AdapTools 2.0: Aspectos de Implementação e Utilização

L. Jesus, D. G. Santos, A. A. Castro JR. e H. Pistori

***Resumo* — Neste trabalho apresentaremos a nova versão da ferramenta AdapTools. Este aplicativo provê mecanismos para implementação, execução e depuração de autômatos adaptativos. Daremos ênfase aos aspectos de implementação da nova estrutura de dados e do modelo de execução não-determinística, ao novo modelo de codificação utilizando o compilador AdapMap e à melhoria na inserção de rotinas semânticas, que torna desnecessária a recompilação do código da ferramenta.**

***Palavras-chave* — Tecnologia Adaptativa, AdapTools, Autômatos Adaptativos.**

I. Introdução

O AdapTools é um software livre, escrito em JAVA, que foi originalmente desenvolvido com o objetivo de executar autômatos adaptativos, bem como suas especializações: autômatos de pilha estruturados e autômatos de estados finitos [1], [2], [3]. No entanto, o estudo, o aperfeiçoamento e a expansão dessa ferramenta pode dar suporte para o desenvolvimento de outras aplicações que façam uso de Tecnologia Adaptativa. A maior vantagem dos dispositivos baseados na tecnologia adaptativa é a sua facilidade de uso, sua relativa simplicidade e o fato de sua operação poder ser descrita de forma incremental, e seu comportamento, programado para se alterar dinamicamente em resposta aos estímulos de entrada recebidos [4], [5]. Desde os autômatos adaptativos, uma série de dispositivos formais adaptativos já foram desenvolvidos e utilizados na elaboração de soluções para problemas complexos em áreas como: linguagens de programação, processamento de linguagens naturais, inteligência artificial, engenharia de software, computação natural, processamento e reconhecimento de padrões, visão computacional, aprendizagem de máquina, entre outras [6].

Além de mostrarmos as utilidades na ferramenta AdapTools (Seção II), apresentaremos, na Seção III-B, uma proposta e o desenvolvimento da integração do AdapTools com o compilador para autômatos adaptativos, já desenvolvido, chamado AdapMap. A linguagem utilizada nesse compilador oferece uma interface de alto nível para codificação e resolve algumas dificuldades encontradas na descrição e análise estrutural dos autômatos. Outra contribuição importante, ressaltada na Seção IV, é a possibilidade de alteração nas rotinas semânticas de um autômato sem necessidade de recompilação do AdapTools.
As modificações na estrutura de dados do AdapTools, como apresentado na Seção V, fornecera métodos mais eficientes de busca de regras. Com isso, podemos reconfigurar os métodos de procura e manipulação de regras da camada adaptativa, principalmente na execução de funções adaptativas de pesquisa, remoção e adição. Outra contribuição importante adicionada à ferramenta AdapTools, descrita na Seção VI, é o tratamento de autômatos não-determinísticos, proporcionando uma solução para uma série de problemas que podem ser pesquisados na área da Tecnologia Adaptativa.

II. AdapTools

Como alternativa para o aprendizado, implementação, experimentos e depuração de *autômatos adaptativos*, foi desenvolvido um ambiente baseado em software livre, chamado de *AdapTools* (Figura 1). O núcleo desse sistema é composto por uma máquina virtual que executa uma versão levemente modificada de um autômato adaptativo. Essas pequenas modificações no formalismo original simplificam e uniformizam o formato de especificação de transições internas, transições externas e das ações adaptativas elementares. Com isso, todos os elementos do dispositivo passam a poder ser apresentados através de uma única tabela [5], [7], [8]. Na versão atual, foram implementados também recursos para tratamento de não-determinismos, os quais não estavam disponíveis na versão anterior.

Dados da Submissão do Paper

L.~Jesus pertence à Universidade Federal de Mato Grosso do Sul, Departamento de Computação e Estatística. Rua Michel Bittar, 177, Coophasul, 79117-192, Campo Grande, MS, Brasil. +55(67) 33653538. (e-mail: leandro_ms@yahoo.com.br).

D.~G.~Santos pertence à Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação. Av Santa Isabel, 1396, AP 05, Barão Geraldo, 13080-012, Campinas, SP, Brasil. +55(19) 9152-2525. (e-mail: gsdenys@gmail.com).

A.~A.~Castro~JR. pertence à Universidade Federal de Mato Grosso do Sul, Campus de Coxim. Rua São Paulo, 685, Silviolândia, 79400-000, Coxim, MS, Brasil. +55(67) 32251013. (e-mail: amaury.ufms@gmail.com).

H.~Pistori pertence à Universidade Católica Dom Bosco, Centro de Ciências Exatas e da Terra. Av Tamandaré, 6000, Bloco C, Sala C102, Jd. Seminário, 79117-900, Campo Grande, MS, Brasil. +55(67) 3123502. (e-mail: pistori@ucdb.br).

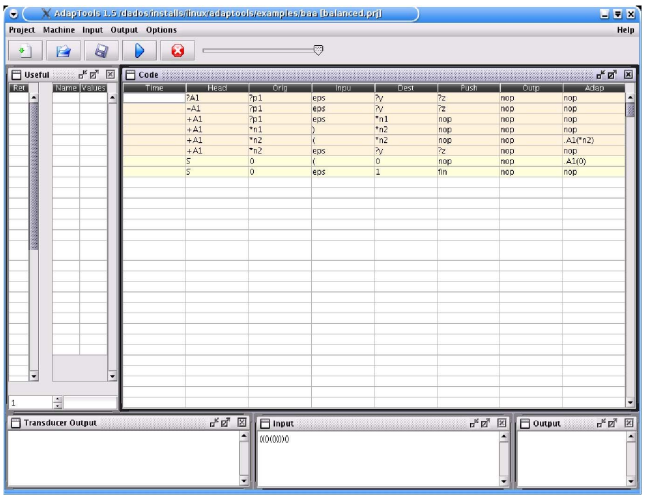

Fig. 1. AdapTools: Ferramenta de apoio à implementação, deputação e execução de *autômatos adaptativos*.

III. Mapeamento de Autômatos Adaptativos no AdapTools

A linguagem usada para a criação de autômatos adaptativos na ferramenta AdapTools é a forma de interação usuário-ferramenta. A linguagem objeto da ferramenta, linguagem AdapTools, é uma das barreiras nas codificações dos autômatos, pois possui um alto grau de complexidade, tendo como principais problemas a difícil manutenção, abstração complexa, tempo elevado para seu aprendizado e a interface que representa o autômato em forma de tabela, o que dificulta sua manipulação.

Desenvolvemos uma nova linguagem para amenizar e resolver algumas dessas dificuldades, além de contemplar características como: fácil utilização, instruções simples, fácil abstração e, principalmente, análise sintática e semântica das descrições de autômatos criadas através do AdapTools, possibilitando assim uma diminuição na quantidade de erros no desenvolvimento de novos autômatos.

A. Linguagem AdapTools

No AdapTools, os autômatos são representados através da tabela de transição. Essa tabela é composta por sete colunas: a coluna (1) (*Head*) dá um nome à sub-máquina ou à função adaptativa contida na linha. Quando o conteúdo dessa coluna for o nome de uma função adaptativa, esta deverá ser identificada pelas ações adaptativas elementares de consulta (?), de inserção (+) e de remoção (-). A coluna (2) (*Orig*), bem como a coluna (3) (*Dest*), mantém identificadores que correspondem, respectivamente, aos estados de origem e destino da transição. A coluna (4) (*Inpu*) mantém o símbolo que pode ser lido a partir de um estado, incluindo transições em vazio (epsilon). A coluna (5) (*Push*) contém um endereço de chamada de sub-máquina. A coluna (6) (*outp*) é utilizada para enviar uma saída para um transdutor. A coluna (7) (*Adap*) pode ser utilizada para a chamada de funções adaptativas [2], [1]. O AdapTools possui ainda um conjunto de palavras reservadas, descritas a seguir:

eps: representa cadeia vazia (epsilon). **nop**: nas colunas *Push*, *Outp* e *Adap*, indica que a função está inativa. **pop**: na coluna *Dest* indica uma transição de retorno de sub-máquinas. **fin**: na coluna *Push*, indica que o estado de destino é um estado final. **spc**: na coluna *Inpu*, representa os caracteres ASCII. **digit**: representa o intervalo [0..9]. **letter**: representa os intervalos [a..z] e [A..Z]. **special**: símbolos diferentes de letras e números. **other**: símbolo que não pode ser consumido encontrando-se no estado de origem da transição. **a..z**: faixa de valores ASCII de uma letra até uma outra.

Na Figura 2, apresentamos um autômato exemplo mapeado na linguagem AdapTools, o qual posteriormente mapeado na linguagem AdapMap, na Seção III-B, para podermos confrontar e verificar as diferenças entre as linguagens de descrição da estrutura do autômato.

Head	Orig	Inpu	Dest	Push	Outp	Adap
?A1	?p1	eps	?y	?z	nop	nop
-A1	?p1	eps	?y	?z	nop	nop
+A1	?p1	eps	*n1	nop	nop	nop
+A1	*n1	)	*n2	nop	nop	nop
+A1	*n2	(	*n2	nop	nop	.A1(*n2)
+A1	*n2	eps	?y	?z	nop	nop
S	0	(	0	nop	nop	.A1(0)
S	0	eps	1	fin	nop	nop

Fig. 2. AdapTools: Autômato para balanceamento de parênteses codificado no formato AdapTools.

B. Linguagem AdapMap

Como todo compilador de linguagem de alto nível, o AdapMap procura minimizar os problemas decorrentes da codificação mostrando ao usuário possíveis erros e fatos indesejáveis na construção do código, através de análises sintáticas e semânticas, gerando um código objeto (linguagem AdapTools) sem erros, o que não ocorre na linguagem AdapTools, que permite ao usuário a inserção de codificação inválida ou sem efeito semântico.

Uma característica adicional e de alta relevância da linguagem AdapMap corresponde à possibilidade de inserção de comentários sem efeitos semânticos no interior do código. Isso permite a construção de um código com maior legibilidade, o que torna os processos didático e de desenvolvimento mais fáceis, contribuindo assim com um dos principais propósitos do AdapTools: ensino de tecnologia adaptativa.

Na linguagem AdapMap, a descrição das diferentes sub máquinas de um mesmo autômato estruturado é realizada de forma mais intuitiva e através de recursos sintáticos semelhantes aos utilizados em algumas linguagens de programação "populares".

As instruções de retorno de submáquinas estão mais fáceis de serem identificadas devido à notação empregada, possibilitando uma melhor estruturação do código, e conseqüentemente, um melhor entendimento do mesmo.

Algumas considerações importantes acerca da estrutura gramatical da linguagem AdapMap são: 1) a identificação dos estados finais ocorre nas linhas iniciais de cada máquina; 2) a organização e agrupamento de máquinas e regras são flexíveis; 3) as várias possibilidades de descrição de regras extinguiram a

identificação de parâmetros não utilizados, mais especificamente, a utilização da instrução **nop** na linguagem AdapTools.

Abaixo apresentaremos um modelo estrutural do código AdapMap, que, quando compilado, gera o código objeto AdapTools da Figura 2:

```
machine main S {
    init 0; final 1;
    rule 0 {
        read '(' transit 0 function .A1( 0 );
        transit 1;
    }
}

function A1 (ARG1) {
    search {
        K = rule ARG1 transit ?y push ?z;
    }
    remove { K; }
    insert {
        rule S.ARG1 {
            transit *n1;
        }
        rule S.*n1 {
            read ')' transit *n2;
        }
        rule S.*n2 {
            read '(' transit *n2 function .A1(*n2);
            transit ?y push ?z;
        }
    }
}
```

IV. Rotinas Semânticas

Ao executar um autômato, a máquina virtual, antes de enviar o símbolo para a janela de saída, chama um método específico do objeto semântico, definido no projeto do autômato. Esse método predefinido, denominado *execute*, recebe o símbolo de saída e pode utilizar os recursos de toda a API (*Application Programing Interface*) da linguagem JAVA para analisar e gerar os eventos necessários, conforme a aplicação. Ao final, o método retorna uma cadeia de caracteres, que pode ser vazia, para ser impressa na saída auxiliar de texto do AdapTools [1].

Uma melhoria importante foi concebida no método de adição de rotinas semânticas. O pacote AdapTools dispõe de uma interface, *adaptools.vm.Semantics*, que, através da simples implementação de seus métodos, gera novos módulos para tratamento de rotinas semânticas. Após a implementação e compilação do novo módulo, é necessária apenas a cópia da nova classe semântica para o local específico da ferramenta, o diretório *semantics*, localizado junto ao arquivo executável do AdapTools.

As rotinas semânticas disponíveis na ferramenta AdapTools podem ser visualizadas no menu *Project*, simplificando a seleção do conjunto de ações semânticas a serem utilizadas pelo autômato. A Figura 3 mostra a seleção da ação semântica.

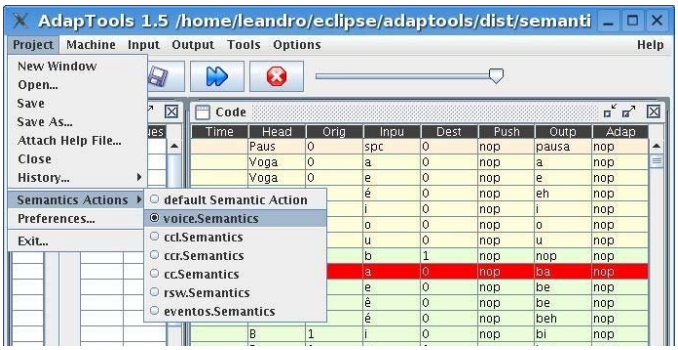

Fig. 3. AdapTools: Seleção de ação semântica.

V. Estrutura de Armazenamento de Regras

A estrutura de armazenamento e manipulação de regras foi dividida em camadas. A primeira camada fornece as regras de negócio necessárias para a manipulação da estrutura dos autômatos e faz parte do pacote *adaptools.bo*. A segunda camada fornece métodos necessários para a persistência e recuperação de dados e faz parte do pacote *adaptools.dao*.

No pacote *adaptools.bo*, encontraremos uma interface para acesso aos dados, chamada de *RulesBO*. Essa interface teve sua estrutura de métodos baseada na tabela de armazenamento utilizada anteriormente pelo AdapTools para não ocasionar maiores problemas em outras partes da ferramenta.

Ainda nesse pacote, encontraremos duas implementações diferentes para essa camada. A primeira implementação, a classe *RulesBODefaultImpl*, provê todas as funcionalidades necessárias para a manipulação da estrutura dos autômatos. A segunda implementação, a classe *RulesBOViewImpl*, contém os mesmos métodos da classe *RulesBODefaultImpl* devido a uma exigência da camada de visualização que já existia, mas com comportamentos diferentes, pois os métodos da *RulesBOViewImpl* utilizam as funcionalidades da primeira implementação, diminuindo assim a quantidade de código replicado.

O pacote *adaptools.dao* oferece um conjunto de métodos através da interface *RulesDAO* que manipulam, persistem e recuperam regras no banco de dados. A implementação dessa interface se dá através da classe *RulesDAOHSQLDBImpl*, que se utiliza de classes auxiliares para criar e executar scripts na linguagem SQL, recuperando, gravando e excluindo regras dos autômatos.

O banco de dados utilizado é o HSQLDB[1], escrito na linguagem JAVA e escolhido por ser executável em qualquer plataforma que possua uma JVM (Máquina Virtual Java). Outra característica interessante desse banco é a facilidade de integração com a ferramenta e a maneira pela qual os dados são recuperados, utilizando comandos SQL. Apesar de inserir uma camada adicional na recuperação das regras, a facilidade de uso e a eficiência na recuperação de dados justificam seu uso.

[1] http://www.hsqldb.org/

VI. Tratamento de Não-Determinismos

A importância do tratamento de não-determinismos em autômatos adaptativos é ilustrada em [8]. Esse trabalho demonstra como o não-determinismo pode ser explorado para resolver um problema relacionado à síntese de voz. A partir desse exemplo, podemos perceber que essa nova funcionalidade pode ser de grande valia na manipulação de linguagens dependentes de contexto e em várias áreas de interesse da Engenharia de Computação.

Para o tratamento e implementação do não-determinismo em autômatos na ferramenta AdapTools, propomos duas soluções: uma utilizando o mecanismo de *threads* e outra utilizando execução distribuída [8]. A seguir, faremos uma breve descrição da tecnologia e da plataforma utilizada para implementação desse recurso.

A. AdapTools Distribuído

Para o AdapTools distribuído, optamos por uma implementação utilizando RMI (*Remote Method Invocation – Chamada de Métodos Remotos*), que possibilita que um objeto operável em uma máquina virtual Java possa interagir com objetos de outras máquinas virtuais Java, independentemente da localização dessas máquinas virtuais.

A seqüência de passos realizada para o tratamento do não-determinismo de forma distribuída é mostrada e descrita com mais detalhes abaixo:

- *Kernel* avalia quantas transições possíveis podem ser executadas em um dado estado consumindo um símbolo da cadeia de entrada.
- *Kernel* delega ao objeto *NoDeterministic*, que trata o não-determinismo para executar as transições.
- Objeto *NoDeterministic* verifica quais são as máquinas disponíveis para execução distribuída.
- Objeto *NoDeterministic* pede para cada máquina que crie uma nova instância de uma *Vmds*[2].
- Objeto *NoDeterministic* cria os clientes para acesso aos objetos remotos do tipo *Vmds*.
- Objeto *NoDeterministic* inicializa os atributos de cada *Vmds* remota e coloca-as em execução.
- Objeto *NoDeterministic* aguarda uma resposta das chamadas remotas seguindo uma das estratégias que podem ser configuradas pelo usuário: 1) Seleciona a primeira transição aceita; 2) Espera todas as transições voltarem e trata a consistência, mesmo tendo o risco de algumas nunca voltarem; 3) Define o tamanho da árvore de chamadas não-determinísticas e testa a consistência das transições que foram aceitas.
- Objeto *NoDeterministic* escreve o retorno escolhido no transdutor do *Kernel* que iniciou o tratamento do não-determinismo e retorna o controle da execução para o *Kernel*.

Esses passos se repetem sempre que uma instância, local ou remota do *Kernel* encontra uma transição não-determinística.

[2] Virtual Machine Data Structures - Vmds. Implementa as principais estruturas de dados e métodos para a execução dos autômatos

B. AdapTools Local

A estratégia utilizada segue um fluxo de execução parecido com o distribuído, mas utiliza \emph{threads} em sua execução, como descrito abaixo:

- *Kernel* avalia quantas transições possíveis podem ser executadas.
- *Kernel* pede ao objeto *NoDeterministic* para executar as transições.
- Objeto *NoDeterministic* verifica que não existem máquinas disponíveis para execução distribuída.
- Objeto *NoDeterministic* cria uma *thread* para cada transição a ser tratada. Cada *thread* cria uma nova instância de uma *Vmds*.
- Objeto *NoDeterministic* fornece para cada *thread* os atributos necessários para a inicialização de cada *Vmds* criada, de modo que as *Vmds* colocam as *threads* em execução. A partir desse momento, os tratamentos realizados sobre os retornos das chamadas são os mesmos utilizados para a execução distribuída.

C. Configuração do Não-Determinismo

O AdapTools oferece três maneiras diferentes de avaliação das transições não-determinísticas. Essas abordagens foram implementadas após percebermos, durante os estudos teóricos, a existência de dois grandes problemas. No primeiro, verificamos que alguns autômatos podem gerar muitas transições não-determinísticas, principalmente quando o crescimento do autômato está dependente da distribuição dos caracteres na cadeia de entrada e das funções adaptativas. O segundo problema diz respeito à consistência dos transdutores, pois verificamos que não é trivial escolher qual regra de transdução deve ser aplicada, uma vez que podemos ter inúmeras (dependendo do número de transições não-determinísticas realizadas paralelamente) [8].

O tratamento das diferentes avaliações das transições não-determinísticas pode ocorrer de três formas:

- Utiliza-se a primeira transição que possua um retorno de um caminho aceito pelo autômato como a única verdadeira. Não se espera pelo retorno das transições que foram executadas em paralelo.
- Espera-se o retorno de todas as transições, selecionam-se aquelas que tiveram o caminho aceito e verifica-se a transdução destas são equivalentes. Caso não sejam, caracteriza-se uma inconsistência no autômato. Esta abordagem pode causar uma espera infinita, pois há o risco de se encontrar uma seqüência cíclica de transições ou chamadas de submáquina.
- Utiliza-se uma abordagem similar à do item anterior, mas respeitando-se um comprimento máximo na seqüência de passos executados em cada uma das trajetórias possíveis (simulando assim uma busca por profundidade limitada).

Para selecionar uma das opções descritas acima, clique sobre os menus *Options → Execution Type* e escolha uma das respectivas opções: First Result, Wait All Result, Limited Search.

Para definir o tamanho máximo da árvore de chamadas não-determinísticas da opção *Limited Search*, clique sobre os menus *Project → Preferences...* e preencha o campo *Limited Search* após o clique em *OK*. O valor configurado será salvo no arquivo de projeto do autômato se essa ação for realizada pelo usuário. Se o usuário preferir, no arquivo de projeto poderá ser adicionada ou alterada diretamente essa opção utilizando a codificação a seguir, que define o tamanho da árvore de chamadas recursivas em 10 (dez): *[LimitedSearch] 10;*.

Se o valor não for definido no arquivo de configuração de projeto ou na interface gráfica do AdapTools, o valor padrão a ser utilizado pela ferramenta será 5 (cinco).

VII. Conclusão

Algumas contribuições ao projeto AdapTools, como o desenvolvimento e a inclusão do compilador AdapMap, facilitaram a criação e o aprendizado dos autômatos adaptativos, através de uma interface que simplifica a codificação em uma linguagem de alto nível, resolvendo algumas dificuldades encontradas na descrição e análise estrutural dos autômatos. Outra contribuição importante é a alteração da adição de rotinas semânticas na ferramenta. Anteriormente, era necessário alteração no código fonte e recompilação do AdapTools: essa característica foi alterada e, no estado atual da ferramenta, a inclusão de novas rotinas semânticas passou a ser dinâmica.

Novas melhorias podem ser realizadas no compilador AdapMap, entre elas, uma verificação de número de parâmetros das chamadas de funções, uma melhoria na gramática para simplificar seu uso, ampliação das verificações semânticas existentes e evolução do editor que facilita sua codificação. Na situação atual, o AdapMap mapeia os autômatos adaptativos para sua configuração inicial, não possibilitando que o usuário acompanhe as modificações realizadas pelas funções adaptativas durante sua execução. Estender a ferramenta para incluir código AdapMap no mapeamento do autômato adaptativo e fornecer a possibilidade de acompanhar a execução nessa estrutura seria de grande contribuição para a pesquisa em tecnologia adaptativa.

A implementação do recurso de execução de autômatos não-determinísticos no AdapTools oferece uma solução para uma nova gama de problemas, como o processamento de linguagens dependentes de contexto, além de ser um recurso atraente para o ensino de linguagens. No AdapTools, o aluno pode depurar o não-determinismo utilizando os recursos gráficos da ferramenta, executando passo-a-passo o autômato e verificando as variáveis, o consumo da cadeia de entrada, o conteúdo da pilha, os estados finais e o estado corrente de cada percurso.

Podemos perceber que esta nova funcionalidade pode ser de grande valia na manipulação de linguagens dependentes de contexto. A evolução na ferramenta ainda ressalta outro ponto interessante a ser explorado: a utilização do AdapTools como ferramenta com finalidades pedagógicas em várias áreas de interesse da Engenharia de Computação, principalmente no ensino de Linguagens Formais e Autômatos.

Agradecimentos

O trabalho apresentado neste artigo recebeu auxílio financeiro da Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul, FUNDECT, da Financiadora de Estudos e Projetos do Ministério de Ciência e Tecnologia, FINEP, e da Universidade Católica Dom Bosco, UCDB. O pesquisador Hemerson Pistori é bolsista de Produtividade em Desenvolvimento Tecnológico e Extensão Inovadora pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq.

Referências

[1] H. Pistori and J. J. Neto, "AdapTools: Aspectos de implementação e utilização", Boletim técnico PCS - USP, 2003.

[2] H. Pistori, J. J. Neto, and E. R. Costa, "A free software for the development of adaptive automata", IV International Forum on Free Software, vol. 2, 2003.

[3] H. Pistori and J. J. Neto, "A free software for the development of adaptive automata", in Proceedings of the IV Workshop on Free Software – WSL (IV International Forum on Free Software), Porto Alegre, Brazil, June 5-7, 2003.

[4] J. Neto and M. Moraes, "Using adaptive formalisms to describe contextdependencies in natural language", Lecture Notes in Artificial Intelligence. N.J. Mamede, J. Baptista, I. Trancoso, M. das Graças, V. Nunes (Eds.): Computational Processing of the Portuguese Language 6th International Workshop, PROPOR 2003, vol. 2721, pp. 94–97, June 2003.

[5] H. Pistori, "Tecnologia adaptativa em engenharia de computação: Estado da arte e aplicações", Ph.D. dissertation, Universidade de São Paulo, São Paulo, Brasil, 2003.

[6] J. J. Neto, "Tecnologia adaptativa (unpublished)," São Paulo, Brasil, setembro de 2004.

[7] D. G. Santos, H. Pistori, A. A. Castro Jr, and J. J. Neto, "Adaptools: Umambiente gráfico de apoio ao desenvolvimento de software adaptativo", SIMS – Simpósio de Informática e Mostra de Software Acadêmico, 2005.

[8] D. G. Santos and L. Jesus, "Aspectos de projeto e implementação do não-determinismo no AdapTools e seus impactos no aperfeiçoamento da ferramenta", Campo Grande – MS, Brasil, dezembro de 2006, Monografia de Conclusão de Curso de Engenharia de Computação – UCDB.

Leandro de Jesus concluiu a graduação na Universidade Católica Dom Bosco, UCDB, em 2006, onde atuou junto ao Grupo de Pesquisa em Engenharia e Computação, GPEC, como aluno pesquisador. Atualmente cursando Mestrado em Ciência da Computação na Universidade Federal de Mato Grosso do Sul, bolsista do programa de mestrado da CAPES, vem concentrando seus estudos na subárea da teoria dos grafos, conhecida como Grafos Conservativos. Seus interesses em pesquisa concentram-se nas áreas da Teoria dos Grafos, Aprendizagem Automática, Fundamentos de Computação, em especial, Tecnologias Adaptativas.

Denys G. Santos, Engenheiro de Computação, graduou-se em 2006 pela Universidade Católica Dom Bosco, UCDB, situada na cidade de Campo Grande, no estado de Mato Grosso do Sul, Brasil, na qual atuou junto ao Grupo de Pesquisa em Engenharia e Computação, GPEC, como aluno de iniciação científica, desenvolvendo trabalhos em computação paralela e autômatos adaptativos. Presentemente cursando mestrado em Engenharia Elétrica na Universidade Estadual de Campinas, UNICAMP, na cidade de Campinas, SP, Brasil, vem atuando no Departamento de Comunicação, DECOM, mais especificamente no Laboratório de Redes de Comunicações, LARCOM, onde desenvolve sistemas de computação voltados para ambientes de redes metropolitanas.

Amaury A. C. Junior possui graduação em Ciência da Computação pela Universidade Federal de Mato Grosso do Sul (1997) e mestrado em Ciência da Computação pela mesma universidade (2003). Atualmente, é membro do corpo editorial da INFOCOMP – Journal of Computer Science (1807-4545) e professor assistente da Universidade Federal de Mato Grosso do Sul (UFMS), Campus de Coxim (CPCX). Tem experiência na área de Ciência da Computação, com ênfase em Teoria da Computação. Atua principalmente nos seguintes temas: Tecnologias Adaptativas, Autômatos Adaptativos, Projeto de Linguagens de Programação, Modelo de Computação.

Hemerson Pistori, concluiu o doutorado pela Universidade de São Paulo, USP, em 2003, e o mestrado pela Universidade Estadual de Campinas, UNICAMP. Atualmente, é bolsista de produtividade em desenvolvimento tecnológico e extensão inovadora do CNPq, professor-pesquisador da Universidade Católica Dom Bosco, UCDB, e coordenador do Grupo de Pesquisa em Engenharia e Computação, GPEC. O professor Hemerson Pistori é membro do conselho editorial do periódico científico Colabora e do International Journal for Computer Vision and Biomechanics. Seus interesses em pesquisa concentram-se na intersecção das áreas de Visão Computacional, Aprendizagem Automática e Fundamentos de Computação, em especial Tecnologias Adaptativas.