futuro) instante de ocorrencia esteja em conformidade com as caracter sticas do fenomeno simulado. [E aqui que a simulacao pode se utilizar dos recursos dinamicos dos programas adaptativos].
10. Voltar ao passo 2, para a extracao do proximo evento a simular.

Diagrama Identi cam-se facilmente os seguintes pontos notaveis do programa acima:

a) Blocos basicos: correspondentes aos itens 1, 2, 3, 4, 5a, 5b, 6, cada um dos sub-itens do item 7, referentes aos diversos tipos poss veis de eventos, e a juncao dos itens 9 e 10 (por serem puramente sequenciais) do algoritmo acima;

b) Blocos dinamicos: bloco 8, que e substitu do pela acao dos sub-itens do bloco 7 a cada vez que o simulador os executa.

c) Blocos que efetuam alteracoes no programa: cada um dos sub-itens do bloco 7, responsaveis pela alteracao dinamica do bloco 8. Assim, ca evidente quais serao os blocos basicos de que o programa se constitui. As acoes adaptativas estarao todas a cargo dos componentes do bloco 7, e portanto as conexoes entre todos os blocos tambem cam bem de nidas. O diagrama resultante, equivalente ao modelo em camadas e mais compacto, e o seguinte:

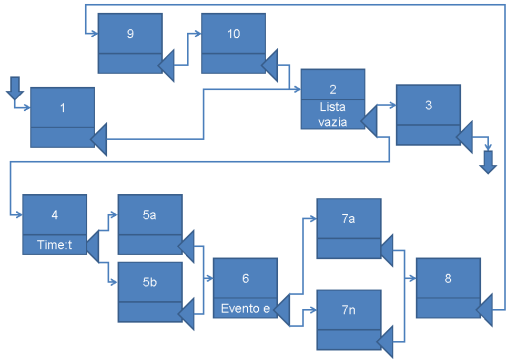

Figura 2. Diagrama do simulador de eventos.

A codi cacao deste diagrama na linguagem proposta ca com o seguinte aspecto:

```
ADAPTIVE MAIN [ NAME = simulador,
        ENTRY = bloco01, EXIT = bloco03 ] IS

CODE bloco01 : < /* inicio de simulação*/
ler conjunto de eventos independentes; construir lista
inicial de eventos;
ordená-la pelo instante de ocorrência; TIME := 0; >;

CODE bloco02 : < /* inicio do loop de simulação */
apontar primeiro evento da lista; if número de elementos
> 0 then bloco2:=1 else bloco2:=0; >;

CODE bloco03 : < /* fim de simulação */
imprimir relatório de resultados de simulação; >;

CODE bloco04 : < /* extração do evento a simular */
obter (e, t, j) na primeira posição da lista de eventos;
remover o primeiro elemento da lista;
if TIME <= t then bloco4:=1 else bloco4:=0; >;

CODE bloco05a : < /* TIME <= t : situação normal */
TIME := t; instante de simulação do evento := t; >;

CODE bloco05b : < /* TIME > t : overhead atrasou
simulação do evento */
instante de simulação do evento := TIME; >;

CODE bloco06 : < /* registro dos fatos da simulação do
evento corrente */
memorizar instante de simulação do evento para o
relatório de simulação; bloco6 :

CODE bloco07 : < /* bloco de isolação, evita alterar
muitas conexões sem necessidade */ >;

CODE bloco08 : < /* este bloco é substituído sempre que
se passa do bloco06 para o bloco08 */ >;

CODE bloco09 : < /* guardar para o relatório de simulação
os efeitos da execução do bloco08 */ >;

CODE bloco10 : < /* se necessário, acrescentar eventos
dependentes adequados à lista
de eventos, antes de reiniciar o loop de simulação */ >;

CONNECTION FROM bloco01 (CASE 0: TO bloco02);
CONNECTION FROM bloco02 (CASE 1: TO bloco04,
                         CASE 0: TO bloco03);
CONNECTION FROM bloco04 (CASE 1: TO bloco05a,
                         CASE 0: TO bloco05b);
CONNECTION FROM bloco05a (CASE 0: TO bloco06);
CONNECTION FROM bloco05b (CASE 0: TO bloco06);
CONNECTION FROM bloco06 (CASE N: T0 bloco07
          PERFORM montacodigo(N) BEFORE,
          ... ;
          CASE 2: T0 bloco07
          PERFORM montacodigo(2) BEFORE,
          CASE 1: T0 bloco07
          PERFORM montacodigo(1) BEFORE}
CONNECTION FROM bloco07 (CASE 0: TO bloco08);
CONNECTION FROM bloco08 (CASE 0: TO bloco09);
CONNECTION FROM bloco09 (CASE 0: TO bloco10);
CONNECTION FROM bloco10 (CASE 0: TO bloco02);

ADAPTIVE FUNCTION montacodigo (i) :          {
REMOVE [ CONNECTION 0 FROM bloco07 ];
REMOVE [ CONNECTION 0 FROM bloco08 ];
REMOVE [ CODE bloco08 ];
INSERT [ CODE bloco08 : < novo código a ser usado para
executar o bloco08 >;];
INSERT [ CONNECTION 0 FROM bloco08 TO bloco09];
INSERT [ CONNECTION 0 FROM bloco07 TO bloco08];  }

END MAIN
```

Figura 3. Programa adaptativo do simulador de eventos.

6 Conclusoes

Este artigo oferece ao leitor um panorama sobre o atual estado do desenvolvimento de tecnologia voltada para a programacao adaptativa, apresentando detalhadamente, para essa classe de programas, um metodo de projeto e desenvolvimento apoiado em uma arquitetura geral, especialmente projetada para a representacao de programas adaptativos.

Nessa arquitetura, representa-se a logica de um programa automodi cavel decompondo-o em quatro camadas mutuamente ortogonais, cuja utilizacao privilegia que sejam evidenciados os quatro tipos fundamentais de elementos de que se constituem os programas adaptativos: (i) blocos basicos de codigo estatico; (ii) decisores; (iii) acoes adaptativas; (iv) conectores.

A linguagem aqui empregada para programacao adaptativa re ete essa arquitetura, e permite que programas adaptativos, concebidos segundo suas diretrizes, possam ser diretamente codi cados em um n vel aderente de abstracao conceitual.

Para que se possa desenvolver, como solucao adaptativa de um problema, um modelo que siga esse padrao, e necessario planejar o uso da adaptatividade nessa solucao, aplicando tecnicas projetadas e escolhidas caso a caso, ou entao partir de alguma solucao tradicional, convertendo-a para que seja representada adaptativamente atraves de alguma forma automodi cavel equivalente.

No presente trabalho, esta ultima forma foi tratada com maior cuidado, complementando outras publicacoes anteriores, cuja tonica estava centrada em solucoes ad-hoc, aplicaveis em uma variedade de casos espec cos entao estudados.

Referencias

1. Anckaert, B; Madou, M; Bosschere, K. D. A Model for Self-Modifyind Code. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, ISSN 0302-9743, Volume 4437/2007, Information Hiding, ISBN 978-3-540-74123-7, Pages 232-248, September 14, (2007)
2. Madou, M. et al. Software Protection Through Dynamic Code Mutation. Wisa 2005. LNCS 3786, pag. 194-206 (2006)
3. Gi n, J. T., Christodorescu, M., Kruger, L. Strenghtening software self-checksumming via self-modifying code. Proceedings of the 21st Annual Computer Security Applications Conference ACSAC (2005)
4. Laboratório de Linguagens e Técnicas Adaptativas LTA, `http://www.pcs.usp.br/~lta`
5. Neto, J.J. Adaptive Rule-Driven Devices - General Formulation and Case Study. Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001, Vol. 2494, Pretoria, South Africa, July 23-25, Springer-Verlag, 2001, pp. 234-250 (2001)
6. Neto, J.J., Silva, S.R.B. Projeto de uma linguagem para programaçao adaptativa. Submetido ao IX Congreso de la Sociedad Peruana de Computación (2010)
7. MacDougall, M.H. ACM Computing Surveys (CSUR) . Volume 2 , Issue 3 (September 1970), pages: 191 209 ISSN:0360-0300 (1970)