

FRAMEWORK PARA GERAÇÃO DE FERRAMENTAS DE EDIÇÃO PARA AUXILIAR O PROJETO DE APLICAÇÕES ADAPTATIVAS

F M B REIS e A R CAMOLESI

fred_bertoluci@hotmail.com; camolesi@femanet.com.br

RESUMO: Este trabalho apresenta a concepção e a codificação de um framework para auxiliar um especialista em dispositivos adaptativos a fim de obter ferramentas de edição que auxiliam o projeto de aplicações adaptativas.

PALAVRAS-CHAVE: Tecnologia Adaptativa, Ferramentas de Edição, Dispositivos Adaptativos.

ABSTRACT: This paper presents the design and coding of a framework to assist an expert in adaptive devices to obtain editing that assist the design of adaptive applications tools.

KEYWORDS: Adaptive Technology, Editing Tools, Adaptive Devices.

1. Introdução

Aplicações complexas são caracterizadas por componentes e aspectos cuja estrutura e comportamento comumente podem modificar-se CAMOLESI (2007). Tais aplicações possuem um comportamento inicial definido por um conjunto de ações, que desempenham suas funções elementares e podem ter o seu comportamento modificado durante a execução para dar suporte a novas funcionalidades. Tais modificações são decorrentes dos estímulos de entrada a que são submetidos no sistema, e/ou da ocorrência de suas ações internas.

Uma técnica utilizada para auxiliar os projetistas no projeto de aplicações com comportamento modificável é a tecnologia adaptativa NETO (1993). A tecnologia adaptativa envolve um dispositivo não adaptativo (subjacente) já existente em uma camada adaptativa que permite realizar mudanças no

comportamento da aplicação definida PISTORI (2003). É possível citar, por exemplo, trabalhos relacionados a reconhecedores sintáticos adaptativos, NETO (1988), os Statecharts Adaptativos ALMEIDA (1995) - empregados na modelagem de sistemas reativos - e a modelagem de aplicações complexas com base no ISDL Adaptativo CAMOLESI (2004a). O desenvolvimento de tecnologia adaptativa, aplicado a sistemas de dispositivos não adaptativos dirigidos por regras, vem sendo pesquisado com totais preocupações a fim de que o usuário consiga gerenciar novos dispositivos adaptativos.

Em REIS (2012) foi criado um framework para simulação de autômatos adaptativos proposto em CAMOLESI (2007). Tal framework auxilia na simulação de autômatos adaptativos, porém, não foi criada a parte de persistência no modelo lógico.

As ferramentas existentes para o projeto de aplicações gráficas são muito específicas e restritas a um ou alguns formalismos apenas, além de não contemplarem o uso de dispositivos adaptativos. Quando um novo dispositivo adaptativo é projetado, novas ferramentas para especificação, simulação e verificação de aplicações devem ser construídas. O desenvolvimento de tais ferramentas é lento, e por tal, acaba desestimulando a criação de novos dispositivos adaptativos. Com o objetivo de melhorar esta etapa foi realizado este projeto, o qual teve como foco a criação de um framework para definição de forma gráfica dos elementos conceituais, e representação de especificações usando um determinado dispositivo adaptativo. Tal framework deve possibilitar, depois de realizado, a definição de uma aplicação usando a ferramenta gráfica, a qual persiste no modelo lógico proposto por CAMOLESI (2007).

Desta forma, tal framework irá complementar o trabalho desenvolvido em REIS (2012) e facilitará o trabalho de especialistas em dispositivos adaptativos na obtenção de ferramentas gráficas para auxiliar a realização de suas tarefas.

Este trabalho apresenta-se organizado da seguinte forma: inicialmente, na Seção 1, foi apresentada uma visão geral do trabalho, seus objetivos e as suas justificativas. Na sequência, a Seção 2 apresenta uma breve introdução dos conceitos de Tecnologia Adaptativa, de ferramentas para o projeto de aplicações que se utilizam deste conceito, e descreve a arquitetura geral do Modelo Lógico utilizado para suportar a representação computacional de dispositivos adaptativos. Com base no Modelo Lógico, será realizado o desenvolvimento do Framework Adaptativo descrito na Seção 3. Por fim, na Seção 4 serão tecidas algumas conclusões e trabalhos futuros.

2. Tecnologia Adaptativa

O trabalho proposto fundamenta-se na Tecnologia Adaptativa, a qual permite que um dispositivo adaptativo seja capaz de se auto modificar, ou seja, permite que seu comportamento mude dinamicamente em tempo de execução e sem influência externa, quando detecta uma situação em que, para ele não é mais possível dar continuidade a sua execução sem antes alterar seu comportamento.

As maiores vantagens dos dispositivos adaptativos são, a sua facilidade de uso, sua relativa simplicidade, e o fato de ser muito rara a necessidade de existir uma descrição completamente especificada do comportamento do dispositivo para que ele possa ser utilizado. Em lugar disso, sua operação pode ser descrita de forma incremental, e seu comportamento, programado para se alterar dinamicamente em resposta aos estímulos de entrada recebidos. Adicionalmente, como o conjunto de regras que definem o seu comportamento também se altera ao longo da sua operação, pode-se dizer que, o próprio dispositivo adaptativo programa, de alguma forma, a representação do conhecimento adquirido através de sua sequência recebida de estímulos de entrada.

Dispositivos adaptativos podem ser empregados em uma grande variedade de situações, principalmente, em segmentos complexos da solução de um problema no qual sejam realizadas tomadas não triviais de decisão.

Para auxiliar o projeto de aplicações que use tecnologia adaptativa, faz-se necessária a utilização de um ambiente que integra um conjunto de ferramentas que dê suporte aos projetistas na realização de seu trabalho. A Figura 1, apresentada por CAMOLESI (2007), ilustra a arquitetura geral de tal ambiente, o qual é organizado em 3 (três) grupos de ferramentas: Ferramentas de Edição, Ferramentas de Análise e Ferramenta de Implementação.

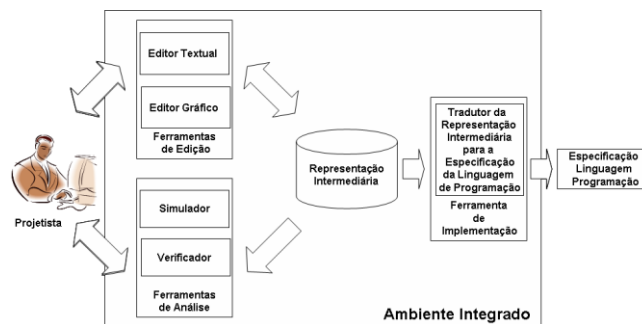


Figura 1. Ambiente para o projeto de aplicações usando Tecnologia Adaptativa CAMOLESI (2007).

Valendo-se das Ferramentas de Edição, um projetista de aplicações, utilizando-se de um dispositivo adaptativo específico, poderá realizar a especificação de sua aplicação com o auxílio de um editor de texto qualquer ou de um editor gráfico. Para possibilitar o intercâmbio das especificações produzidas, os editores

devem gerar objetos no Modelo Lógico (Figura 1). Caso a especificação seja produzida em um editor textual, esta deverá ser compilada para transformar a codificação realizada no formato definido para o Modelo Lógico. Depois de realizada a especificação, o projetista de aplicações poderá utilizar as Ferramentas de Análise. Tais ferramentas utilizam a codificação da especificação (no formato do Modelo Lógico) como base, e permitem a realização de uma análise do comportamento da aplicação adaptativa em desenvolvimento.

Por fim, depois de especificada e analisada a representação de uma aplicação, o projetista pode utilizar-se das Ferramentas de Produção de Representações Físicas, gerar uma representação de uma aplicação em um determinado padrão de linguagem de representação física e obter a aplicação desejada.

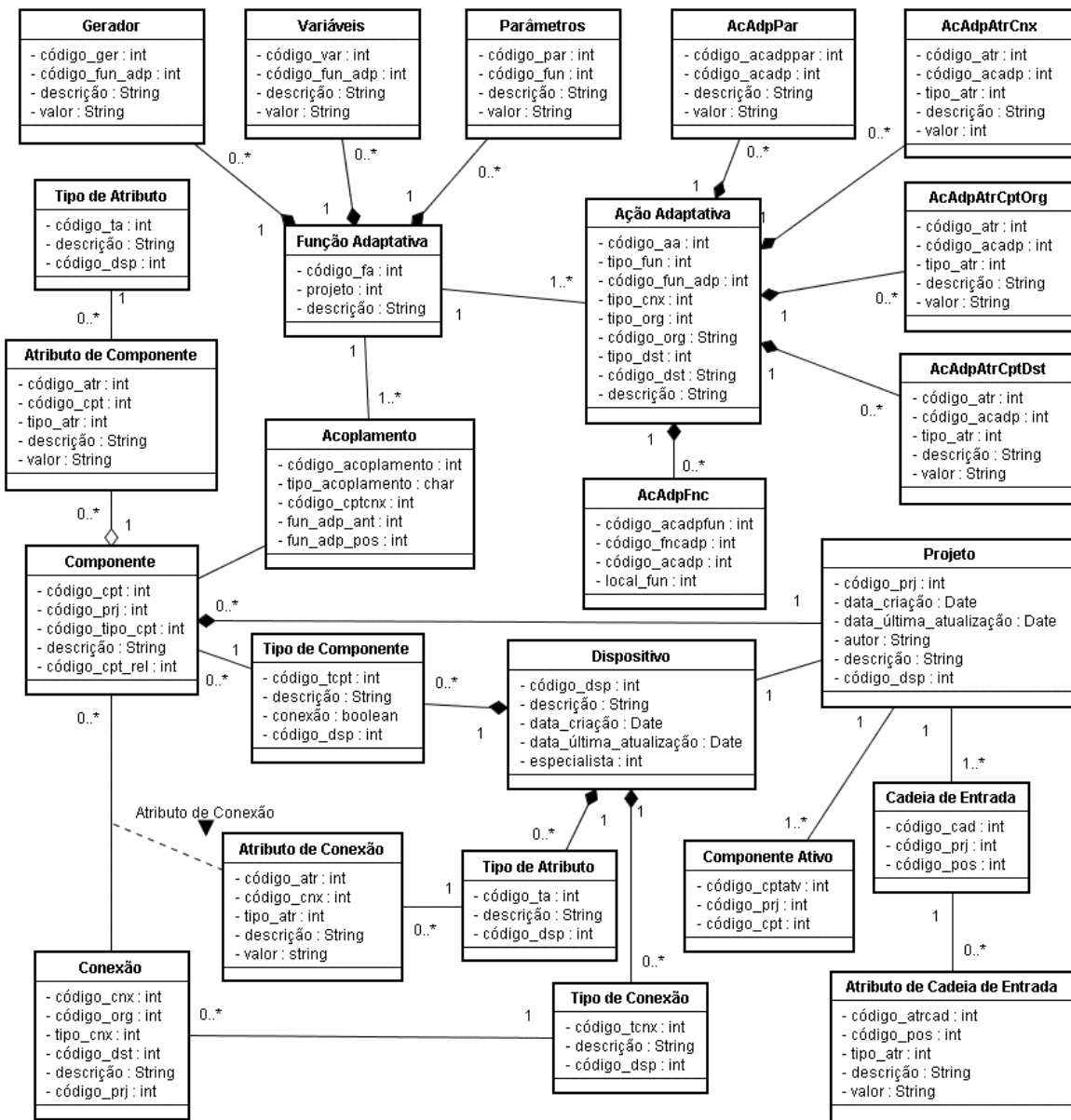


Figura 2. Modelo Lógico proposto em CAMOLESI (2007).

O ponto fundamental para a integração destas ferramentas é o Modelo Lógico proposto em CAMOLESI (2007) é o núcleo central do Framework Adaptativo proposto. Por tal motivo, será descrito nesta seção, um resumo do Modelo Lógico para representação de dispositivos adaptativos e aplicações modeladas com base nestes dispositivos.

O Modelo Lógico proposto é dividido em quatro camadas:

- Camada de especificação de dispositivos;
- Camada de especificação de aplicações – não adaptativa;
- Camada de especificação de funções e ações adaptativas;
- Camada de representação de memória;

A definição do modelo em camadas é fundamental para o desenvolvimento do trabalho, uma vez que permite separar a camada não adaptativa da camada adaptativa. Acima (Figura 2) foi mostrado o Modelo Lógico (diagrama de classe) completo.

A seguir serão apresentados os elementos que constituem cada camada e os seus respectivos relacionamentos.

A. Camada de Especificação de Dispositivos

Os elementos da Camada de Configuração de Dispositivos, também denominada na teoria como Núcleo Subjacente NETO (1991), representa os elementos lógicos referentes à definição de um determinado dispositivo dirigido por regras não adaptativo. A Figura 3 ilustra esta camada, na qual um especialista, depois de estender os conceitos formais de um determinado dispositivo, deve realizar um mapeamento da estrutura conceitual do referido dispositivo para esta camada. Ao instanciar objetos nesta estrutura, o especialista define os elementos conceituais: tipos de componentes, tipos de conexões e tipos de atributos que definem a estrutura de dispositivo dirigido por regras não adaptativas.

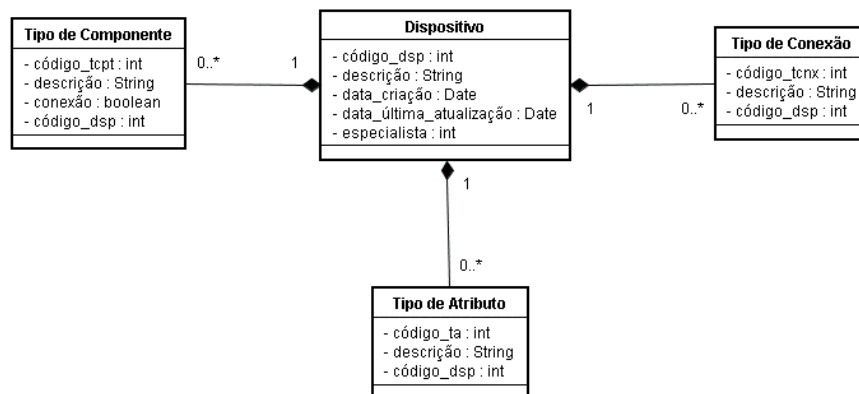


Figura 3. Camada de Configuração de Dispositivos - CAMOLESI (2007).

B. Camada de Especificação de Aplicações

A Camada de Especificação de Aplicações – Não adaptativa tem por objetivo representar os elementos referentes ao comportamento de uma determinada aplicação que está sendo modelada por certo projetista. Um projetista, ao modelar a aplicação, instancia objetos da camada de núcleo subjacente, e define o comportamento da aplicação gerando novos objetos na camada de especificação de aplicações.

Para cada elemento definido nesta camada, são associados os seus tipos correspondentes à camada de núcleo subjacente, bem como é realizada uma relação entre os elementos dos diversos conjuntos que representam esta camada. Esta, como foi dito anteriormente, representa o comportamento da aplicação e, conseqüentemente, a especificação não adaptativa de uma determinada aplicação. A Figura 4 apresenta a organização estrutural da referida camada.

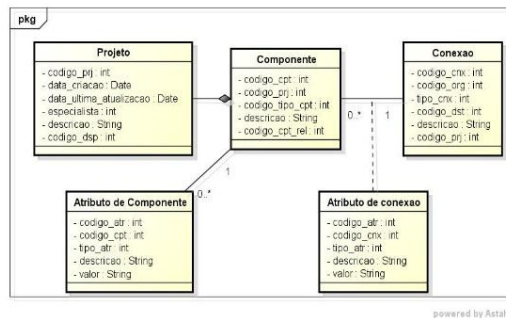


Figura 4 Camada de Especificação de Aplicações – Subjacente. - CAMOLESI (2007).

C. Camada de Especificação de Funções e Ações Adaptativa

A Camada de Especificações de Funções e Ações Adaptativas (Figura 5) tem por objetivo armazenar informações referentes às funções e ações adaptativas que são utilizadas em uma determinada aplicação. Os objetos desta camada são definidos com base nos objetos da camada de especificação, pois deve estar associado aos componentes e conexões de um determinado projeto. A utilização desta camada é de fundamental importância para a estrutura proposta neste trabalho.

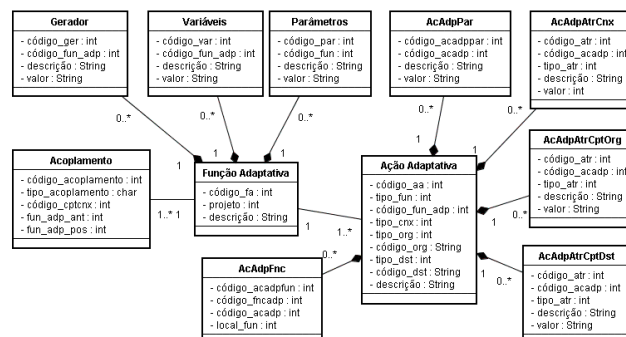


Figura 5. Camada de Especificação de Funções e Ações Adaptativa - CAMOLESI (1997).

Esta camada representa os elementos conceituais definidos em NETO (1991) para a concepção de um dispositivo adaptativo. Desta forma uma vez que um determinado projetista define um novo dispositivo na Camada de Configuração de Dispositivos, estará desta forma estendendo o referido dispositivo para suportar tecnologia adaptativa.

D. Camada de Representação de Memória

Por fim, a Camada de Memória (Figura 6) representa os estímulos de entrada (cadeia de entrada, estímulos externos oriundos de outros agentes, etc.) e o status (estados ou transições habilitadas a ocorrerem). Esta camada é de fundamental importância para a construção de ferramentas que darão suporte a simulações e verificações de aplicações projetadas utilizando-se de tecnologia adaptativa.

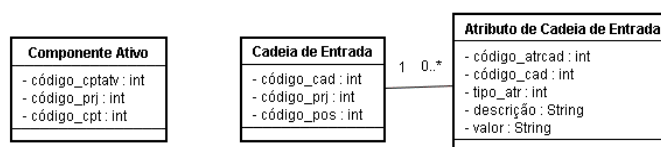


Figura 6. Camada de Representação de Memória - CAMOLESI (2007).

3. Framework Gráfico para criação de Tecnologias Adaptativas

Com o foco no estudo e aplicação dos conceitos de tecnologias adaptativas, foi definido um framework gráfico para auxiliar na obtenção de ferramentas gráficas que permitem a geração de dados no modelo lógico.

O framework proposto foi estruturado de forma a facilitar os estudos sobre tecnologias adaptativas a partir do Modelo Lógico já existente. O framework deve gerar dados para o modelo lógico de um dispositivo, porém ele terá de colocar as funções adaptativas com as informações necessárias para que permita a criação correta de diversos tipos de comportamento previsto. O framework foi desenvolvido com as regras de orientação a objetos, ou seja, foi estruturado em camadas.

A. Desenvolvimento

Para iniciar o desenvolvimento do framework gráfico foi necessário um estudo do banco de dados SQLSERVER da MICROSOFT. Tal banco se fez necessário para armazenar o banco de dados responsável por representar o Modelo Lógico de CAMOLESI (2007). Neste contexto, REIS (2012) apresenta o mapeamento dos conceitos do Modelo Lógico para o banco de dados MODELO_LOGICO.

Verificou-se com este estudo que para um bom funcionamento do banco de dados seria necessário uma instalação um pouco mais avançada do Gerenciador de Banco de Dados SQL Server para que outras linguagens de programação, se conectassem a ele, e não somente as linguagens da MICROSOFT.

Ao iniciar a instalação do produto o usuário deve seguir os assistentes de instalação, utilizando-se das opções padrões definidos para cada etapa, até que a interface de usuário abaixo seja apresentada.

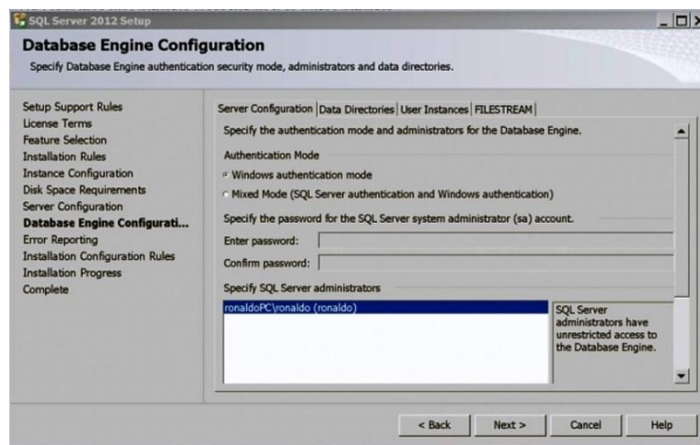


Figura 7. Interface para configuração do SQL Server.

Nesta etapa faz-se necessário a escolha da segunda opção “MixedMode”. Com isso, serão habilitados os campos de senha para o usuário “sa” (modo supervisor) do SQLSERVER. Escolha uma senha que seja de fácil memorização e siga em frente com o resto da instalação normalmente.

Finalizada esta etapa, o próximo passo é realizar a configuração no serviço do SQLSERVER. Para tal, abra o “Sql Server Configuration Manager”, que fica em “Iniciar-Todos os Programas-Microsoft Sql Server-Ferramentas de Configuração”, e será exibida a seguinte interface.

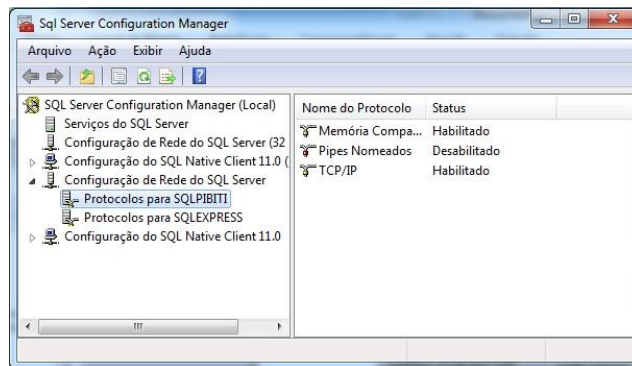


Figura 8. Sql Server Configuration Manager.

Depois de aberta a interface da Figura 8, selecione a opção “Protocolo para SQLPIBITI” (Instancia que foi criada na hora da instalação do banco de dados), em cima da opção TCP/IP clique com o botão direito em seguida escolha a opção “Propriedades”, será apresentado a interface da Figura 9.

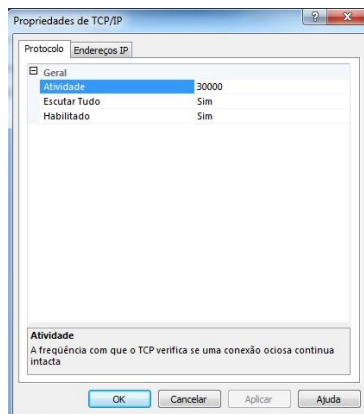


Figura 9. Configuração de Protocolo.

Caso a opção “Habilitado” esteja como “Não”, mude a para “Sim”, e selecione a opção “Endereços IP” (Figura 10).

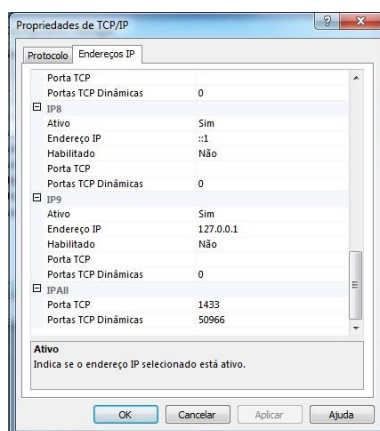


Figura 10. Configuração de Endereços IP.

Nesta tela, encontre a opção “Porta TCP” no final da barra de rolagem e insira o valor padrão do SQLSERVER que é “1433”. Em seguida, clique em ok e reinicie o computador para que o banco de dados adote todas as configurações feitas.

Pronto, agora o SQLSERVER está configurado para trabalhar em todas as linguagens que tenham suporte para sua instanciação.

Depois de apresentada a configuração do banco de dados, tem-se na sequência, o desenvolvimento do framework. Para tal, a linguagem de programação escolhida foi o JAVA. Para permitir o acesso do banco de dados em tal linguagem foi necessário obter, pela internet, os arquivos do Java Data Base Conector (JDBC) para o banco de dados SQLSERVER. O JDBC é uma biblioteca que permite a comunicação dos programas JAVA com um determinado banco de dados.

No framework, para ter acesso aos dados, faz-se necessário realizar a configuração do acesso ao banco de dados. Para que isso ocorra, foi codificada uma classe para armazenar as configurações. No trecho de Código 1 é apresentada a codificação JAVA da classe de configuração com o banco de dados.

Código 1. Classe Configuração

```
package util;

import java.io.Serializable;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class GerenciaConexoes implements Serializable {

    private static GerenciaConexoes gerenciaConexoes = null;
    private static Connection conn = null;
    private static PreparedStatement stmt = null;
    private static final String DRIVER =
"com.microsoft.sqlserver.jdbc.SQLServerDriver";
    //private static final String IP = "127.0.0.1";
    private static final String IP = "localhost:1433";
    private static final String PostgreSQL = "jdbc:sqlserver://";
    private static final String database = "MODELO_LOGICO";
    private static final String user = "sa";
    private static final String password = "123456";
    private static final long serialVersionUID = 1L;

    private GerenciaConexoes() {}
    public static GerenciaConexoes getInstance() {
        if (gerenciaConexoes == null) {
            gerenciaConexoes = new GerenciaConexoes();
        }
        return gerenciaConexoes;
    }
    public void conectar() {
        if (conn == null) {
            try {
                String dbPostgree = PostgreSQL + IP + ";databaseName=" +
database;
                Class.forName(DRIVER);
                conn = DriverManager.getConnection(dbPostgree, user,
password);
                conn.setAutoCommit(true);
                System.out.println("Conectou!!!");
            } catch (ClassNotFoundException | SQLException e) {
            }
        } else {
            System.out.println(conn + " já esta conectado!!!");
        }
    }
    public void desconectar() {
        if (conn != null) {
            try {
                if (stmt != null) {
                    stmt.close();
                }
                conn.close();
                conn = null;
                stmt = null;
                System.out.println("Desconectou!!!");
            } catch (SQLException e) {
            }
        } else {
            System.out.println(conn + " esta desconectado!!!");
        }
    }
    public PreparedStatement getPreparedStatement(String statement) {
        try {
            if (conn == null) {
                conectar();
            }
            stmt = conn.prepareStatement(statement);
        } catch (SQLException e) {
        }
        return stmt;
    }
}
```

O trecho do código (Código 1) apresentado acima, representa a configuração do banco de dados que será utilizado no projeto de construção do framework Gráfico. Esta classe também garante que somente uma instância da conexão com o banco seja aberta, este tipo de programação é conhecido como padrão *Singleton*.

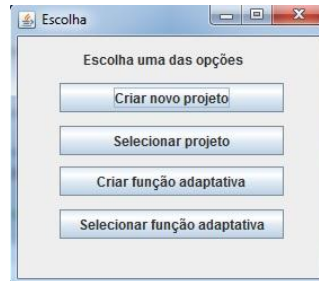


Figura 11. Interface Inicial do Framework Gráfico.

A Figura 11 apresenta a interface inicial do Framework Gráfico. Nesta interface, podem-se escolher quatro opções: Criar novo projeto, Selecionar projeto, Criar função adaptativa e Selecionar Função adaptativa.

Ao selecionar a primeira opção, o usuário é redirecionado para a interface ilustrada na Figura 12.

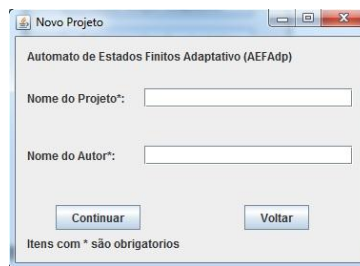


Figura 12. Criação de Novo Projeto.

Ao definir um Novo Projeto (Figura 12), o usuário deverá informar o nome escolhido para o seu projeto, além do seu próprio nome. Caso ele selecione o botão “Voltar”, a qualquer momento, durante a utilização, o usuário será redirecionado para a interface inicial ilustrada na Figura 11. Caso o usuário escolha a opção “Continuar” a interface responsável pela edição do projeto (Figura 13) será apresentada.

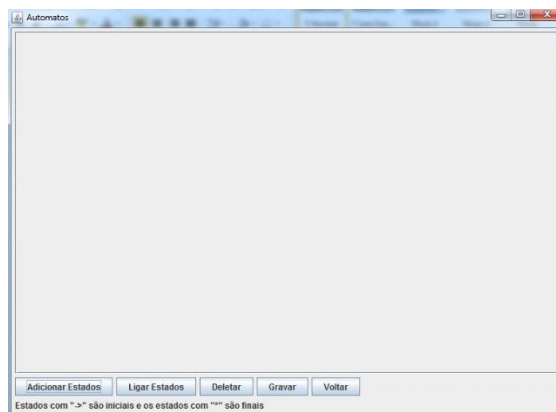


Figura 13. Interface gráfica para criação de projeto de autômatos finitos determinísticos.

Caso o usuário selecione a opção “Adicionar Estados” a interface da Figura 14 é aberta. Ao adicionar um novo componente, o usuário poderá configurar algumas propriedades que permitem, por exemplo, dizer se um estado é inicial ou final; ou até mesmo os dois, e terá que dar um nome a este componente.

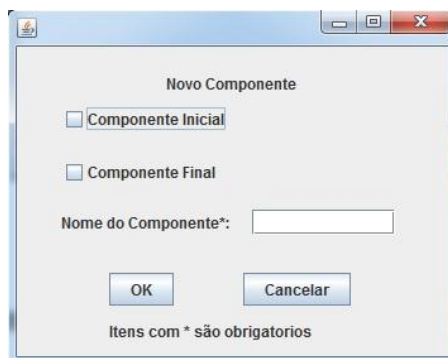


Figura 14. Criação de Estados.

Se o usuário escolher a opção “Ligar Estados”, a interface ilustrada na Figura 15, somente será aberta se tiver no mínimo um estado criado. Tal interface permite ao usuário escolher os estados que ele deseja ligar e informar um valor para a transição que será criada.

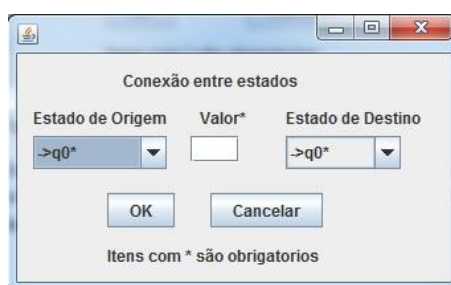


Figura 15. Interface Ligar Estados.

Para o usuário apagar um estado ou transição, basta selecionar o estado e/ou transição desejado com um clique do mouse, e em seguida escolher a opção “Apagar”.

A opção “Selecionar projeto” abre uma interface onde são apresentados todos os projetos existentes em uma caixa de seleção. O usuário pode selecionar um dos projetos disponíveis, e ao pressionar botão “Ok” o projeto é carregado. Caso contrário, se o usuário pressionar o botão “Voltar” nada acontece, e ele retorna a interface da Figura 13.

Com o projeto selecionado, o usuário é redirecionado para a interface representada na Figura 13 com o projeto carregado, após isso os procedimentos serão os mesmos.

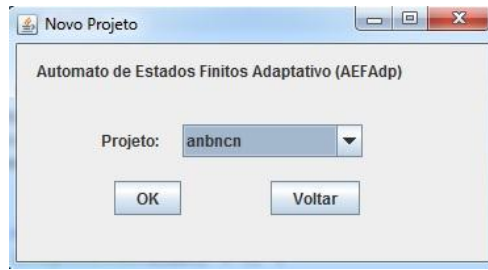


Figura 16. Seleção de Projetos.

Por fim, na interface descrita na Figura 17 é possível criar uma função adaptativa. As funções são criadas uma por vez, e podem ter mais de uma função adaptativa por projeto. Ao se definir uma ação adaptativa é possível escolher o seu tipo que pode ser “Busca”, “Remoção” e/ou “Adição”.

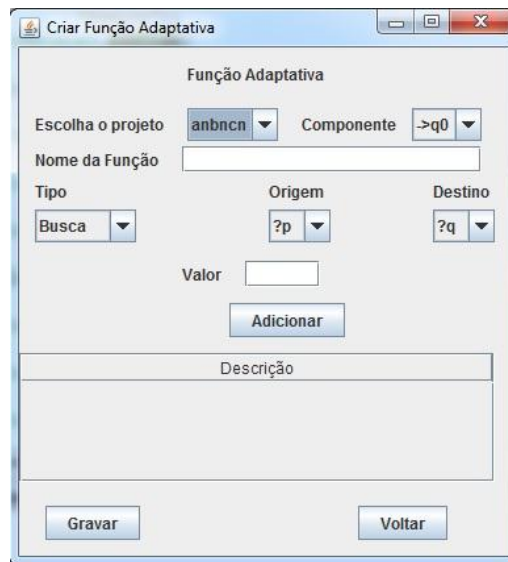


Figura 17. Criação da função adaptativa.

4. Conclusão

Neste trabalho foram apresentados os conceitos de Tecnologia Adaptativa. A partir destes conceitos, foi realizado o desenvolvimento de um framework gráfico para auxiliar a construção de ferramentas com dispositivos adaptativos.

O framework gráfico desenvolvido foi instanciado para a criação de autômatos finitos adaptativos e demonstrou-se eficiente para a sua realização. Com base na ferramenta de criação obtida, foram realizados testes e verificou-se que tal ferramenta pode auxiliar os projetistas de aplicações no desenvolvimento de suas tarefas.

Com a concretização desse trabalho obteve-se uma ferramenta que permite a um especialista, em certo dispositivo dirigido por regra, realizar o mapeamento deste para uma meta estrutura que represente sua extensão adaptativa e a sua forma de operação. Além do auxílio aos especialistas em extensão de dispositivos adaptativos, a ferramenta também auxilia os projetistas na realização do trabalho de especificação de suas aplicações.

Fundamentado na pesquisa desenvolvida, várias vertentes para trabalhos futuros podem ser identificadas. Como possíveis trabalhos futuros, pode-se citar:

- Utilização do framework gráfico desenvolvido para instanciação de outros dispositivos adaptativos, com o intuito de avaliar a consistência da implementação realizada;
- Desenvolvimento de uma ferramenta textual, acoplada ao framework para auxiliar a definição de novos dispositivos adaptativos e o projeto de aplicações com base nos dispositivos obtidos;
- Construção de uma ferramenta que permita a tradução automática das especificações descritas no modelo lógico para uma linguagem de programação.

5. Referências

ALMEIDA, J.R. STAD. **Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos**. Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, 1995.

CAMOLESI, A.R. e NETO, J.J. **Modelagem Adaptativa de Aplicações Complexas**. XXX Conferência Latino americana de Informática - CLEI'04. Arequipa - Peru, Setiembre 27 - Octubre 1, 2004a.

CAMOLESI, A.R. **Proposta de um gerador de ambientes para a modelagem de aplicações usando Tecnologia Adaptativa**. Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, 2007.

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, Springer-Verlag, Vol.2494, pp. 234-250, Pretoria, South Africa, July 23-25, 2001.

_____. **Contribuições à metodologia de construção de compiladores.** Tese de Livre Docência, USP, São Paulo, 1993.

_____.; KOMATSU, W. **Compilador de Gramáticas Descritas na Notação de Wirth Modificada.** Anais EPUSP - Engenharia de Eletricidade - série B, vol. 1, pp. 477-517, São Paulo, 1988.

PISTORI, H. **Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações.** Tese de Doutorado, USP, São Paulo, 2003.

REIS, F.M.B. **Framework para Simulação e verificação de Aplicações Adaptativas.** Trabalho de Iniciação Científica, Projeto PIBIC, FEMA, 2012.

Documentação do SQLSERVER. Disponível em <<http://www.microsoft.com/pt-br/download/confirmation.aspx?id=347>> Acesso em 15 fev. 2014.

Documentação do JAVA. Disponível em <<http://www.oracle.com/technetwork/pt/java/javase/documentation/index.html>> Acesso em 23 jul. 2014.