The AdapLib Library for Execution of Adaptive Devices: Architecture and Use

F. L. Siqueira

***Abstract*— In this paper it is presented the theoretical background, the architecture (using the "4+1" model), and the use of the library for execution of adaptive devices, *AdapLib*. This library was created seeking to be accurate to the adaptive devices theory, and to allow its easy extension considering the specific details of solutions that employ this kind of device. As an example, it is presented a case study in which the library was used to create a proof of concept to monitor and diagnose problems in an online news portal.**

***Keywords*— Architecture, software tool, adaptive systems.**

I. Introdução

OS DISPOSITIVOS adaptativos são uma proposta de formulação teórica para formalismos baseados em regras auto-modificáveis. Seguindo essa teoria, existem na literatura propostas de dispositivos como autômatos [1], árvores de decisão [2], tabelas de decisão [3], entre outros. Por mais que as formulações definam e provem o funcionamento desses dispositivos, ainda é necessário implementá-los – de alguma forma – para permitir a sua execução. Entretanto os usuários dessas teorias encontram pouco ou nenhum suporte ferramental para aplicá-las na prática. Por mais que esses usuários decidam criar uma implementação, muitas vezes ela é específica para um problema, indisponível para a comunidade, ou até mesmo apresenta problemas (não é eficiente, possui defeitos, é limitada conceitualmente, etc), o que inviabiliza uma reutilização e pode gerar a necessidade de criar uma nova implementação para o formalismo em questão.

Considerando esse contexto, neste trabalho é apresentada a biblioteca *AdapLib*, discutindo a sua arquitetura e seu uso. Essa biblioteca implementa a teoria de dispositivos adaptativos usando os conceitos de orientação a objetos, buscando ao mesmo tempo fidelidade à teoria, eficiência e extensibilidade. A intenção foi criar uma biblioteca de código aberto que possa ser usada em aplicações (comerciais ou não comerciais) sem a preocupação de como os conceitos são implementados; esses detalhes são tratados pela *AdapLib*.

Para apresentar essa biblioteca, este artigo está organizado da seguinte forma: na Seção II é apresentada a fundamentação teórica sobre dispositivos adaptativos. Em seguida, na Seção III é apresentada a biblioteca *AdapLib*, discutindo sua aderência à teoria, arquitetura e trabalhos relacionados. Na Seção IV é apresentado o uso da biblioteca em um estudo de caso. Por fim, na Seção V é apresentada a conclusão e as perspectivas de trabalhos futuros.

F. L. Siqueira, Universidade de São Paulo (POLI-USP), São Paulo, Brasil, levy.siqueira@poli.usp.br

II. Dispositivos adaptativos

A teoria de dispositivo adaptativo é uma formulação geral de formalismos auto-modificáveis baseados em regras que busca uma maior proximidade com o formalismo não adaptativo original [3]. Dessa forma, há uma clara separação entre a parte não adaptativa, chamada de *camada subjacente*, e a parte que permite alterar o comportamento do dispositivo de forma dinâmica, chamada de *camada adaptativa*. Essa formulação pode ser vista como uma generalização da proposta de autômatos adaptativos feita em [1] em decorrência da aplicação de seus conceitos em outras representações [2].

A camada subjacente representa um dispositivo baseado em um conjunto finito de regras. A partir de uma configuração inicial, essas regras são aplicadas em resposta a eventos de entrada, fazendo com que o dispositivo mude a sua configuração. Ao final do processamento dos eventos, o dispositivo atinge uma determinada configuração que pode aceitar ou rejeitar a entrada em questão. Segundo Neto [3] esse dispositivo pode ser formalmente definido como $D=(C, R, S, c_0, A, O)$, onde:

- C é o conjunto de possíveis configurações, com c_0 sendo a configuração inicial do dispositivo;
- S é o conjunto de eventos válidos para o dispositivo;
- $A \subseteq C$ é o conjunto de configurações que aceitam a entrada;
- O é o conjunto de símbolos que podem ser saída após a execução de uma determinada regra; e
- $R \subseteq C \times S \times C \times O$ é o conjunto finito de regras definidas para o dispositivo. Uma regra $r \in R$ é representada por $r=(c_i, s, c_f, o)$, significando que a partir de uma configuração atual $c_i \in C$, um evento $s \in S$ estimula a passagem para uma configuração $c_f \in C$ e gera o símbolo $o \in O$ na saída.

É importante ressaltar que essa formulação permite que haja mais de uma regra aplicável ao considerar uma determinada configuração atual e um evento de entrada. Caso o dispositivo possua apenas uma regra possível para cada par configuração atual e evento, ele é dito *determinístico*; caso contrário, ele é dito *não-determinístico*. Neste último caso, o dispositivo deve aplicar cada regra possível em paralelo, obtendo como resposta diversas configurações possíveis. Se a subseqüente aplicação de regras em cada uma dessas possíveis

configurações levarem a uma configuração de aceitação, o dispositivo aceita a cadeia, caso contrário a rejeita.

Sobre esse tipo de dispositivo pode ser colocada uma camada adaptativa que permite alterar o conjunto de regras e de configurações. Essa alteração é feita através de *ações adaptativas* que ficam associadas às regras da camada subjacente e são executadas antes da aplicação da regra (ações *anteriores*) ou após (ações *posteriores*). Essas ações adaptativas podem ser de três tipos: de inserção (que inserem regras), de remoção (que removem regras) e de busca (que apenas obtêm regras e dados).

Ao aplicar ações adaptativas que alteram a camada subjacente se tem, conceitualmente, um novo dispositivo adaptativo. Com isso, a aplicação das ações faz com que se transite por um espaço de dispositivos válidos a partir de um dispositivo inicial. Dessa maneira, um dispositivo adaptativo pode ser definido como $AD_k=(D_k, AM)$, considerando k execuções de regras com ações adaptativas (ou passos). No caso, D_k é o dispositivo não adaptativo (camada subjacente) e AM é o mecanismo adaptativo que define as ações adaptativas a serem aplicadas a partir de cada regra da camada subjacente. De uma forma mais detalhada, segundo [3] o dispositivo adaptativo pode também ser formalmente definido como $AD_k=(C_k, AR_k, S, c_k, A_k, O, BA, AA)$ em que:

- C_k é o conjunto de possíveis configurações após a execução de k passos, com c_k sendo a configuração inicial do dispositivo nesse passo;
- S é o conjunto de eventos válidos para o dispositivo, considerando $\varepsilon \in S$;
- $A_k \subseteq C_k$ é o conjunto de configurações que aceitam a entrada no passo k;
- O é o conjunto de símbolos que podem ser saída após a execução de uma determinada regra;
- $AR_k \subseteq BA \times C_k \times S \times C_k \times O \times AA$ é o conjunto de regras adaptativas; e
- BA e AA são conjuntos de ações adaptativas anteriores e posteriores, respectivamente (com isso, o mecanismo adaptativo $AM \subseteq BA \times R_k \times AA$, considerando as regras R_k da camada subjacente).

Em geral, as ações adaptativas são organizadas em funções adaptativas que além de agrupar as ações também recebem parâmetros, o que permite o reuso das ações adaptativas.

III. ADAPLIB

A teoria de dispositivos adaptativos define formalmente os elementos que constituem esses dispositivos e apresenta a forma de executá-los (através de um pseudo-algoritmo [3]). Buscando prover uma implementação aderente ao formalismo original que possa ser usada em aplicações, foi criada a biblioteca *AdapLib*. Essa biblioteca, feita em Java, permite representar dispositivos não adaptativos e acoplá-los como camada subjacente de um dispositivo adaptativo. Seguindo a definição, os dispositivos representados na biblioteca possuem configurações, regras, eventos e símbolos, enquanto que o mecanismo adaptativo trata das regras, ações e funções.

Além de buscar uma proximidade à teoria, um outro objetivo da *AdapLib* é ser extensível. Desenvolvedores podem implementar outros dispositivos não adaptativos e usar a infra-estrutura da biblioteca para usá-los como camada subjacente de um dispositivo adaptativo e executá-los. Mais que isso, usuários da biblioteca podem estender o formalismo original de um dispositivo para adequar a teoria às suas necessidades práticas. Pode-se, por exemplo, fazer com que ao aplicar uma regra seja executado um código específico (em linguagem de programação), executar um código ao passar para uma determinada configuração, ou até mesmo definir novos elementos para um dispositivo.

A *AdapLib*, portanto, tem como objetivo prover um ambiente de execução de dispositivos adaptativos que seja extensível e eficiente e que possa ser usado em aplicações.

A. Arquitetura

Para apresentar de uma forma mais detalhada a biblioteca *AdapLib*, será descrita a arquitetura de software idealizada usado o modelo "4+1" [5] para a definição dos pontos de vista relevantes para a arquitetura. Esse modelo propõe 5 pontos de vista: o ponto de vista lógico (que descreve as abstrações principais e seus relacionamentos), o ponto de vista de implementação (que trata da organização do código fonte, componentes, arquivos de dados e outros artefatos), o ponto de vista de processo (que trata das questões de concorrência do software), o ponto de vista de implantação (que trata do mapeamento dos elementos de software nos elementos de hardware) e o ponto de vista de caso de uso (que junta os elementos dos outros pontos de vista, representando os cenários de uso e abstraindo os principais requisitos).

Seguindo essa divisão, a seguir será apresentado cada um desses os pontos de vista, ressaltando a relação da biblioteca com o formalismo original.

1) Ponto de vista de caso de uso

Por simplicidade, ao invés dos casos de uso da biblioteca na Figura 1 são apresentadas as suas principais funcionalidades. Ao definir essas funcionalidades pretendeu-se cobrir os principais conceitos da teoria de dispositivos adaptativos, permitindo que um usuário da biblioteca possa representar um dispositivo teórico com fidelidade ao usá-la. Ainda assim existem algumas limitações, sendo que talvez a principal delas seja a obrigação dos dispositivos serem determinísticos. Por mais que a existência do não determinismo seja útil em diversas aplicações, existem algumas dificuldades de implementá-lo considerando a existência de uma camada adaptativa. Entre as dificuldades pode-se considerar a necessidade de criação de uma "onda de clones" [4] (ou seja, cópias da estrutura do dispositivo já que ao aplicar as regras adaptativas se tem diferentes dispositivos) e a ineficiência inerente a implementações de algoritmos de *backtracking* ou paralelismo.

Uma outra limitação é a respeito do uso de ações adaptativas de busca, que atualmente não são definidas. Por causa disso também não é possível definir variáveis (além dos geradores) dentro de uma função adaptativa.

- Separação entre a camada subjacente e a camada adaptativa.
- Possibilidade de uma camada adaptativa ser usada como camada subjacente (adaptatividade multi-nível).
- Possibilidade de definir comportamentos específicos ao chegar a uma determinada configuração ou ao aplicar uma regra.
- Criação de ações adaptativas de remoção que podem considerar qualquer combinação de configuração inicial, configuração final e evento (menos a que remove todas as regras do dispositivo).
- Criação de ações adaptativas de inserção, definindo chamadas de função adaptativas anteriores e posteriores na regra a ser inserida.
- Possibilidade de definir funções adaptativas, permitindo definir geradores e parâmetros.
- Representação e execução de autômatos finitos e autômatos finitos adaptativos determinísticos.
- Possibilidade de definição ou alteração do comportamento das regras, configurações e eventos, ou até mesmo da execução do dispositivo subjacente ou do dispositivo adaptativo.
- Existência de *log* para a execução do dispositivo.

Figura 1. As principais funcionalidades da biblioteca.

Seguindo a formulação dos dispositivos adaptativos, uma outra consideração é em relação à adaptatividade multi-nível. Existem algumas possíveis interpretações de quais são as regras e as configurações da camada adaptativa subjacente. A versão atual da biblioteca adota, por simplicidade, que as regras são as regras adaptativas e as configurações são as mesmas da camada subjacente não adaptativa. Entretanto ainda é necessária uma melhor análise dessa questão do ponto de vista teórico para considerar que a implementação existente atualmente na biblioteca é suficientemente adequada.

Por fim, atualmente está disponível apenas a implementação de autômatos de estados finitos como camada subjacente. Entretanto, considerando a arquitetura adotada, pode-se implementar facilmente outros tipos de dispositivos como camada subjacente (na Seção III.A.2 é descrito brevemente como um desenvolvedor pode fazê-lo).

2) Ponto de vista lógico

As classes e interfaces da atual versão da biblioteca serão apresentadas nesta seção de uma forma simplificada, focando em alguns aspectos principais: a descrição do dispositivo contemplando a camada subjacente e a adaptativa (Figura 2), as funções adaptativas (Figura 3), um cenário de execução (Figura 4) e um exemplo de camada subjacente (Figura 5).

Seguindo a definição apresentada na Seção II, na Figura 2 são apresentados os principais elementos que descrevem um dispositivo e um dispositivo adaptativo. Um *Dispositivo* possui *Eventos*, *Configurações* (algumas sendo de aceite e uma sendo a inicial), *Símbolos de saída* e *Regras*. Um dispositivo pode ser camada subjacente de um *Dispositivo Adaptativo*, que também é um *Dispositivo* – o que permite a existência de adaptatividade multi-nível. Nesse tipo de dispositivo as *Regras* são especializadas considerando que elas podem fazer *Chamadas a funções adaptativas*. Além disso, o *Dispositivo Adaptativo* tem um *Mecanismo Adaptativo* que é responsável por tratar das mudanças da estrutura da camada subjacente (usadas pelas ações adaptativas).

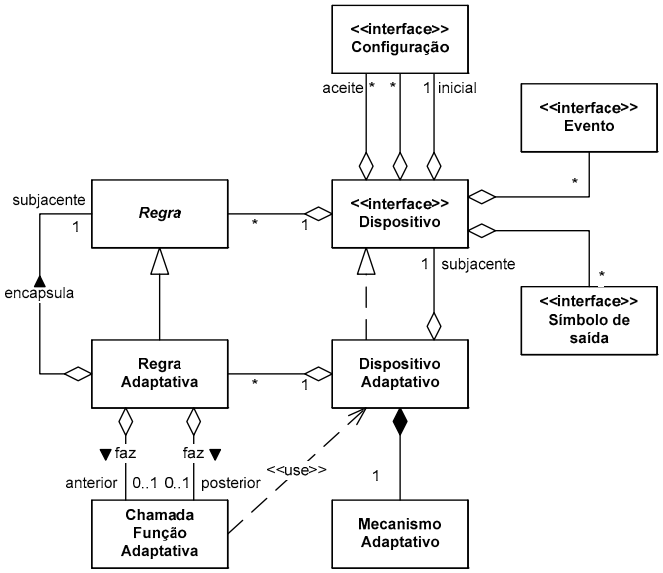

Figura 2. As classes que tratam do dispositivo e do dispositivo adaptativo.

Especificamente sobre a função adaptativa, na Figura 3 são apresentadas as classes que a representa. As *Chamadas de Função Adaptativas* (que são parte de uma regra adaptativa) recebem parâmetros passados por valor (*Parâmetro Valor*). Com esses valores é então chamada uma *Função Adaptativa* que executa uma *Ação Adaptativa* antes da execução das demais ações (pré) e uma ação que é executada após as demais ações (pós). Essas ações podem ser *Ação Adaptativa Inserção* e *Ação Adaptativa Remoção*. Uma *Ação Adaptativa* define parâmetros que estabelecem padrões para a configuração de origem e de destino e também para o evento (ações de inserção precisam definir todas essas informações, seja com valores ou com referência a valores que serão resolvidas durante a chamada da função, enquanto que as ações de remoção devem definir pelo menos uma dessas informações).

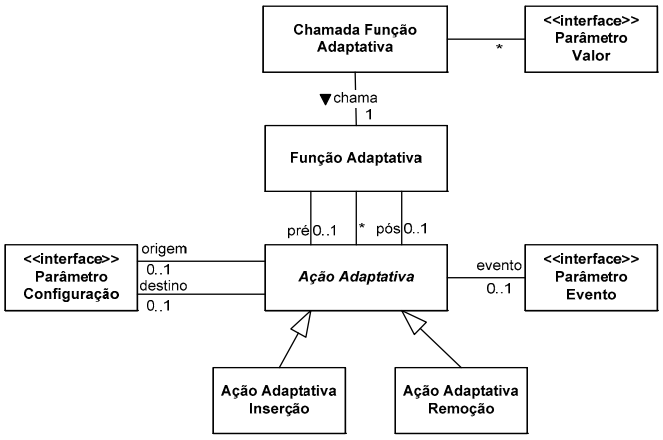

Figura 3. As classes que tratam das funções adaptativas.

Outra parte fundamental da biblioteca são as classes de execução. A dinâmica entre essas classes segue a idéia do algoritmo de funcionamento de um dispositivo adaptativo apresentado em [3] – considerando as limitações existentes na biblioteca. Esse cenário de execução é apresentado de forma simplificada na Figura 4. O responsável pela execução de um

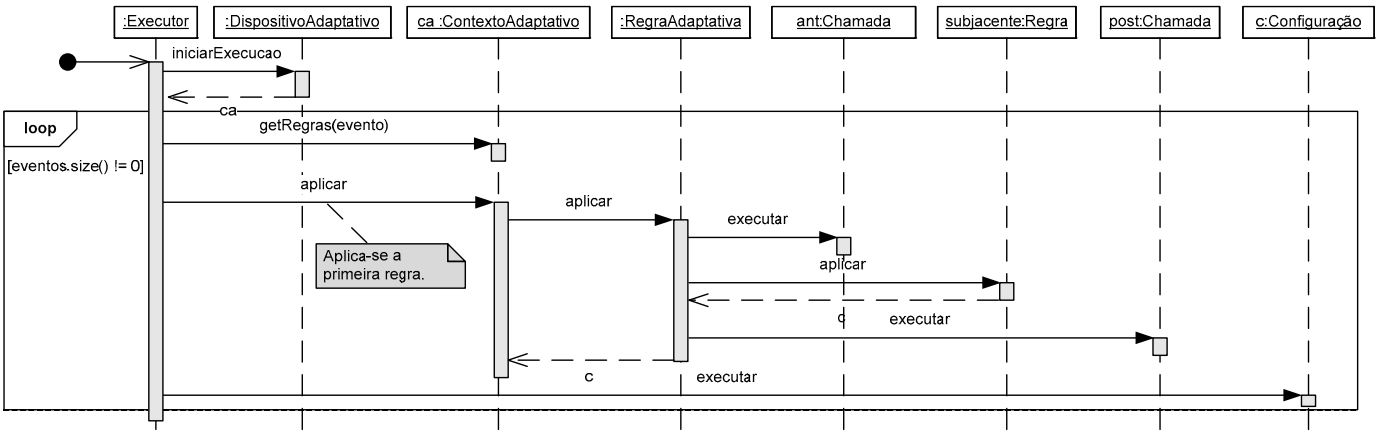

Figura 4. O cenário de execução de um dispositivo adaptativo.

dispositivo é a classe *Executor*. Ela solicita ao *Dispositivo Adaptativo* um *Contexto de Execução Adaptativo* e assim começa a executar o dispositivo, usando as informações desse contexto. Enquanto houver eventos, pede-se ao contexto *Regras Adaptativas* que possam ser aplicadas. Caso não haja regras a execução termina, rejeitando a entrada. Se houver regras, aplica-se uma delas (a primeira de uma seqüência). Essa aplicação é feita ao chamar a função adaptativa anterior (se ela estiver definida), ao aplicar a regra subjacente e ao chamar a função adaptativa posterior (se ela estiver definida), nessa ordem. Se ao chamar a função adaptativa anterior à estrutura do dispositivo for alterada de tal forma que a regra que está sendo aplicada não exista mais, interrompe-se a aplicação da regra e o laço é reiniciado. Entretanto, caso a regra atual seja removida ao executar a função adaptativa posterior, nada é feito. Com a aplicação da regra, o dispositivo atinge uma nova configuração, que é executada e que com isso se volta ao laço para aplicar mais regras.

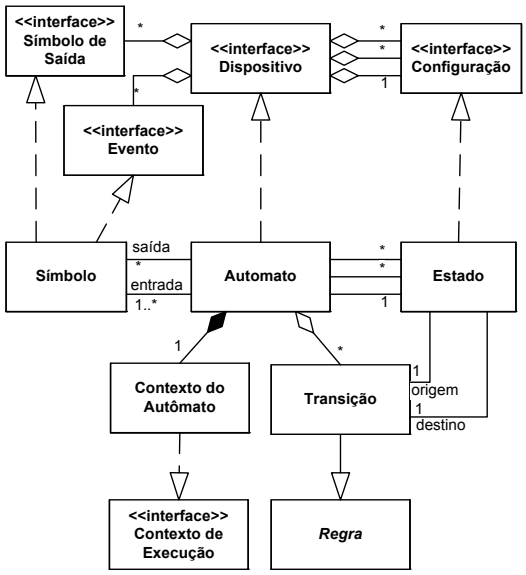

Figura 5. As classes que tratam do autômato como dispositivo subjacente.

Por fim, na Figura 5 são apresentadas as classes que tratam da implementação do autômato como dispositivo subjacente. Atualmente a biblioteca disponibiliza apenas esse dispositivo, mas caso seja necessário um outro dispositivo, o desenvolvedor deverá implementar as interfaces e classes abstratas que representam os conceitos de dispositivo e os conceitos associados a ele, ou seja, *Dispositivo*, *Contexto de Execução*, *Regra*, *Configuração*, *Evento* e *Símbolo de Saída*. Por exemplo, no caso do autômato, as classes criadas foram respectivamente *Autômato*, *Contexto do Autômato*, *Transição*, *Estado* e *Símbolo* (implementando ao mesmo tempo *Evento* e *Símbolo de Saída*). O desenvolvedor pode reusar essas classes para implementar outros tipos de autômato como dispositivos, apenas redefinindo e adicionando operações e atributos que sejam necessários.

Aos interessados em maiores detalhes sobre a execução e as classes, recomenda-se olhar o código fonte (disponível em: http://www.levysiqueira.com.br/projetos/adaplib/).

3) Ponto de vista de Implementação

As classes definidas para a biblioteca foram organizadas em 4 pacotes, conforme apresentado no diagrama de pacotes da Figura 6. O pacote raiz contém as interfaces e as classes que definem o que é dispositivo e permitem a sua execução. O pacote *Adaptativo* contém os elementos referentes a adaptatividade e no seu sub-pacote *Função* estão as classes e interfaces que tratam das funções adaptativas e a hierarquia de tipos de parâmetros. As classes referentes ao mecanismo subjacente estão organizadas no pacote *Subjacente*. Por enquanto nesse pacote estão disponíveis apenas as classes que permitem um autômato como dispositivo subjacente.

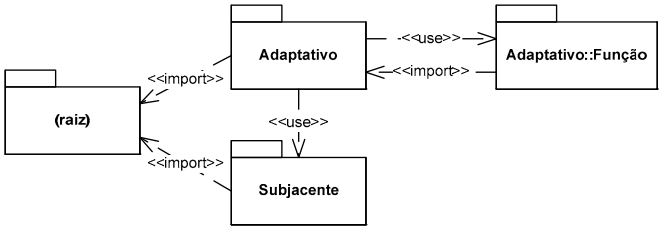

Figura 6. Os pacotes definidos para organizar as classes da biblioteca.

4) Ponto de vista de Processo

Como atualmente o software utiliza apenas um *thread* de execução e não são especificados requisitos não funcionais de desempenho, este ponto de vista não será detalhado. Em futuras versões que tratem da questão do não determinismo, possivelmente haverá uma necessidade de trabalhar com mais

de um *thread* de execução. Considerando essa necessidade, decidiu-se separar a execução do dispositivo do dispositivo em si, criando a classe *Contexto de Execução*. A intenção é permitir que existam diferentes contextos, um para cada execução possível. Entretanto, como ainda não foram analisados os detalhes de possíveis soluções, provavelmente serão necessárias mudanças no modelo em questão.

5) Ponto de vista de Implantação

A biblioteca é distribuída em duas versões: o código fonte e um arquivo Jar (já que a biblioteca é feita em Java), o que permite agregar todos os arquivos compilados em um só.

B. Trabalhos relacionados

Na literatura existem alguns outros trabalhos que propõem criar ambientes de execução para dispositivos adaptativos. O principal deles é o *AdapTools*, que é uma ferramenta para aprendizado, implementação e depuração de dispositivos adaptativos [6]. Nessa ferramenta é possível representar autômatos de pilhas estruturados e autômatos de estados finitos usando uma linguagem própria (linguagem *AdapTools*). Dentre as principais funcionalidades dessa ferramenta está a possibilidade de ver em tempo de execução o funcionamento do autômato, as regras aplicadas, criadas e removidas (organizadas em uma tabela) e a possibilidade de criar "rotinas semânticas", executadas após o término do processamento da cadeia. Além disso, é também possível trabalhar com não-determinismos e armazenar as regras em um banco de dados.

O *AdapTools* é, portanto, um ambiente mais completo e maduro (desenvolvido desde 2001) que a *AdapLib* – que é apenas uma biblioteca para a execução. Por mais que o *AdapTools* possa ser usado de forma "embutida" em outra aplicação [2], não há muitas informações de como fazê-lo – ou até mesmo de como estendê-la para tratar de detalhes específicos de algumas soluções. Além disso, a ferramenta trata de autômatos e não dispositivos adaptativos em geral. Dessa maneira, a *AdapLib* pretende ser uma biblioteca simples e focada apenas na execução de dispositivos, sendo possível projetar um dispositivo usando as abstrações definidas na própria linguagem de programação – usando para isso o paradigma orientado a objetos.

Alguns outros trabalhos não apresentam ferramentas, mas provas de conceito que podem ser usadas para simulação de autômatos adaptativos. Werneck [7] apresenta uma prova de conceito que permite executar um autômato adaptativo a partir de uma entrada. Entretanto, esse simulador apresenta algumas limitações teóricas: não é possível trabalhar com ações de remoção e de busca, não é possível passar parâmetros para as funções adaptativas e trata-se apenas de autômatos determinísticos. De forma mais abstrata, Vega [4] discutiu o projeto de um ambiente de execução de autômatos adaptativos. Por mais que não seja proposta uma ferramenta em si, uma prova de conceito foi discutida e apresentada – servindo como base para a discussão de problemas e de detalhes de implementação.

IV. Estudo de caso

Um dos principais objetivos da biblioteca *AdapLib* é que ela possa ser facilmente absorvida por aplicações que usem os conceitos de dispositivos adaptativos. Para discutir como a biblioteca pode ser usada será apresentado um projeto que usou a biblioteca para a criação de uma prova de conceito.

O projeto em questão era a criação de um software para o monitoramento e diagnóstico de problemas em um portal de notícias *online*. O monitoramento precisaria considerar os ambientes de produção e de administração do portal, levando em conta as máquinas existentes (servidores *web*, servidores de aplicação e servidores de banco de dados) e alguns aplicativos específicos. Com as informações obtidas através de sondas existentes nas máquinas e nos aplicativos, era necessário diagnosticar o problema existente, buscando assim direcionar o trabalho da equipe de manutenção do portal. A solução adotada deveria considerar a grande quantidade de acessos ao portal e a possibilidade de inserção de novas máquinas e aplicativos, além da preocupação em se obter uma alta disponibilidade. Considerando que o objetivo do estudo de caso neste trabalho é apenas analisar o uso da *AdapLib*, não serão aqui discutidos maiores detalhes sobre o contexto e o racional para se chegar a solução aqui apresentada.

A solução adotada para a prova de conceito foi a criação de um autômato de estados finitos adaptativo que usava como alfabeto de entrada as possíveis informações obtidas por sondas nas máquinas e nos aplicativos. A partir dessas entradas, projetou-se um autômato que permitia diagnosticar problemas de desempenho e de exceções causadas nos aplicativos. Na Figura 7 é apresentada uma parte do autômato que trata do alarme de exceção – a que trata da análise do problema ocorrida no *Servidor de Aplicação 1*.

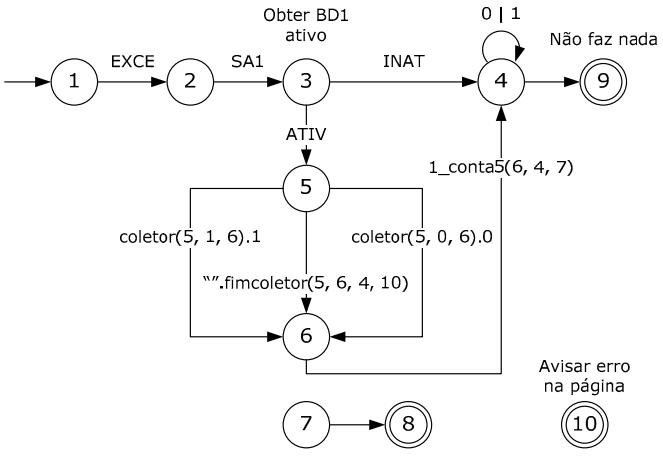

Figura 7. Parte do autômato de estados finitos adaptativo implementado (os círculos representam estados, as setas transições e os círculos com bordas duplas estados finais).

Para executar o autômato de estados finito projetado foi utilizada a *AdapLib*. Como a biblioteca já possui uma implementação de autômato de estados finitos como dispositivo subjacente (apresentada na Seção III.A.2), só foi necessário adicionar algumas classes para definir o comportamento específico para o projeto. Considerando o contexto, foram necessárias duas mudanças no formalismo de

autômato de estados finitos adaptativos: (1) permitir que os estados obtivessem informações e (2) que os estados finais emitissem avisos sobre o problema diagnosticado. A mudança (1) fez com que fosse criado um tipo especial de estado que obtém informações e a solução (2) fez com que fossem criados dois outros tipos de estados: um estado final que executava alguma ação na interface homem-computador e um estado final que informava à interface homem-computador os detalhes do problema diagnosticado. Para representar esses três novos tipos de estado foi necessário apenas criar especializações da classe *Estado* e redefinir a operação *executar*, que permite que um estado tenha um comportamento ao ser alcançado.

Após criar essas novas classes para permitir uma comunicação diferenciada com a interface homem-computador, foi necessário passar para a linguagem de programação o autômato e as funções adaptativas projetadas. Como exemplo disso, uma parte desse código em Java está apresentada na Figura 8, sublinhando uma variável que referencia a função adaptativa para o coletor de nomes (na transição do estado 5 para o 6). Cada estado é representado por um objeto e as transições entre eles são adicionadas à camada subjacente (caso não haja chamadas de funções adaptativas) ou à camada adaptativa (que se encarrega de registrar na camada subjacente a regra subjacente).

```
Automato a = new Automato();// Criando o autômato

// Colocando uma camada adaptativa no autômato
DispositivoAdaptativo aa =
     new DispositivoAdaptativo(a);

Estado e1 = new Estado("1"); // Estado 1
a.adicionarConfiguracao(e1, true, false); // inicial
Estado e2 = new Estado("2"); // Estado 2
a.adicionarConfiguracao(e2, false, false);
... // os outros estados

// Criando as transições
a.adicionarRegra(e1, "EXCE", e2); // (1, "EXCE, 2)

// (5, coletor(5, "0", 6)."0", 6)
ArrayList p = new ArrayList();
p.add(new ParametroValorConfiguracao("5"));
p.add(new ParametroValorEvento("0"));
p.add(new ParametroValorConfiguracao("6"));

aa.getMecanismoAdaptativo().adicionarRegraAdaptativa(
     new ChamadaFuncaoAdaptativa(coletor, p),
     e5, "0", e6, null);
... // as demais transições
```

Figura 8. Fragmento de código exemplificando o uso da *AdapLib* – sublinhado está a referência a um objeto com a função adaptativa do coletor.

Com o autômato completamente representado na linguagem Java foram necessários diversos testes para garantir que o autômato idealizado estava corretamente representado. Além dos erros na passagem para a linguagem de programação, também haviam erros no autômato idealizado – o que dificultou ainda mais a atividade de teste.

V. Conclusão

Neste trabalho foi apresentada a fundamentação teórica, a arquitetura e o uso em um estudo de caso da biblioteca *AdapLib*. Essa biblioteca busca seguir com fidelidade a teoria de dispositivos adaptativos, permitindo executá-los seguindo os conceitos dessa teoria. Um outro requisito importante no projeto da biblioteca foi permitir que os usuários a estendam, buscando fazer com que aplicações possam usar a biblioteca com facilidade – mesmo nos casos em que a teoria não possa ser seguida à risca ao considerar questões práticas. Além dessas questões, um outro ponto importante na construção da biblioteca foi a eficiência. Entretanto esse requisito não funcional não foi discutido neste trabalho, cabendo a trabalhos futuros fazê-lo.

Na versão atual da biblioteca alguns conceitos definidos pela teoria de dispositivos adaptativos ainda não foram implementados: ação adaptativa de busca e o não determinismo. Além disso, o único dispositivo disponível pela biblioteca é o autômato (o que não impede que outros dispositivos sejam criados) e ainda é preciso uma análise mais criteriosa para permitir adaptatividade multi-nível. Em trabalhos futuros pretende-se tratar dessas limitações e adicionar outras funcionalidades à biblioteca, sejam elas para tratar de aspectos teóricos ou de implementação.

Agradecimentos

À Reginaldo Inojosa Filho pelo auxílio na versão inicial da biblioteca e pela realização em conjunto do estudo de caso.

À FAPESP pelo apoio na realização de meu doutorado, durante o qual foi feito este trabalho.

Referências

[1] J. J. Neto, “Adaptive automata for context-dependent languages”, ACM SIGPLAN Notices, vol. 29, i. 9, pp. 115-124, Sep. 1994.

[2] H. Pistori, “Tecnologia adaptativa em engenharia de computação: Estado da arte e aplicações”, Tese de doutorado, orientada por J. J. Neto, Universidade de São Paulo, São Paulo, Brasil, 2003.

[3] J. J. Neto, “Adaptive Rule-Driven Devices - General Formulation and Case Study”, Lecture Notes in Computer Science: Implementation and Application of Automata, vol. 2494, pp. 466-470, 2002.

[4] I. S. Vega, “Implementação OO para Autômatos Adaptativos: Impactos das Tomadas de decisão efetuadas durante o Design”, Resumo do Mini-curso, Segundo Workshop de Tecnologia Adaptativa (WTA 2008), São Paulo, pp. viii-ix, 2008.

[5] P. Kruchten, “The 4+1 View Model of Architecture”, IEEE Software, vol. 12, no. 6, pp. 42-50, Nov. 1995.

[6] L. Jesus, D. G. Santos, A. A. Castro JR. e H. Pistori, “AdapTools 2.0: Aspectos de Implementação e Utilização”, IEEE Latin America Transactions, vol. 5, no. 7, pp. 527-532, Nov. 2007.

[7] N. L. Werneck, “Biblioteca para autômatos adaptativos com regras de transição armazenadas em objetos feita em C++”, Segundo Workshop de Tecnologia Adaptativa (WTA 2008), São Paulo, pp.22-26, 2008.

Fábio Levy Siqueira é Engenheiro de Computação formando na Escola Politécnica da Universidade de São Paulo (2002) e Mestre em Engenharia Elétrica pela mesma instituição (2005). Atualmente é doutorando em Engenharia Elétrica e pesquisador na Escola Politécnica da Universidade de São Paulo. Tem experiência na área de Engenharia da Computação, com ênfase em Engenharia de Software, atuando principalmente nas seguintes áreas: engenharia de requisitos, transformação de modelos e desenvolvimento distribuído de software.