O uso de conceitos de injeção de código no desenvolvimento de aplicações adaptativas

Carlos Roberto ROSSINI Junior, Almir Rogério CAMOLESI

junior_rossini9@hotmail.com, camolesi@femanet.com.br

RESUMO: A injeção de código é uma técnica que pretende inserir dependência em uma classe por meio de código externo. O presente trabalho descreve a utilização da injeção de código junto com a tecnologia adaptativa para o desenvolvimento de aplicações complexas. Para ilustrar o estudo, foi realizado a criação de um jogo de dominó adaptativo, o qual são introduzido os conceitos de injeção de código e da tecnologia adaptativa em sua criação, como forma de demonstrar o estudo de caso.

PALAVRAS-CHAVE: Injeção de Código; Tecnologia Adaptativa; Adaptativo; Orientação a Objetos, Jogos.

ABSTRACT: The code injection is a technique who intends inject a dependecy in a class per way of code external. This work describe an utilization of dependecy injection with the adaptative technology for the development of complex application complex. To illustrate the study, was created a game of adaptive domino. This game uses the concepts of code injection and of adaptive technology in its creation, as a way to demonstrate the study of case.

KEYWORDS: Code Injection; Adaptive technology; Adaptive; Object-Oriented; Games.

1.Introdução

Quando se fala em sistemas, logo se vem a mente sistemas de computadores que automatizam tarefas diárias de uma empresa, mas sua atuação é mais ampla, para acompanharmos esta evolução é necessário entender a evolução do desenvolvimento cientifico e da inteligência humana. Além do desenvolvimento cientifico, parâmetros contidos em problemas crescem de uma forma assustadora e cada vez mais complexa.

Impossibilitando que apenas uma área compreenda todas as informações e explicações destes fenômenos existentes.

Sistemas devem ser construídos para atender as demandas das empresas. Neste contexto os desenvolvedores de softwares a cada dia mais têm se deparado com aplicações complexas e que tem que ser adaptadas às exigências mais fundamentais segundo as características de cada empresa.

Aplicações complexas são caracterizadas por componentes e aspectos cuja estrutura e comportamento, comumente, podem modificar-se (Camolesi, 2004a). Tais aplicações possuem um comportamento inicial definido por um conjunto de ações que desempenham suas funções elementares, e durante a execução podem ter o seu comportamento modificado 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 na modelagem 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).

Estudos já foram realizados com o objetivo de implementação de aplicações adaptativas. CASACHI (2011) realizou um estudo que teve por objetivo apresentar o uso da Programação Orientada a Aspectos (KICZALES, 1997) no desenvolvimento de aplicações que utilizam os conceitos de Tecnologia Adaptativa. Tal estudo permitiu a implantação da Tecnologia Adaptativa de uma forma fácil e segura, além de permitir que novas funcionalidades (aspectos) sejam adicionadas ao software sem a necessidade de modificar o código fonte já produzido. O programador deve apenas acrescentar novos aspectos ao software e o mesmo se adapta ao código já existente.

O objetivo desse trabalho foi estudar o conceito de desenvolvimento de aplicações complexas com foco no uso de Injeção de Código para implementação dos conceitos de de Tecnologia Adaptativa com foco na resolução de tais problemas. Por fim, para ilustrar os estudos realizados foi desenvolvido um jogo, com base nas regras de um jogo de Dominó Adaptativo para demonstrar os conceitos estudados.

Este trabalho está organizado da seguinte forma, no Capítulo 2 são apresentados os principais conceitos de Tecnologia Adaptativa. A seguir, no Capítulo 3 são descritas as

técnicas para o uso de injeção de código. No Capítulo 4 é apresentado o estudo de caso fundamentado nos conceitos estudados e no jogo de Dominó Adaptativo. Por fim, no Capítulo 5 são realizadas algumas conclusões e trabalhos futuros.

2. Tecnologia Adaptativa

Um dos primeiros trabalhos relacionado ao estudo da Tecnologia Adaptativa é o apresentado por NETO e MAGALHÃES (1981), que fornece uma visão de métodos de análise sintática e de geração de reconhecedores sintáticos. Este trabalho foi resultado de um primeiro esforço em busca da inclusão dos conceitos de mecanismos adaptativos em um sistema para apoio à construção de compiladores.

Na sequencia, foram realizados estudos com objetivo de resolução de problemas relacionados a compilação e em (NETO, 1993) foi apresentado o autômato e o transdutor adaptativo como dispositivos de reconhecimento e transdução sintática.

Com base no trabalho de Neto (1993), foram realizados outros trabalhos nos quais foi aplicada a tecnologia adaptativa no projeto de sistemas reativos. Almeida (1995) apresentou uma evolução da notação de Statechart (HAREL et al., 1987), na qual foram acrescentadas características provenientes da teoria de Autômatos Adaptativos.

O Statechart Adaptativo tem capacidade de modificar sua configuração em resposta às entradas impostas ao sistema por ele representado. O trabalho realizado por Almeida (1995) permitiu também a implementação de uma ferramenta de análise e desenvolvimento de aplicações valendo-se de Statechart Adaptativo. Essa ferramenta, intitulada STAD (STAD, 2005), constitui-se de um editor e de um simulador de sistemas que utiliza a notação desenvolvida. Além de ter propiciado uma visão prática da teoria do uso de tecnologia adaptativa, a ferramenta STAD comprovou a sua viabilidade, proporcionando assim uma forma bastante útil de difusão desses conceitos.

Um estudo que teve por objetivo melhorar a especificação de um conjunto de sistemas reativos complexos e sincronizados entre si pode ser encontrado em (NOVAES, 1997). Um dos resultados desse trabalho foi o desenvolvimento de um formalismo, o Stad-Sinc, que se fundamenta na junção das notações de Rede de Petri, de Statechart convencional e de Statechart Adaptativo. O novo formalismo criado permite representar por meio de diagramas os mecanismos de sincronização existentes nas aplicações projetadas.

Um Ambiente de Desenvolvimento de Reconhecedores Sintáticos Baseado em Autômatos Adaptativos foi descrito em (PEREIRA, 1997). Este trabalho introduziu uma ferramenta de auxílio ao desenvolvimento de reconhecedores sintáticos denominada

Reconhecedor Sintático para Window (RSW) (PEREIRA, NETO, 1999). A ferramenta RSW proporciona aos seus usuários um ambiente integrado, e os reconhecedores sintáticos reconhecidos e gerados pela ferramenta são baseados na teoria de autômato de estados finitos, autômatos de pilha estruturados e nos autômatos adaptativos. A possibilidade de implementação de reconhecedores baseados em Autômatos Adaptativos (NETO, 1993) constitui uma das importantes metas alcançadas pela ferramenta RSW, comprovando sua viabilidade prática.

Um formalismo dual ao autômato, o qual visava a facilitar o desenvolvimento de linguagens complexas ou outras aplicações que necessitassem especificar linguagens dependentes de contexto na forma de gramáticas foi denominado como Gramática Adaptativa (IWAI 2000). Tal formalismo possui como característica principal a capacidade de se alterar a medida que vai sendo feita a geração da sentença pertencente à linguagem que é representada pela gramática adaptativa.

Rocha (2000) elaborou um estudo que visa o desenvolvimento de um método de construção de modelos e resolução de problemas complexos utilizando-se dispositivos adaptativos. Para tal, foram realizados estudos comparativos entre autômatos adaptativos, redes neurais, algoritmos genéticos e agentes, extraindo destes dispositivos, características que permitiram a composição do modelo denominado Busca de Soluções por Máquina Adaptável (BSMA) (ROCHA; NETO, 2000).

Em relação ao projeto de linguagens, Neto e Silva (2005) apresentaram uma estrutura adaptativa para design de linguagens de especificação de software. Neste trabalho foram apresentadas algumas características adaptativas presentes em linguagens de especificação, e também foi descrita uma estratégia para estender a especificação de linguagens de programação. Um exemplo simples também foi apresentado para demonstrar a estrutura proposta. Neto (2001) busca características comuns presentes nos dispositivos adaptativos dirigidos por regras, o que permite a um especialista estender um dispositivo dirigido por regras para suportar tecnologia adaptativa. Com base nisso Camolesi e Neto (2003) apresentaram uma proposta de extensão do dispositivo Interaction System Design Language (ISDL) (QUARTEL, 1997) definindo então o dispositivo ISDLAdp. O estudo também demonstra a aplicação da tecnologia adaptativa na modelagem de aplicações hipermídia. Em estudo posterior (CAMOLESI; NETO, 2004a), foi apresentada uma formulação completa do ISDLAdp e a aplicação de sua linguagem na modelagem de aplicações complexas

No trabalho desenvolvido por Camolesi e Neto (2004b) foi descrita a extensão de Rede de Petri Adaptativa (RPAdp) conforme a formulação de (NETO, 2001). Neste estudo, propunha-se a definição de uma estrutura de dados para a representação da Rede de Petri Adaptativa. As etapas de extensão de um dispositivo adaptativo e uma estrutura de dados geral para a representação de dispositivos adaptativos dirigidos por regras pode ser encontrado em (CAMOLESI, 2005).

No trabalho desenvolvido por Pistori (2003) foi empregada a Tecnologia Adaptativa para facilitar a interação de jogadores que possuem deficiência motora. Foi desenvolvido um jogo tradicional, o "Jogo da Velha", que permite ao jogador escolher o local onde jogará utilizando para isto a direção do olhar por meio de uma câmera. Outros trabalhos relacionados ao uso da Tecnologia Adaptativa empregada a jogos também foram realizados, porém estes foram apenas exemplos em sala de aula e não foram realizadas publicações sobre os mesmos. Neste sentido espera que o desenvolvimento desta pesquisa possa contribuir com publicações para esta área. O principal diferencial na Tecnologia Adaptativa é a forma razoavelmente simples que podemos transformar teorias já existentes, bem como fazer um reaproveitamento e estruturação destas para melhorar sua capacidade de respostas e interação.

O desenvolvimento níveis de dificuldades com Tecnologia Adaptativas faz com que seja facilitada a interação com o usuário, sem a necessidade de uma base de dados contemplando todas as possíveis reações de um jogador, porque cada regra inicial pode ser modificada de acordo com o ambiente ao qual está sendo trabalhado de forma que não necessita de alguém programando isso a cada nova ocorrência encontrada.

Este conceito de auto-modificação da Tecnologia Adaptativa, torna a Inteligência Artificial (muito encontrada em jogos que exigem determinados tipos de respostas ao usuário), possa ser trabalhada de forma mais simples, e eficiente desde que seja feito de forma correta o seu desenvolvimento, seguindo passos que por mais simples que pareçam, mostrem uma complexidade no que diz a atenção dispensada para não ter um sistema com falhas futuras.

3. Injeção de Código

No desenvolvimento de sistemas comerciais, o analista vai programá-lo para solucionar problemas, independentemente da quantidade de problemas. Esse sistema jamais vai modificar seu comportamento ou estrutura para solucionar um problema, pois ele irá realizar tudo o que foi programado sem fazer alterações em si mesmo, não permitindo que o sistema consiga se autoanalisar conforme novas necessidades que aparecerem

Porém ao utilizar Injeção de Código é possível o sistema receber códigos durante a execução do programa, os quais somente poderiam ser resolvidos com adição de novos comandos na programação. Técnicas de injeção de código são populares em sistema maliciosos, utilizados geralmente por hackers, para obter informações, senhas acesso ou privilégios não autorizado a um sistema. A injeção de código, inicialmente, foi usada de forma maldosa para muitos propósitos. Atualmente, verifica-se que tal conceito, uma vez utilizado de forma consciente e controlada pode auxiliar na resolução de problemas complexos de nosso cotidiano.

3.1. Padrões de Projetos

O padrão de projetos são técnicas utilizadas para se resolver um problema o qual pode ter ocorrido em várias situações diferente.

É muito recorrente que no desenvolvimento de um software, um desenvolvedor travará em algum determinado problema, é muito provável com que esse problema tenha acontecido com diversos outros desenvolvedores, devido a isto, os padrões de projetos vem cada dia mais ganhando um grande espaço no campo científico, pois ele traz soluções testadas e de alto nível para se resolver determinado problema.

3.2. Acoplamento e Coesão

Na produção de sistemas, um fator muito importante é a sua manutenibilidade. É de extrema importância que o sistema tenha uma fácil manutenção, pois erros poderão aparecer após a sua implementação, além de ser preciso ter que adicionar novas funcionalidades.

O acoplamento está extremamente ligado com a manutenibilidade do sistema, ele seria o nível em que as classes estão ligadas uma com as outras.

Um acoplamento forte significaria que as classes possuem uma forte dependência uma com a outra, isso faz com que o sistema possua uma manutenção difícil, já que ao se alterar uma classe com acoplamento forte, provavelmente terá que fazer alterações em outras classes para se realizar as mudanças necessárias.

Já uma classe que possui um acoplamento fraco significa que as suas possuem uma baixa dependência uma da outra, isso faz com que a manutenção seja facilitada, devido à possibilidade de se poder fazer alterações nas classes, sem precisar ter que realizar mudanças em outras classes.

A coesão mede o nível de entendimento de uma classe, ela segue o principio da responsabilidade única, o qual diz que uma classe deve ser apenas responsável por uma coisa. Diante dessa consideração, uma classe que possui um baixo acoplamento, vai possuir uma alta coesão, já que ela não possui dependência de uma outra classe.

Tendo em vista os aspectos observados, pode-se concluir que na construção de um sistema, é importante buscar um acoplamento fraco, devido ao fato da fácil manutenção e a possibilidade de se fazer teste unitários.

3.3. Inversão de Controle

A inversão de controle é um padrão de projetos que visa diminuir o acoplamento de uma classe e aumentar a sua coesão.

O principio básico da inversão de controle é fazer com que os controles de uma classe seja invertidos, no caso, ao se realizar a inversão de controle em uma classe, faz com que as responsabilidade que essa classe possui seja retirada dela, desta forma, a classe não precisará ser responsável por declarar suas dependência.

Muitos frameworks utilizam do padrão de inversão de controle, devido ao fato das classes poderem ser reutilizáveis e de ser possível de criar novas funcionalidades sem precisar ter que alterar as já existentes.

3.4. Injeção de Dependência

Uma das maneiras de se utilizar do padrão de inversão de controle é utilizando-se a injeção de dependência, a qual faz com que as dependências de uma classe seja providas por um código externo a ela.

Podemos supor uma classe Aluno, o qual precisa receber uma lista de objeto da classe Nota para realizar determinado método de consulta.

Código 1. Exemplo da classe aluno.

```
public class Aluno
{
    private List<Nota> lstNota;
    public void receberNotas()
    {
        Nota nota = new Nota();
        lstNota = nota.notasAluno(this);
    }
}
```

Está classe não possui nenhum erro, ela funcionaria normalmente, porém ela possui uma dependência de um objeto de uma outra classe, isso faz com que ela tenha um alto

acoplamento. Para se retirar a responsabilidade da classe Aluno de criar um objeto da classe Nota.

Vão ser exemplificados neste artigo a Injeção de Dependência via Construtor e a via Setter.

Código 2. Injeção de Dependência via Construtor.

```
public class Aluno
{
    private List<Nota> lstNota;
    public Aluno(List<Nota> lstNota)
    {
        this.lstNota = lstNota;
    }
}
```

No devido caso, a classe Aluno não irá ser responsável de instanciar um objeto do tipo Nota, já que o objeto nota irá ser referenciando através de um construtor, fazendo com que o objeto nota seja uma dependência que foi inserida na classe o qual foi instanciado fora dela.

Código 3. Injeção de Dependência via Setter Injection

```
public class Aluno
{
    public List<Nota> lstNota
    {
        get
        {
            return lstNota;
        }
        set
        {
            lstNota = value;
        }
    }
}
```

Devido a classe ter sido escrito em C#, foi possível injetar a dependência na própria propriedade do atributo, o qual faz com que a dependência teria que ser declarada fora do contexto da classe Aluno.

4. Estudo de Caso

Nesta seção inicialmente será apresentado os conceitos relacionado as regras de um jogo de Dominó Adaptativo. Tal jogo foi escolhido por ser um jogo que não demanda de muito tempo para o desenvolvimento da interface gráfica para o mesmo, sobrando assim, mais tempo para o pesquisador avaliar e aplicar os conceitos estudados no desenvolvimento do mesmo.

4.1. Dominó Adaptativo

O dominó adaptativo é uma ideia apresentada por (NETO, 2007), o qual introduz fundamentos da tecnologia adaptativa dentro de um jogo de dominó.

Em um jogo de dominó comum, para se realizar uma jogada, é necessário com que a pedra a ser jogada, tenha ao menos uma das pontas com valores iguais a uma das duas extremidades das pontas do tabuleiro. Na proposta do dominó adaptativo, o jogador tem a possibilidade de alterar as regras de encaixe do jogo em tempo de execução de partida, em função disto, ao se fazer uma jogada, a pedra não irá necessariamente ter que possuir um dos lados iguais a alguma das extremidades das pontas do tabuleiro, devido as regras de encaixe terem sido modificadas.

No início de uma partida, são introduzidas aleatoriamente dentro do jogo, algumas pedras adaptativas, essas pedras permitem com que as regras de encaixe sejam alteradas na execução da jogada, uma vez a regra modificada, essa modificação permanecerá até o fim do jogo, ao menos que alguma outra pedra adaptativa seja jogada e que ela tenha influência na regra modificada.

4.2. Arquitetura do Sistema

Primeiramente foi desenvolvido um jogo de dominó tradicional (não adaptativo). Com base no jogo produzido foi implementado os conceitos da Tecnologia Adaptativa. Após a finalização da base inicial do jogo, foi realizada a implementação dos métodos que permitem que uma função adaptativo modifique em tempo de execução as regras do jogo de dominó, por fim, para realizar o desenvolvimento dos conceitos de programa adaptativo foi utilizadas o foi-se utilizado o padrão de injeção de dependências.

O Dominó Adaptativo teve suas classes modeladas pela a ferramenta Astash Communty[1] e foi desenvolvido na plataforma .NET[2] utilizando Visual Studio Community 2015[3].

[1] http://www.astah.net/
[2] https://www.msdn.microsoft.com/
[3] https://www.visualstudio.com/

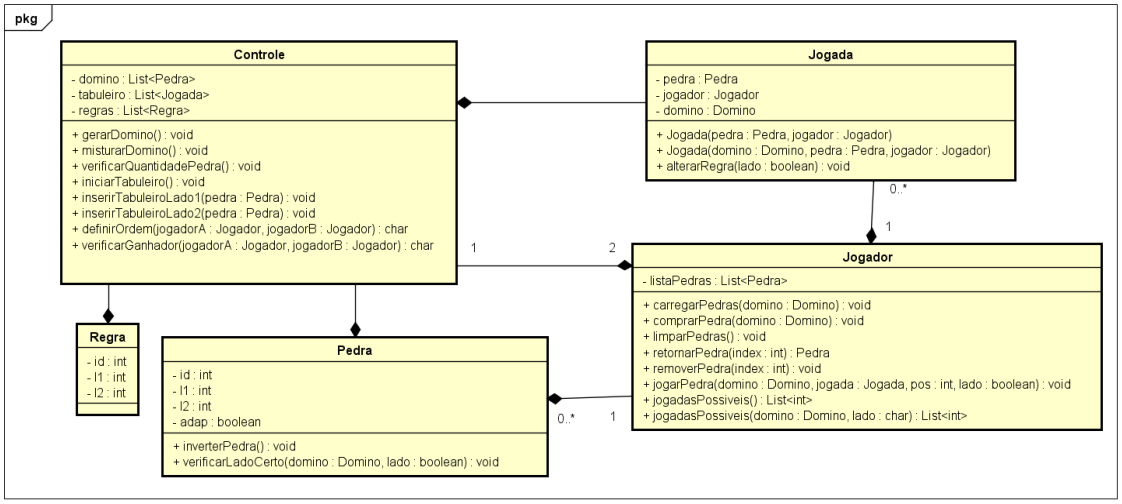

Figura 1. Modelagem das classes

A modelagem de classe foi estruturada de forma que pudesse criar as regras de encaixe, onde as jogadas só poderiam acontecer se estivesse de acordo com a regra vigente. Essas regras são definidas na classe Dominó no método gerarDominó(). Tal método tem por função gerar as 7 regras padrões e as 28 pedras do jogo.

Com as regras e pedras geradas, é preciso embaralhar essas pedras e sortear qual delas vão ser pedras adaptativas, para o jogo não se tornar vicioso, utiliza-se de uma função aleatório com base nos valores da data e hora do embaralhamento como um índice aleatório , conforme demonstrado no Código 4.

Código 4. Método para ser realizar o embaralhamento do dominó.

```
public void misturarDomino()

    int seed = DateTime.Now.Millisecond*DateTime.Now.Year*DateTime.Now.Second;
    Random rand = new Random(seed);
    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < domino.Count; j++)
        {
              int iRand = rand.Next() % 28;
              Pedra aux = domino[j];
              domino[j] = domino[iRand];
              domino[iRand] = aux;
        }
    }
    for (int i = 0; i < 4; i++)
    {
        int iRand = rand.Next() % 28;
          domino[iRand].adap = true;
    }
}
```

O método jogadasPossiveis() é um método polimórfico que verifica as possíveis jogadas do jogador, conforme as pedras que o jogador possua na mão. Este é polimórfico devido ao fato de que caso o tabuleiro esteja vazio, a verificação das jogadas possíveis é diferente da verificação de quando estiver com uma ou mais pedras no tabuleiro.

Código 5. Método verificador das possíveis jogadas.

```
public List<int> jogadasPossiveis(Domino domino, char lado)
{
  List<int> lstIndices = new List<int>();
  int pl;
  if (lado == 'E')
     pl = domino.tabuleiro[0].pedra.l1;
  else
     pl = domino.tabuleiro[domino.tabuleiro.Count - 1].pedra.l2;
     Pedra pedra = new Pedra();
     int dest;
     for (int i = 0; i < listaPedras.Count; i++)
     {
         pedra = retornarPedra(i);
         dest = domino.regras.First(r => r.l1 == pedra.l1).l2;
         if (dest == pl)
             lstIndices.Add(i);
         else
              {
                 dest = domino.regras.First(r => r.l1 == pedra.l2).l2;
                 if (dest == pl)
                        lstIndices.Add(i);
              }
            }
            return lstIndices;
        }
```

No código 5 demonstra a forma em que foram verificadas as peças válida para realizar a jogada. As verificações são feitas utilizando as regras de encaixe já definidas anteriormente, caso houver alguma alteração de regra, ela já vai estar em vigor .Após as verificações serem feitas, o método irá retornar uma lista dos índices das peças, este índice será usado na hora que o jogador for realizar a jogada, caso o índice da peça condiz com algum índice da lista em que o método retorno, essa jogada poderá ser realizada.

Para ser possível implementar a tecnologia adaptativa, foi utilizado dos conceitos do padrão de injeção de dependência, desta forma, foi possível fazer a implementação da tecnologia adaptativa sem precisar ter que alterar outras classes do código.

Era necessário aplicar a tecnologia adaptativa na classe Jogada, pois as regras de encaixe do jogo só iriam sofrer alteração quando houvesse uma jogada adaptativa, no entanto, as regras de encaixes eram guardadas por um objeto da classe Dominó, devido a isto, a classe Jogada iria precisar instanciar um objeto da classe Dominó. A fim de evitar que a classe Jogada tivesse a responsabilidade de instanciar um objeto de uma outra classe, foi utilizado a injeção de dependência via construtor, deste modo, a classe irá deixar de ser responsável de criar uma instância do objeto dominó, o qual foi passado a responsabilidade para o construtor, que recebe uma instância da classe Dominó.

Código 6. Construtor da classe Jogada.

```
    public Jogada(Domino domino, Pedra pedra, Jogador jogador)
    {
          this.domino = domino;
          this.pedra = pedra;
          this.jogador = jogador;
    }
```

Com o objetivo de demonstrar o uso da tecnologia adaptativa para alterar as regras de encaixe do jogo de dominó é apresentado no Código 7, o qual é um método que é utilizado quando alguma pedra adaptativa é usada no jogo. Devido ao fato que este método depende do lado em que a peça foi jogada, o código precisou ser dividido em duas partes para se fazer as devidas verificações, no caso, o código apresentado corresponde a peça que foi jogada no inicio do tabuleiro.

Código 7. Método Adaptativo.

```
public void alterarRegra(bool lado)
...
for (int i = 0; i < domino.regras.Count; i++)
 {
if (domino.regras[i].l1 == pedra.l2)
        {
              domino.regras[i].l2 = domino.tabuleiro[0].pedra.l1;
              i = domino.regras.Count;
        }
  }
for (int i = 0; i < domino.regras.Count; i++)
  {
     if (domino.regras[i].l1 == domino.tabuleiro[0].pedra.l1)
                    {
                        domino.regras[i].l2 = pedra.l2;
                        i = domino.regras.Count;
                    }
                }
...
```

4.3. Execução da Aplicação

Para fim de experimento da aplicação, uma interface com poucos componentes foi criada.

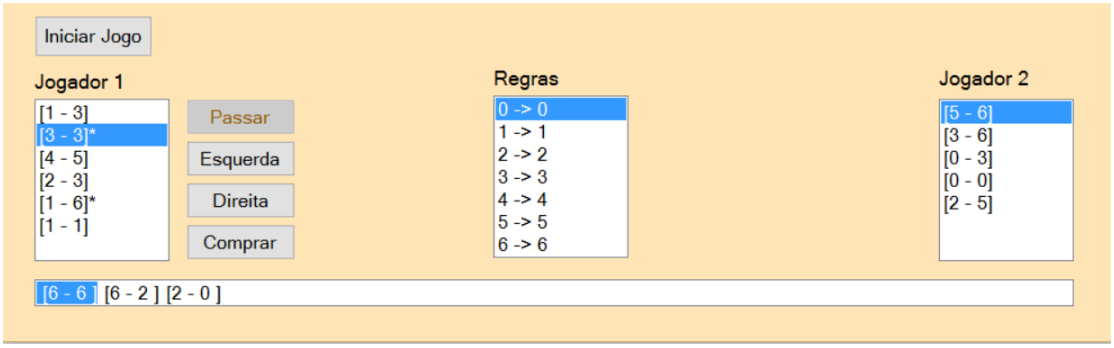

Figura 2. Tabuleiro Jogo de Dominó Adaptativo

Para devidos testes, as peças do computador ficaram visíveis, todas as suas jogadas são feitas automaticamente.

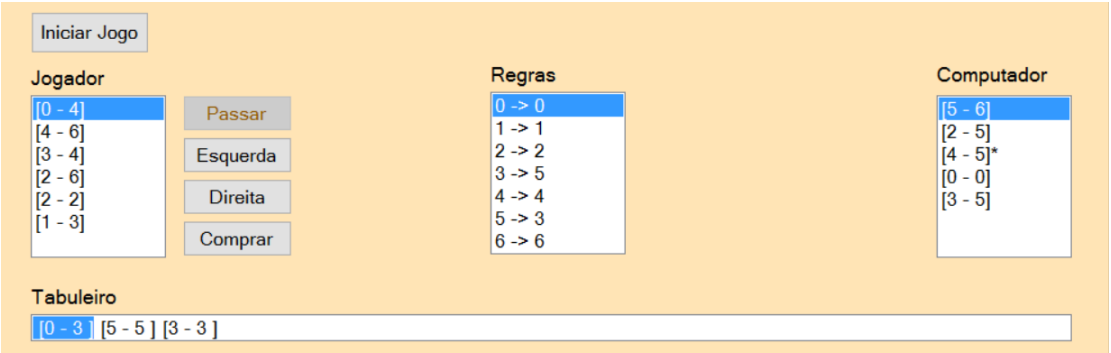

Figura 3. Realização da Jogada Adaptativa

As pedras que possuírem um asterisco, tem o significado que ela é uma peça adaptativa, com isso, o jogador tem a possibilidade de realizar uma jogada com ela desde o inicio do tabuleiro, tanto como no fim do tabuleiro, independentemente se é uma combinação válida ou não, caso a combinação não for válida, o método adaptativo é utilizando e modifica as regras do jogo, as quais estão em vigor até o fim do jogo.

A figura 3 demonstra como ficam as regras de encaixe após acontecer uma jogada adaptativa, no caso referente, as pedra que tiverem lado com valor igual a três só fará combinação com pedras que tiverem valor igual a 5. Após a jogada adaptativa, o computador já identificou as novas regras e realizou uma nova jogada utilizando delas.

O jogo acaba quando as peças do jogador ou do computador acabarem, caso não haver mais pedras para comprar e não existir mais nenhuma possibilidade de jogada, é realizado uma soma utilizando como base os valores das pedras, o jogador que tiver mais pontos será o vencedor.

5. **Conclusões**

Considerando-se os dados apresentado sobre a tecnologia adaptativa e a injeção de dependência, é possível constatar que a utilização de injeção de dependência pode contribuir positivamente na implementação da tecnologia adaptativa em um sistema.

Conforme foi demonstrado no Dominó Adaptativo, houve uma fácil integração da tecnologia adaptativa, devido ao fato do uso da injeção de dependência na construção do sistema.

Existem diversos outros tipos de padrão de projetos além do padrão de injeção de dependência, fica como trabalhos futuros, o estudo da aplicações destes outros padrões de projeto dentro de um contexto que utiliza a tecnologia adaptativa, o desenvolvimento

de uma interface com mais recursos para o Dominó Adaptativo e a implementação de um novo modo de jogo, o qual irá possibilitar jogos online por redes.

REFERENCIA BIBLIOGRÁFICA

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. **Modeling a tool for the generation of programming environments for adaptive formalism.** International Conference on Adaptive and Natural Computing Algorithms (ICANNGA 2005), Springer Verlag editor, pp. 341-344, Coimbra University, Coimbra, Portugal, 21 - 23 march 2005.

CAMOLESI, A.R. **Uma metodologia para o Design de Serviços de TV-Interativa**. Dissertação de Mestrado, PPG-CC, UFSCar, 2000.

CAMOLESI, A.R.; NETO, J.J. **An adaptive model for specification of distributed systems.** IX Congreso Argentino de Ciencias de la Computación, La Plata, Argentina, 6-10 de Outubro, 2003.

CAMOLESI, A.R.; NETO, J.J. **Modelagem Adaptativa de Aplicções Complexas**. XXX Conferencia Latinoamericana de Informática - CLEI'04. Arequipa - Peru, Setiembre 27 - Octubre 1, 2004a.

CAMOLESI, A.R.; NETO, J.J. **Modelagem AMBER$_{Adp}$ de um Ambiente para Gerenciamento de Ensino a Distância.** Anais do XIII Simpósio Brasileiro de Informática na Educação - SBIE 2002, pp. 401-409, São Leopoldo, RS, Novembro 12-14, 2002.

CAMOLESI, A.R.; NETO, J.J. **Representação Intermediária para Dispositivos Adaptativos Dirigidos por Regras**. 3rd International Information and Telecommunication Technologies Symposium, UFSCar, São Carlos, Brasil, 2004b.

CASACHI, R. A. **Aplicação do Paradigma de Programação Orientada a Aspectos no Desenvolvimento de Software.** Trabalho de Conclusão de Curso, Bacharelado em Ciência da Computação, FEMA-IMESA, 2011.

FOWLER, Martin. **Inversion of Control Containers and the Dependency Injection pattern.** Disponível em: <http://martinfowler.com/articles/injection.html>. Acesso em 20 set. 2016.

FREEMAN Eric.; FREEMAN Elizabeth. **Use a Cabeça! Padrão de Projetos.** Rio de Janeiro: Alta Books. 496 p.

HAREL D. et al. **On the formal semantics of statecharts.** In: Symposyum on logic in Computer Science, 2°, Ithaca, Proceedings, IEEE Press, pp. 54-64, New York, 1987.

IWAI; M.K. **Um formalismo gramatical adaptativo para linguagens dependentes de contexto.** Tese de Doutorado, USP, São Paulo, 2000.

KICZALES, Gregor; LAMPING, John; MENDHEKAR, Anurag; MAEDA, Chris; LOPES, Cristina Videira; LOINGTIER, Jean-Marc. **Aspect-Oriented Programming**. In: European Conference on Object-Oriented Programming (ECOOP), 06,1997. Finlândia. Anais Springer-Verlag LNCS 1241, 06, 1997.

MACORATTI, Jose Carlos. **.NET - Inversão de Controle (IoC) e Injeção de Dependência (DI).** Disponível em: <http://www.macoratti.net/11/07/ioc_di1.htm>. Acesso em 20 set. 2016.

MAES, P. **Concepts and experiments in computational reflection.** ACM Sigplan Notices, v. 22, n. 12, p. 147-155, Dec. 1987

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.

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

NETO, J.J. e MAGALHÃES, M.E.S. **Um Gerador Automático de Reconhecedores Sintáticos para o SPD.** VIII SEMISH - Seminário de Software e Hardware, pp. 213-228, Florianópolis, 1981.

NETO, J.J.; SILVA, P.S.M. **An adaptive framework for the design of software specification language**. Proceedings of the International Conferenceon Adaptive and Natural ComputINing Algorithms (ICANNGA), Springer Verlag editor, pp. 349-352, Coimbra University, Coimbra, Portugal, 21 - 23 march 2005.

NOVAES, J.M. **Um formalismo adaptativo com mecanismo de sincronização para aplicações concorrentes.** Dissertação de Mestrado, USP, São Paulo, 1997.

PEREIRA, J.C.D. **Ambiente integrado de desenvolvimento de reconhecedores sintáticos, baseado em autômatos adaptativos**. Dissertação de Mestrado, USP, São Paulo, 1999.

PEREIRA, J.C.D.; NETO, J.J. **Um Ambiente de Desenvolvimento de Reconhecedores Sintáticos Baseado em Autômatos Adaptativos**. II Simpósio Brasileiro de Linguagens de Programação - SBLP97, pp. 139-150, Campinas, 1997.

PIETRASZEK T. and BERGHE C. V. **Defending against Injection Attacks through Context-Sensitive String Evaluation.** Proceedings of the 8th International conference on Recent Advances in Intrusion Detection, pg. 124-145, Springer-Verlag Berlin, Heidelberg, 2006.

PISTORI H. et al. **Adaptive finite states automata and genetic algorithms: merging individual adaptation and population evolution**. International Conference on Adaptive and Natural Computing Algorithms (ICANNGA), Springer Verlag editor, pp. 333-336, Coimbra University, Coimbra, Portugal, 21 - 23 march 2005.

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

QUARTEL, D**. Actions Relations – Basic design concepts for behaviour modelling and refinement.** Ph.D Thesis Twente University, Netherlands, 1997.

ROCHA, R.L.A. **Um método de escolha automática de soluções usando tecnologia**

SALLEM, M. A. S. **Adapta: um arcabouço para o desenvolvimento de aplicações distribuídas adaptativas.** Trabalho de Pós-Graduação, Universidade Federal do Maranhão. 2007

SOUSA, M.A.A.; HIRAKAWA, A.R. **Robotic mapping and navigation in unknown environment using adaptive automata**. International Conference on Adaptive and Natural Computing Algorithms (ICANNGA), Springer Verlag editor, pp 345-348, Coimbra University, Coimbra, Portugal, 21 - 23 march 2005.

STAD. **State-Charts Adaptativos**. Disponível em *www.pcs.usp.br/lt*a , visitado em Julho de 2014.

Carlos Roberto Rossini Junior
Orientado

Dr. Almir Rogério Camolesi
Orientador