Uma experiência no Desenvolvimento de Linguagens Adaptativas de Programação

Aparecido Valdemir de Freitas
USCS - Universidade Municipal de São Caetano do Sul
Av. Goiás N° 3400 – Vila Barcelona – São Caetano do Sul – CEP 09550-051
São Paulo – Brasil - avfreitas@uscs.edu.br

e

João José Neto
Escola Politécnica da Universidade de São Paulo
Depto. de Engenharia de Computação e Sistemas Digitais
Av. Prof. Luciano Gualberto, trav. 3, N° 158 - Cidade Universitária
São Paulo – Brasil - joao.jose@poli.usp.br

Abstract

Adaptive devices comprehend a subjacent (usually non-adaptive) device, e.g. an automaton, a grammar, etc., to which an adaptive mechanism is added which performs the self-modification operations usual in adaptive devices. Adaptive languages are special adaptive devices whose subjacent formalism is a programming language. This work defines adaptive programming languages, describes their main concepts and discusses their particular development issues. Adaptive programming languages suggest a new programming style, since applying adaptive technology induces a somewhat new way to think about programs. Such adaptive style may become an alternate way to obtain adequate code in self-modifying applications.

Keywords: Adaptive devices, self-modifying devices, adaptive programming language.

Resumo

Um dispositivo adaptativo é constituído por um dispositivo subjacente (geralmente não adaptativo), por exemplo, um autômato, uma gramática, etc., no qual se adiciona um mecanismo adaptativo que é responsável pela automodificação autônoma que caracteriza os dispositivos adaptativos. As linguagens adaptativas são instâncias de dispositivos adaptativos, cujo formalismo subjacente é uma linguagem de programação. O artigo tem como objetivo conceituar linguagens adaptativas de programação, descrever seus pontos essenciais e considerar os aspectos e particularidades que afetem seu desenvolvimento. A concepção de linguagens adaptativas motiva um novo estilo de programação, uma vez que a aplicação da tecnologia adaptativa sugere uma nova forma de pensar. O estilo adaptativo de programação poderá tornar-se uma alternativa viável para se obter códigos aderentes às aplicações de códigos automodificáveis.

Palavras chaves: Dispositivos adaptativos, dispositivos auto-modificáveis, linguagem de programação adaptativa.

1 INTRODUÇÃO

A adaptatividade considera o uso de dispositivos que apresentam a característica de se modificarem dinamicamente em resposta a estímulos de entrada, sem interferência de agentes externos. Estas propriedades de automodificação, características dos dispositivos adaptativos, lhes dão naturalidade para expressarem e manusearem diversos aspectos de sistemas automodificáveis. Dispositivos adaptativos devem, portanto, ser capazes de detectar possíveis necessidades de modificação de comportamento para, em seguida, reagirem a elas executando as correspondentes modificações [1].

A idéia principal que permeia a adaptatividade reside na habilidade que um dispositivo tem de executar operações de automodificação, denominadas ações adaptativas, as quais podem ser vistas como atividades internas ao dispositivo e que são executadas em resposta ou reação à detecção das situações que exigem alterações comportamentais.

Historicamente, dispositivos adaptativos foram inicialmente estudados e aplicados na área de construção de compiladores para a obtenção de mecanismos puramente sintáticos destinados a ampliar a capacidade de expressão dos autômatos de pilha estruturados [2].

Um dispositivo adaptativo é constituído por um dispositivo subjacente (geralmente não adaptativo), por exemplo, um autômato, uma gramática, uma árvore de decisão, etc, ao qual se acrescenta um mecanismo adaptativo, responsável por permitir que a estrutura do dispositivo subjacente seja dinamicamente modificada. Por exemplo, ao se acrescentar um mecanismo adaptativo a um autômato de estados finitos, este adquire a possibilidade de efetuar inclusões ou remoções de transições de estados, durante o processamento da cadeia de entrada, o que aumenta seu poder de expressão.

A adaptatividade tem, portanto, como aplicação básica estender formalismos consolidados, aumentando seu poder de expressão [3].

Posteriormente aos autômatos, outros dispositivos adaptativos foram concebidos a partir de diferentes dispositivos subjacentes, como por exemplo, statecharts [4] [5], redes de Markov [6], gramáticas [7], tabelas de decisão [1], árvores de decisão [3], etc. Aplicações surgiram em diversas áreas, tais como: aprendizagem computacional [8], processamento de linguagens naturais [9], ambientes multilinguagens [10], robótica [11], etc.

Todas essas contribuições incorporaram à adaptatividade um conjunto de poderosos recursos, de natureza não apenas conceitual e teórica, mas também de caráter aplicativo, na forma de métodos, técnicas e ferramentas, que foram sendo desenvolvidos ao longo dessa história [12].

2 MOTIVAÇÃO E OBJETIVOS

Anteriores ao surgimento de trabalhos explicitamente relacionados à adaptatividade [12], pode-se considerar que a extensibilidade de linguagens de programação seja uma das mais importantes manifestações do fenômeno da adaptatividade em linguagens de programação.

O poder de atuação da extensibilidade se limita, entretanto, às componentes da linguagem de programação que podem ser tratadas em tempo de compilação, e se refere, portanto, às alterações que o programador pode efetuar sobre a linguagem, de forma que possa adequá-la aos particulares propósitos dos seus programas. Entretanto, isso ainda não atende aqueles usuários que necessitem que seus programas possam exprimir atividades, a serem executadas em tempo de execução, de alteração na estrutura de seus próprios algoritmos.

Para isso, tais programas necessitariam incorporar recursos de adaptatividade e, portanto, deveriam dispor de meios para, sem intervenção externa, determinar e realizar dinamicamente as automodificações estruturais necessárias.

Linguagens adaptativas podem ser vistas como dispositivos adaptativos, cujo formalismo subjacente é uma linguagem de programação. Sendo possível nessas linguagens a aplicação dinâmica de ações adaptativas, torna-se natural expressar, com relativa facilidade, fenômenos de automodificação nos programas nelas codificados.

A principal motivação deste artigo está associada ao levantamento dos requisitos para o desenvolvimento de uma linguagem de programação que atenda a tais requisitos, e para isso pode valer-se das propriedades dos formalismos adaptativos, que lhes são aderentes, como conseqüência.

Programas automodificáveis codificados usando linguagens adaptativas tendem a ser mais robustos, visto que já na época da desses programas, é possível fazer algumas verificações com a finalidade de garantir que as automodificações efetuadas sejam relativamente seguras, evitando problemas decorrentes de uma excessiva liberdade nas operações de automodificação presentes nos programas.

Este artigo tem como objetivo geral efetuar um levantamento de requisitos, em relação a diversos aspectos de projeto e implementação de linguagens adaptativas de programação.

3 DISPOSITIVOS GUIADOS POR REGRAS

O conceito da adaptatividade [12] pode ser aplicado a qualquer dispositivo cuja operação seja definida por um conjunto de regras. Na construção de tais dispositivos, conceitualmente é possível separar de forma clara dois importantes componentes: um dispositivo subjacente, tipicamente não-adaptativo, e um mecanismo adaptativo, responsável pela incorporação da adaptatividade.

Nessa formulação, a característica principal de um dispositivo adaptativo é sua capacidade de realizar ações adaptativas, que se traduzem na execução de operações de automodificação ao serem identificadas determinadas situações específicas [1].

Da aplicação da adaptatividade a um sistema, torna-se possível que, em função dos estímulos recebidos, instâncias idênticas de uma mesma configuração evoluam para comportamentos finais totalmente diferentes, uma vez que esta capacidade de automodificação está associada à diversidade de eventos a que o sistema adaptativo é submetido durante sua respectiva utilização.

Partindo de uma configuração inicial, estando em uma dada configuração, e recebendo um determinado estímulo, o dispositivo, através da aplicação de alguma das regras que definem seu comportamento, assume uma nova configuração, de acordo com a regra adequada àquela situação.

O dispositivo é dito determinístico se e somente se, para qualquer configuração dada e qualquer estímulo de entrada, seu conjunto de regras determina uma, e somente uma, próxima configuração. No caso não-determinístico, havendo mais de uma regra aderente à situação corrente do dispositivo, as diversas possíveis alternativas são tratadas em paralelo [13].

4 LINGUAGENS ADAPTATIVAS

Dispositivos adaptativos guiados por regras iniciam a operação em alguma configuração inicial pré-estabelecida e evoluem sucessivamente para novas configurações até que seja atingida alguma configuração de parada (aceitação ou rejeição). Caso o dispositivo nunca atinja tal situação, ocorre algo similar ao problema da parada da máquina de Turing, ou seja, o dispositivo entra em “loop” e não consegue decidir a situação.

Linguagens de programação podem ser vistas como um caso particular de dispositivos dirigidos por regras, em que as regras correspondem às diversas estruturas que representam a configuração do programa. Assim, um programa expresso em linguagem adaptativa assume sucessivas configurações CF1, CF2, ..., CFn, de tal modo que, a partir da configuração inicial de código CF1 e

por meio de chamadas de ações adaptativas, o código do programa evolua para sucessivas configurações CF2, CF3,.., CFn, à medida em que a execução for sendo progredindo [15].

Seguindo o mesmo esquema de formalização empregado para os dispositivos adaptativos guiados por regras, define-se a seguir as linguagens de programação adaptativas.

Não será feita aqui uma formalização completa, explícita, das linguagens adaptativas consideradas. Entretanto, são apresentados os pontos principais dos quais podem resultar formalizações adequadas:

a) Dispositivo Subjacente: é a linguagem (não-adaptativa) que servirá de suporte para as ações adaptativas responsáveis por sua automodificação. O conjunto de regras que o definem têm o formato e o significado descritos a seguir.

b) Camada Adaptativa: Compreende a definição de um conjunto de funções adaptativas, como combinações das funções elementares (? , + , -) clássicas [14], e o acoplamento de suas chamadas parametrizadas às regras básicas do dispositivo subjacente.

c) Programa Adaptativo: Descrito na forma de um conjunto de regras adaptativas, representa o código cuja execução efetua o trabalho desejado.

Neste artigo, considera-se que o dispositivo subjacente é um programa escrito na linguagem Lisp, que pode ser formalizado [1] da seguinte maneira:

```
<dispositivo>              →     <programa-Lisp-completo>
<programa-Lisp-completo>   →     <lista-de-um-nível>
<lista-de-um-nível>        →     ( identificação-do-nó  <seqüência> )
<seqüência>                →     ε
                           →     <lista-de-um-nível>
                           →     <seqüência>  <lista-de-um-nível>
```

Dado um programa Lisp com tal estrutura, pode-se representá-lo como uma seqüência de regras:

1ª regra: Lista de um nível que representa o programa completo (representando toda a árvore).

Outras Regras: Listas de um nível que compõem a seqüência que define a lista de um nível de hierarquia imediatamente superior (representando alguma sub-árvore do programa).

Programas adaptativos exigem automodificação em tempo de execução. Linguagens adaptativas devem fornecer meios para que o programador expresse tais automodificações em seus programas. É possível construir-se linguagens que realizem estas operações de forma nativa, e com sintaxe própria, por meio de bibliotecas de operações adaptativas em um ambiente de execução para fornecer o suporte ao compilador.

Alternativamente, é possível partir-se de linguagens existentes, estendendo-as para se incorporar adaptatividade. Há duas formas de se tratar a extensibilidade: extensão funcional e sintática. A extensibilidade funcional corresponde à criação de uma biblioteca própria para tratamento de adaptatividade, enquanto que a sintática permite se obter uma linguagem parecida com as genuinamente adaptativas, a partir da extensão sintática de uma linguagem extensível.

Uma linguagem adaptativa baseada em linguagem funcionalmente extensível deve disponibilizar operadores (funções) não-adaptativos e adaptativos (expressos na linguagem subjacente). Os programas dela oriundos terão comportamento automodificável em tempo de execução e o processamento dos mesmos será efetuado com o auxílio do interpretador da linguagem subjacente.

Tendo em vista que as funções adaptativas devem expressar todas as possibilidades de automodificação do programa, torna-se conveniente disponibilizar algum recurso que permita ao programador referenciar, univocamente, as porções de código que serão objeto de tal

automodificação. É natural que, especialmente no caso de linguagens funcionais, a introdução de tais mecanismos de referenciação afaste o programa do paradigma funcional. Entretanto, sua utilização pode simplificar os procedimentos necessários à implementação da camada adaptativa, razão pela qual foi mantido neste projeto.

Preferencialmente, a linguagem subjacente deve oferecer meios para o programador fazer essa referenciação. Caso a linguagem subjacente não ofereça formas nativas para tal referenciação, torna-se necessário que algum recurso extra seja disponibilizado ao usuário, por exemplo, na forma de algum tipo de extensão, adicionado à linguagem subjacente.

No caso particular da linguagem Lisp, os programas podem ser representados como árvores. Dessa forma, sua automodificação pode ser implementada através da alteração estrutural da árvore que o representa.

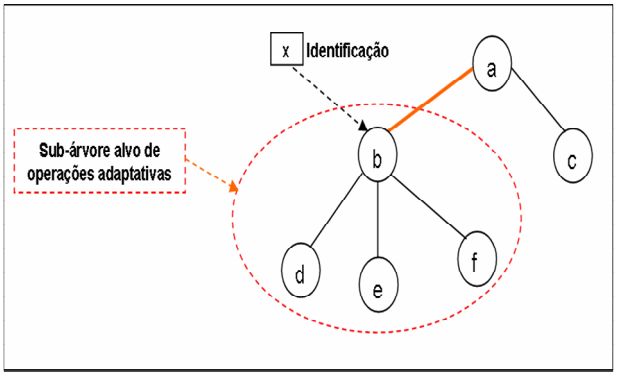

Figura 1 – Identificação de nós na árvore que representa o programa

Conforme esboçado na Figura 1, a identificação da região a ser modificada corresponde a um nome simbólico associado à raiz da sub-árvore que a representa. Considerando que, neste trabalho, não há funções nativas que enderecem os nós internos da árvore que representa o programa, torna-se necessário que se disponibilize, na camada adaptativa alguma operação que permita ao usuário referenciar uma sub-árvore.

Assim, propõe-se como primitiva da camada adaptativa, uma função, denominada *mark*, que terá como parâmetros um nome (que define um identificador para ser usado como referência) e uma porção de código (que define a sub-árvore que será alvo da adaptatividade).

Para ilustrar a operação desta primitiva, considere-se o programa, esboçado na forma de árvore, na Figura 1. Na Figura 2, mostra-se a árvore que representa o programa, devidamente marcada pela primitiva *mark*, com a identificação 'x'. Estabelecida a forma de vinculação de identificações às porções correspondentes de código automodificável do programa, podem-se construir funções adaptativas que se responsabilizem pelas operações de automodificação do código do programa adaptativo.

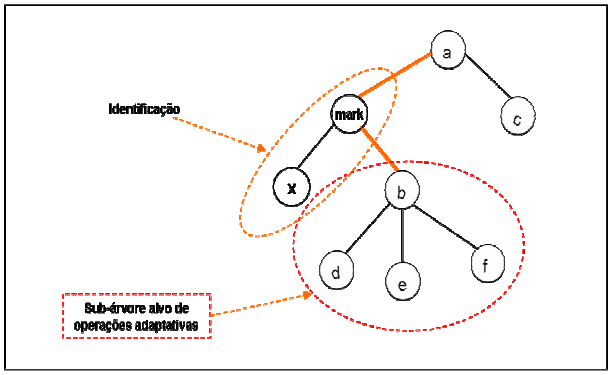

Figura 2– O mesmo programa da Figura 1, em sua representação de árvore, com nó identificado pela primitiva mark.

Optando-se por não utilizar identificações explícitas, as funções adaptativas deverão ativar funções equivalentes da biblioteca do Lisp para localizar e manipular as sub-estruturas desejadas.

5 FUNÇÕES ADAPTATIVAS

Considerando-se que a linguagem-base do nosso dispositivo adaptativo é formada por um conjunto de expressões, seja **Exp** o domínio sintático da linguagem subjacente, no qual:

Exp = { E | E é uma expressão}.

As funções adaptativas são aplicadas em **Exp** e retornam também elementos pertencentes a **Exp**, ou seja:

$$\mathcal{F} : \mathbf{Exp} \rightarrow \mathbf{Exp}$$

Uma vez que uma função adaptativa $\mathbf{a} \in \mathcal{F}$ produz valores pertencentes a **Exp**, suas denotações representam membros de **Exp**.

Para modelarmos esta idéia, considera-se a função semântica Φ, o tipo **Exp** → **Exp** e $\mathbb{E}(\Phi)$ representando os resultados da aplicação da função $\mathcal{F}$ sobre **Exp**. Assim:

$\mathbb{E}(\Phi)$ = **x** se **x** é um elemento de **Exp**.

$\mathbb{E}(\Phi)$ = erro se a avaliação de Φ causar um erro.

As funções adaptativas, aplicadas a um nó específico da árvore que representa o programa, são codificadas a partir da combinação de um conjunto de ações elementares de alteração, inclusão e exclusão de nós, que, em tempo de execução, afetarão a forma da árvore que representa o programa.

Por exemplo, uma função qualquer $\mathcal{F}$ pode ativar funções primitivas de alteração, inclusão e remoção de sub-árvores.

Essas primitivas de alteração, inclusão e remoção de porções de código devem ser disponibilizadas pela camada adaptativa para serem utilizadas na programação das funções adaptativas.

Durante a aplicação da função adaptativa $\mathcal{F}$ mencionada acima, poderá ocorrer, por exemplo, a remoção de uma sub-árvore, em um ponto especificado da árvore que representa o programa, conforme esboçado na Figura 3.

Após a execução de uma ação de remoção especificada na função adaptativa $\mathcal{F}$ mencionada anteriormente, a árvore que representa o programa terá a forma esboçada na Figura 4.

No caso em questão, está se admitindo que a ação de remoção é aplicada na sub-árvore apontada pelo endereçamento definido pela função *mark*.

Dessa forma, o endereçamento continua a existir na estrutura do programa e a sub-árvore alvo de adaptatividade é substituída por um nó representando ***nil***.

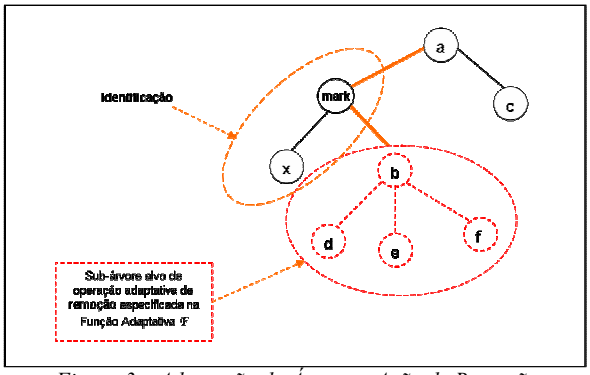

Figura 3 – Adaptação da Árvore – Ação de Remoção

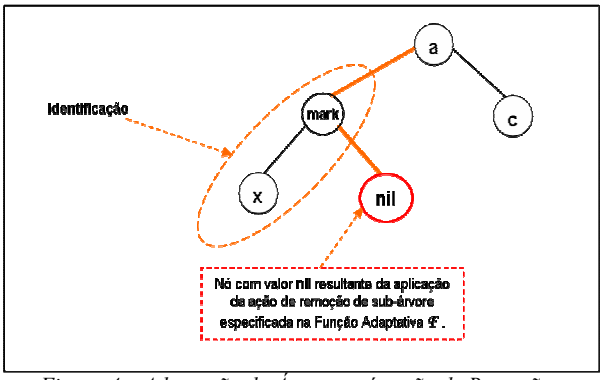

Figura 4 – Adaptação da Árvore após ação de Remoção

Para que o programa, obtido a partir da linguagem adaptativa proposta nesse artigo, possa ser convenientemente processado, necessita-se de um ambiente de execução adequado, que é composto

pelo interpretador da linguagem subjacente e pela camada adaptativa, que terá como responsabilidade avaliar chamadas adaptativas e administrar a execução do programa.

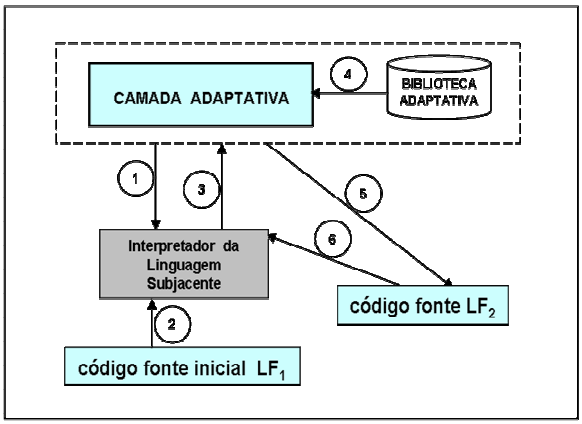

Figura 5 – Operação da Linguagem Adaptativa

Conforme a Figura 5, os seguintes eventos ocorrem durante o processamento de programas escritos na linguagem adaptativa:

1. A camada adaptativa efetua a chamada do interpretador da linguagem subjacente, passando-lhe o código-fonte inicial fornecido pelo usuário.
2. O interpretador da linguagem subjacente inicia a avaliação das funções de forma usual, até que ocorra alguma chamada de função adaptativa. Caso não haja chamadas adaptativas, o avaliador atuará tal qual um interpretador não-adaptativo.
3. O controle é retornado para a camada adaptativa, que irá providenciar o tratamento da chamada adaptativa.
4. Considerando-se que as ações adaptativas são compostas por ações elementares, a camada adaptativa utilizará funções da biblioteca adaptativa.
5. Como resultado da execução das ações adaptativas, um novo código-fonte é gerado.
6. Este novo código-fonte é passado ao interpretador da linguagem subjacente, que se encarregará da continuidade da execução do programa adaptativo.

O programa adaptativo será processado em uma camada adaptativa, especificamente preparada para comportar operações adaptativas, a qual terá como responsabilidade dar todo o suporte necessário para que a adaptatividade seja devidamente validada no programa, em tempo de execução.

Para ilustrar o emprego das primitivas da camada adaptativa apresenta-se um simples programa que apresenta uma decisão que se baseia na chamada de uma função adaptativa. No exemplo, caso os parâmetros passados ao programa sejam diferentes, o programa executa uma função adaptativa que, em tempo de execução, acrescenta um código alternativo para a decisão. O exemplo mostra assim a situação de um "*if adaptativo*". A Figura 6 apresenta o programa em *Lisp* e a Figura 7 o código da função adaptativa.

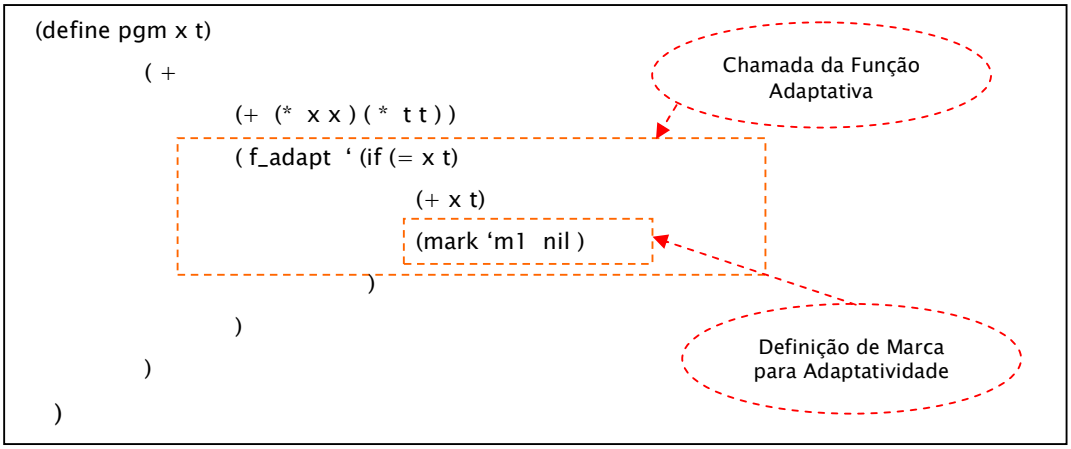

Figura 6 – Exemplo de um programa adaptativo

```
(define f_adapt  arvore)
                    ( upd  arvore 'm1  '(+ 2 x)
                    ( eval arvore)
  )
```

Figura 7 – Exemplo de uma função adaptativa com ação de update

6 DIRETRIZES PARA DESENVOLVIMENTO DE LINGUAGENS ADAPTATIVAS

Em caso de a linguagem adaptativa ter como base uma linguagem não-adaptativa e um mecanismo de extensão, pode-se citar como requisito de funcionalidade que a linguagem adaptativa deve ter condições de efetuar todas as operações presentes na correspondente linguagem-base não-adaptativa. Essa característica é obtida por meio de algum mecanismo de extensão presente na linguagem-base.

Outro requisito de funcionalidade de uma linguagem adaptativa, nativa ou baseada em linguagem não-adaptativa, é a existência de algum recurso que permita, de forma dinâmica, referenciar e modificar partes do próprio programa em execução.

Neste artigo, considerou-se a forma de endereçamento, utilizada para se referenciar e modificar partes do programa, por meio do desenvolvimento e disponibilização de uma primitiva de rotulação presente na camada adaptativa.

No entanto, em uma linguagem genuinamente adaptativa tal recurso deveria estar incorporado na sintaxe e no ambiente de execução da linguagem, sem a necessidade, portanto, de extensões ou de chamadas de biblioteca para tal propósito.

Como uma característica estrutural desejável, uma linguagem de programação adaptativa, baseada ou não na extensão de uma linguagem não-adaptativa, deve estar associada a um ambiente de execução que dê ao compilador a possibilidade de referenciar no código objeto gerado, as operações adaptativas presentes na biblioteca ou no ambiente de execução. No primeiro caso, o ambiente deve ser visível ao programador enquanto que no segundo, deverá, de preferência, ser implícito e transparente ao programador.

Dentre os diversos recursos que podem fazer parte do ambiente de desenvolvimento e de execução para uma linguagem adaptativa, pode-se citar: recursos de edição, manuseio de arquivos, operações de rastreamento, operações de depuração, auxílio à escrita de programas adaptativos, acesso à camada adaptativa, monitoração do estado do programa (em caso de linguagens em estilo imperativo, incluir variáveis e valores). Este ambiente deve, preferencialmente, ter uma interface gráfica, para melhor interação com o usuário.

O mecanismo de extensão presente na linguagem-base permite ao programador que esta seja estendida com a inserção de construtos adaptativos. Assim, com o auxílio de uma biblioteca de operações adaptativas, de um ambiente de execução e de recursos de extensibilidade, pode-se ensaiar construtos adequados para a implementação de uma linguagem genuinamente adaptativa. Este assunto deverá ser alvo de pesquisas futuras, estando fora do escopo deste artigo.

A seguir descrevem-se algumas recomendações adicionais que se tornam desejáveis para a implementação de linguagens adaptativas, no tocante às características sintáticas e que serão objeto de futuras investigações.

Do ponto de vista sintático, no caso de a linguagem adaptativa ser baseada em uma linguagem não-adaptativa, todas as extensões sintáticas devem ser projetadas de forma aderente ao estilo sintático da linguagem-base. Tal recomendação trará ao programador maior facilidade no emprego da

linguagem adaptativa, uma vez que a manutenção do estilo sintático para os recursos de adaptatividade pode proporcionar um aprendizado mais natural dos novos construtos.

Para que esta característica seja consolidada, recomenda-se também que o projeto da linguagem adaptativa mantenha uniforme a granularidadade da linguagem-base, de forma que não sejam expostos detalhes da linguagem estendida que sejam próprios de níveis de abstração inferiores aos da linguagem-base, nem tampouco do seu ambiente de execução, os quais devem se manter, preferenciamente, encapsulados na biblioteca da linguagem ou então totalmente transparentes ao usuário.

Uma questão, que deve ser considerada no desenvolvimento de uma linguagem adaptativa, se refere à forma pela qual se definem as partes do programa que são alvo da adaptatividade. Essas partes do programa compreendem regiões contíguas que delimitam o escopo das alterações adaptativas.

Alternativamente, é necessário que se defina, também através de referências, o ponto exato do programa onde será processada alguma operação adaptativa. Na sintaxe de uma linguagem genuinamente adaptativa recomenda-se que sejam definidos os construtos próprios para essas formas de referência, bem como, no mínimo, para operações adaptativas básicas de inserção e remoção de partes do programa.

Embora não seja regra, a forma adotada neste trabalho para estabelecer pontos referenciáveis se dá através do emprego de rótulos. Para delimitar regiões contíguas referenciáveis pode-se empregar o mesmo mecanismo de rótulos utilizado para blocos de instruções do programa. Alternativamente, mecanismos similares à delimitação de regiões (escopos) utilizados em outras linguagens permitem obter efeitos semelhantes sem o emprego de nomes.

Outra questão aderente ao mecanismo de endereçamento de porções do programa se refere à possibilidade de o programador, ao projetar suas funções adaptativas, definir meta-rótulos, ou seja, dar um nome a uma família de rótulos a ser instanciada diversas vezes, sempre com nomes diferentes, permitindo assim que as referências aos componentes instanciados possam ser conjuntamente especificadas, de maneira uniforme.

Assim, quando o programador, numa função adaptativa, define um rótulo fixo, estabelece uma referência direta e explícita a um ponto referenciável do programa. No entanto, com o emprego de parâmetros, variáveis e geradores (no estilo clássico da teoria dos dispositivos adaptativos), as referências se tornam instanciáveis.

As Figuras 8 e 9 apresentam uma sumarização das diretrizes apontadas neste artigo, para auxiliar no projeto e implementação das linguagens adaptativas de programação.

- Linguagem Adaptativa (LPA) = Linguagem Não-Adaptativa (LNA) + Extensibilidade
- LPA é superset de LNA
- LPA – LNA é obtido por meio de Extensibilidade
- Mecanismo Adaptativo introduzido por extensão e por bibliotecas
- Ambiente de Execução visível ao programador
- Necessita de mecanismo de referência para modificação dinâmica
- Primitivas de Identificação e de Modificação de Código
- Estilo sintático aderente à Linguagem-Base (LNA)
- Facilidade aprendizagem

Figura 8 – Diretrizes para Desenvolvimento de Linguagens Adaptativas obtidas por Extensibilidade

- Necessita de mecanismo de referência para modificação dinâmica
- Primitivas de Rotulação e de Modificação de Código
- Estilo sintático aderente à Linguagem-Base
- Facilidade de aprendizagem
- Ambiente de Execução implícito ao programador
- Ambiente de Desenvolvimento c/edição, *trace*, *debug*, auxílio à escrita de programas, acesso à camada adaptativa, monitoração de estado, interface gráfica.
- Encapsulamento do ambiente de execução
- O projeto da linguagem adaptativa deve manter uniforme a granularidade da Linguagem-Base
- Construtos próprios para operações de adaptatividade
- Definição de meta-rótulos
- Definição do Escopo de Adaptatividade

Figura 9 – Diretrizes para Desenvolvimento de Linguagens Nativamente Adaptativas

7 CONSIDERAÇÕES FINAIS

Dentre as contribuições deste artigo, pode-se destacar a conceituação de linguagens adaptativas de programação e consideração das diversas características que uma linguagem de programação necessita ter para ser considerada uma linguagem adaptativa.

A implementação usada neste trabalho considerou uma linguagem funcional de programação à qual se acrescentou uma camada adaptativa responsável pela automodificação de código.

Para se exercitar a linguagem adaptativa desenvolvida neste artigo, desenvolveu-se uma aplicação, com características adaptativas, na qual foram ensaiados e validados os diversos recursos presentes na camada adaptativa da linguagem.

Esses recursos se limitam, entretanto, aos componentes de linguagem que são tratados em tempo de compilação e correspondem, portanto, às alterações que o usuário pode fazer sobre a linguagem, de modo a torná-la mais adequada a seus particulares objetivos.

A partir da disponibilização da linguagem adaptativa desenvolvida neste trabalho estão sendo contempladas as necessidades daqueles usuários que necessitem que seus programas possam dinamicamente (ou seja, em tempo de processamento) efetuar alterações no fluxo de controle de execução.

Com base nas experimentações e estudos desenvolvidos neste trabalho, pode-se também citar como contribuição a apresentação de um conjunto de diretrizes que permitem definir os requisitos de uma linguagem de programação adaptativa.

Ainda como contribuição pode-se citar, como decorrência da disponibilização das linguagens adaptativas de programação, a possibilidade do emprego destas em inúmeras áreas emergentes de aplicações de códigos automodificáveis tais como, proteção de software, ofuscação de código, engenharia reversa, etc.

Durante o desenvolvimento da aplicação com a linguagem adaptativa desenvolvida neste artigo, constatou-se que ao se empregar tal linguagem, uma forma própria de pensar ou de se raciocinar deve ser exercitada, de maneira natural e espontânea, como conseqüência das ações adaptativas presentes nos programas dela gerados.

Isso exige uma forma própria de raciocínio ao programar, uma vez que o comportamento do programa em execução está diretamente relacionado à prévia execução de ações adaptativas, cuja

operação define o comportamento dinâmico do programa. Em outras palavras, exige uma disciplina de programação que antecipe os efeitos das ações adaptativas sobre o comportamento do dispositivo.

Como trabalho futuro, espera-se que seja desenvolvido um método que oriente o programador a bem utilizar a adaptatividade na sua programação. Neste trabalho, como conseqüência dos experimentos realizados, observou-se que um primeiro passo na escrita de código adaptativo é a cuidadosa definição da primeira instância do programa, uma vez que, a partir desta instância, são aplicadas, ações de automodificação do código.

Esse método, para auxílio ao desenvolvimento do estilo adaptativo de programação, deve contemplar uma forma de avaliação da taxa de crescimento de código, tanto no espaço como no tempo, em conseqüência da sucessiva aplicação de ações adaptativas pela execução do programa.

Os ensaios desenvolvidos neste trabalho mostraram que, como trabalho futuro, pode ser viável a construção de linguagens genuinamente adaptativas, nas quais a extensibilidade seria dispensável, uma vez que a nova linguagem pode incorporar todas as operações adaptativas que forem necessárias para a geração de programas adaptativos. Por exemplo, neste trabalho, a forma de referenciação proposta, utilizada para se identificar partes do programa, se fez por meio do desenvolvimento e disponibilização de uma primitiva de identificação presente na camada adaptativa. No entanto, em uma linguagem genuinamente adaptativa, tal recurso poderia estar incorporado à sintaxe e ao seu ambiente de execução, sem a necessidade, portanto, de extensões ou de chamadas explícitas de biblioteca para tal propósito.

Para propiciar ao usuário maior facilidade no emprego de linguagens adaptativas de programação é desejável também que, como trabalho futuro, se desenvolva um ambiente de execução que propicie outros recursos de desenvolvimento para programação adaptativa, tais como: recursos de edição, manuseio de arquivos, operações de rastreamento da execução de ações adaptativas e suas conseqüências, rastreamento das instâncias de código, monitoração de variáveis, geradores e outros elementos da camada adaptativa, etc.

Como trabalhos futuros, pode-se vislumbrar a conveniência do desenvolvimento de linguagens adaptativas de programação apoiadas em uma variedade de linguagens subjacentes.

Apesar da pesquisa envolvendo a tecnologia adaptativa ter contribuído de forma significativa em diversas áreas, poucos foram os trabalhos desenvolvidos com o foco de se explorar a questão da adaptatividade em linguagens de programação [15][16][17][18][19][20].

Assim, este artigo representa um ponto de partida para a pesquisa da área de linguagens adaptativas de programação, e mostra a viabilidade do uso de uma linguagem de programação como dispositivo subjacente de um dispositivo adaptativo, não tendo a pretensão de apresentar uma proposta de linguagem completa para fins práticos. Além disso, extrai dos resultados de seus ensaios algumas diretrizes que possam nortear o projeto de linguagens genuinamente adaptativas.

REFERÊNCIAS

[1] 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.

[2] Neto, J. J. Contribuição à metodologia de construção de compiladores. São Paulo, 272p. Tese (Livre-Docência), Escola Politécnica, Universidade de São Paulo, 1993.

[3] Pistori, H. Tecnologia em Engenharia da Computação: Estado da Arte e Aplicações. Tese de Doutorado – Escola Politécnica da Universidade de São Paulo. 2003.

[4] Santos, J.M.N. Um formalismo adaptativo com mecanismo de sincronização para aplicações concorrentes. Dissertação (Mestrado) – Escola Politécnica da USP - São Paulo, Brasil, 1997.

[5] Rady, J. A. J Stad: uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos. Tese de Doutorado, Escola Politécnica da Universidade de São Paulo, 1995.

[6] Basseto, B. A., Neto, J.J. A stochastic musical composer based on adaptative algorithms. In: Anais do XIX Congresso Nacional da SBC-99. PUC-Rio, Rio de Janeiro, Brasil – 1999.

[7] Iwai, M. K. – Um formalismo gramatical adaptativo para Linguagens dependentes de Contexto. Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2000.

[8] Rocha, R. L. A.; Neto, J.J. Uma proposta de método adaptativo para a seleção automática de soluções. In: Proceedings of ICIE Y2K – International Congress on Informatics Engineering. Buenos Aires, Argentina. 2000.

[9] Menezes, C.E.D. – Um método para a construção de Analizadores Morfológicos, Aplicados à Lingua Portuguesa, B aseado em Autômatos Adaptativos. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, São Paulo – Brasil, Julho 2000.

[10] Freitas, A. V.; Neto, J.J. Uma ferramenta para construção de aplicações multilinguagens de programação. In: CACIC 2001 – Congreso Argentino de Ciencias de la Computacion. - Argentina.

[11] Souza, M. A. A., Hirakawa, A. R., Neto, J. J. Adaptive Automata for Mobile Robotic Mapping. Proceedings of VIII Brazilian Symposium on Neural Networks - SBRN'04. São Luís/MA - Brazil. September 29 - October 1, 2004.

[12] Neto, J. J. Um levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa. Revista IEEE América Latina. Volume 5, Número 7, ISSN: 1548-0992, Páginas: 496-505, Novembro, 2007.

[13] Castro J.A.A; Neto, J. J.; Pistori, H. Determininsmo em Autômatos de Estados Finitos Adaptativos. Revista IEEE América Latina. Vol. 5, Número 7, ISSN: 1548-0992, pág. 515-521, Novembro 2007.

[14] Neto, J. J. Adaptive Automata for Context -Sensitive Languages. SIGPLAN NOTICES, Volume 29, Issue 9, Pages: 115-124, September, 1994.

[15] Freitas, A. V.; Neto, J.J. Adaptive Device with underlying mechanism defined by a programming language. Proceedings of the 4th International Conference on Information Security, Communications and Computers, December 16-18, Pages 423-428, Tenerife, Canary Islands, Spain, 2005.

[16] Freitas, A. V.; Neto, J.J. Conception of Adaptive Programming Languages. Proceedings of the 17h IASTED International Conference Modelling and Simulation, Pages: 453-458, May 24-26, Montreal, Canada, 2006.

[17] Freitas, A. V.; Neto, J.J. Um ambiente para o processamento de Linguagens Adaptativas de Programação. CACIC, Congreso Argentino de Ciencias de la Computación, San Luis, Argentina, 2006.

[18] Freitas, A. V.; Neto, J.J. Adaptive Languages and a new Programming Style. Proceedings of the 6th International Conference on Applied Computer Science, Pages: 220-225, Tenerife, Canary Islands, Spain, December 16-18, 2006.

[19] Freitas, A. V.; Neto, J. J. Linguagens de Programação aderentes ao Paradigma Adaptativo**.** Revista IEEE América Latina, Volume 5, Número 7, Páginas: 522-526, ISSN: 1548-0992, Novembro, 2007.

[20] Freitas, A. V.; Neto, J. J Adaptivity in Programming Languages**.** WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS, Volume 4, Issue 4, ISSN: 1709-0832, Pages: 779-786, April, 2007.