O Uso de Conceitos de Injeção de Dependência no Desenvolvimento de Aplicações Adaptativas

C. R. Rossini Junior, A. R. Camolesi

Resumo — ***A injeção de dependência é uma técnica que pretende inserir dependência em uma classe por meio de código externo. O presente trabalho descreve a utilização da injeção de dependência 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 dependência e da tecnologia adaptativa em sua criação, como forma de demonstrar o estudo de caso.***

Palavras Chaves — ***Tecnologia Adaptativa, Injeção de Dependência, Orientação a Objetos.***

I. 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 essa evolução é necessário entender a evolução do desenvolvimento científico e da inteligência humana. Além do desenvolvimento científico, 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 desses 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 deste trabalho foi estudar o conceito de desenvolvimento de aplicações complexas com foco no uso de Injeção de Dependência para implementação dos conceitos da 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, na Seção 2 são apresentados os principais conceitos de Tecnologia Adaptativa. A seguir, na Seção 3 são descritas as técnicas para o uso de injeção de dependência. Na Seção 4 é apresentado o estudo de caso fundamentado nos conceitos estudados e no jogo de Dominó Adaptativo. Por fim, na Seção 5 são apresentadas algumas conclusões e trabalhos futuros.

II. 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. Esse 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.

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 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 Windows (RSW) (PEREIRA, NETO, 1999).

Um formalismo paralelo ao autômato, o qual visava 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).

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.

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.

III. INJEÇÃO DE DEPENDÊNCIA

Nesta seção, serão apresentados conceitos ligados a injeção de dependência.

A. PADRÃO DE PROJETOS

Para Alexander (1978) um padrão de projeto descreve um problema o qual ocorre inúmeras vezes em um determinado contexto, e descreve ainda a solução para esse problema, de modo que essa solução possa ser reutilizada sistematicamente em distintas situações. Alexander idealizou esta ideia dentro do contexto da engenharia civil, porém anos depois estes conceitos foram trazidos dentro da engenharia de software.

Na projeção de um software, é muito comum em que o projetista encontre algum problema o qual possa ocorrer durante a implementação deste projeto, com isto, o projetista pode utilizar algum dos padrões de projetos que corresponde ao seu problema, para que consiga solucionar de uma maneira eficaz.

Muitos padrões de projetos estão ligado a qualidade do código, problemas de acoplamento e coesão são muito comuns em grandes projetos, o que dificulta novas implementações de funcionalidades após a finalização do projeto, com isto, existem diversos padrões de projetos os quais buscam minimizar estes problemas.

B. ACOPLAMENTO

O acoplamento significa o quanto uma classe depende de uma outra classe para realizar as suas funcionalidades.

Uma classe que possui um alto nível de acoplamento, irá ser responsável por instanciar suas dependências (objetos de outras classes). Em consequência, qualquer alteração nas classes das dependência, poderá ocasionar algum erro na classe que é responsável de instanciar essa dependência.

Já uma classe que possui um baixo nível de dependência, significa que esta classe não é responsável pela a criação de suas dependências.

O ideal seria buscar sempre um baixo acoplamento entre as classes, porém muitas das vezes para se buscar este baixo nível de acoplamento requer muito trabalho na projeção do projeto, entretanto o resultado de um projeto de software com um baixo nível de acoplamento é satisfatório, pelo o fato de uma manutenção facilitada e a possibilidade de alterações das funcionalidades deste software.

Devido a estes motivos, existem diversos padrões de projetos os quais buscam diminuir o acoplamento entre as classes ou módulos de um sistema, facilitando para o projetista o desenvolvimento deste projeto.

C. INVERSÃO DE CONTROLE

A inversão de controle é um padrão de projetos o qual visa diminuir o acoplamento de uma classe.

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 será responsável por instanciar suas dependência, fazendo com que esta classe adquire um baixo nível de acoplamento.

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, facilitando a maneira em que o usuário irá a utilizar e as possíveis futuras manutenções nessa framework.

D. INJEÇÃO DE DEPENDÊNCIA

A injeção de dependência é uma das maneiras de se realizar a inversão de controle. Ela permite com que seja retirado a responsabilidade de uma classe tenha que instanciar suas dependência, passando a tarefa de instanciar essas dependência para código externo.
Uma das vantagens de se utilizar o padrão de injeção de dependência é a facilidade de se aplicar ela, existem três maneiras de se aplicar das quais são as mais utilizadas: Injeção por Construtor, Injeção por Método Setter e Injeção por Interface.
Para exemplificar a maneira de se aplicar a injeção de dependência dentro de uma classe, será demonstrados alguns códigos os quais irão demonstrar o método injeção por construtor e a injeção utilizando o método setter escritos na linguagem de programação C#.

```
public class Nota
{
    private Disciplina disciplina;
    private Aluno aluno;
    private int valor;

    public Nota(Disciplina disciplina, Aluno aluno)
    {
        this.disciplina = disciplina;
        this.aluno = aluno;
    }
}
```

Figura 1. Código exemplificando a injeção por construtor.

No caso demonstrado, a classe Nota não será responsável de instanciar os objetos de suas dependências, devido ao fato desses objetos serem instanciados fora da classe e pelo fato deles serem passado através dos parâmetros do construtor.

```
public class Nota
{
    private Disciplina disciplina;
    private Aluno aluno;
    private int valor;

    public setNota(Aluno aluno)
    {
        this.aluno = aluno;
    }
    ...
}
```

Figura 2. Código exemplificando a injeção pelo método setter.

Neste caso, um objeto da dependência é entregue a classe a partir de um método dela, sendo assim, caso a classe necessite utilizar desta dependência para executar alguma ação a qual a dependência seja necessária, é necessário com que se utilize o método para receber o objeto desta dependência.

IV. 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, com isso, o foco do estudo de caso seria a implementação dos conceitos de tecnologia adaptativa e de injeção de dependência para a a construção do jogo de dominó adaptativo.

A. 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 que a regra seja 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.

B. 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 permite que uma função adaptativa modifique em tempo de execução as regras do jogo de dominó, por fim, para realizar o desenvolvimento dos conceitos da tecnologia adaptativa foi utilizadas os conceitos do padrão de injeção de dependências.
O Dominó Adaptativo teve suas classes modeladas pela a ferramenta Astash Communty e foi desenvolvido na plataforma .NET utilizando Visual Studio Community 2015.

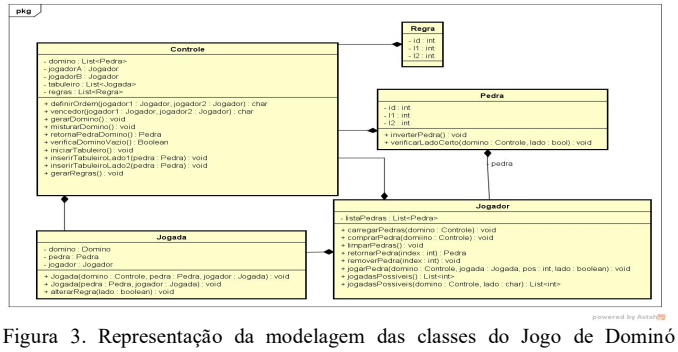

Figura 3. Representação da modelagem das classes do Jogo de Dominó Adapatativo.

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 de geração do dominó. 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 as pedras adaptativas, para que o jogo não se torne vicioso, utiliza-se de uma função aleatória com base nos valores da data e hora do embaralhamento como um índice aleatório, conforme demonstrado na Figura 4.

```
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;
   }
}
```

Figura 4. Representação do método de embaralhamento das pedras do dominó

Para o jogo não se tornar injusto, é definido um limite de pedras que possam realizar jogadas adaptativas, as quais são escolhidas aleatoriamente dentro do tabuleiro já com as pedras embaralhadas.
O dominó adaptativo também possui um método de verificar as possíveis jogadas, o qual é um método polimórfico, o qual irá fazer a verificação de todas as possíveis jogadas do jogador, conforme as pedras que o jogador possua na mão e as regras de encaixe vigentes. Este é um método 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. O método foi utilizado para verificar se uma jogada é realmente válida, de acordo com as regras vigente, as quais podem ser alteradas por uma jogada adaptativa tanto do jogador como do computador.
Para ser possível implementar a tecnologia adaptativa, foi utilizado dos conceitos do padrão de injeção de dependência, desta forma, foi possível realizar a implementação da tecnologia adaptativa sem precisar ter que alterar a estrutura do jogo de dominó.
As regras do dominó foi utilizado como o mecanismo subjacente desse sistema, conforme foi definido na projeção inicial do dominó adaptativo, as regras irão se modificar apenas no momento de uma jogada, sendo ela uma jogada adaptativa, contanto a classe jogada precisava deste mecanismo subjacente para realizar as alterações das regras, o qual estava contido na classe Controle, porém não cabe a classe Jogada instanciar essa dependência, portanto foi utilizado dos conceito da injeção de dependência para retirar a responsabilidade da classe Jogada de instanciar um objeto da classe Controle, passando esta responsabilidade para um código externo.
A técnica utilizada para a injeção de dependência na classe jogada foi por via Construtor.

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

Figura 5. Utilização da Injeção de Dependência na classe Jogada.

Com o objetivo de demonstrar o uso da tecnologia adaptativa para alterar as regras de encaixe do jogo de dominó é apresentado no Figura 6, 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.

```
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;
    }
}
...
```

Figura 6. Método Adaptativo.

O método inicia realizando uma busca nas regras, para que assim localize a regra que será modificada. Ao ser localizada é realizado uma alteração na regra, para que as regras fiquem condizentes a ultima jogada adaptativa.

C. EXECUÇÃO DA APLICAÇÃO

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

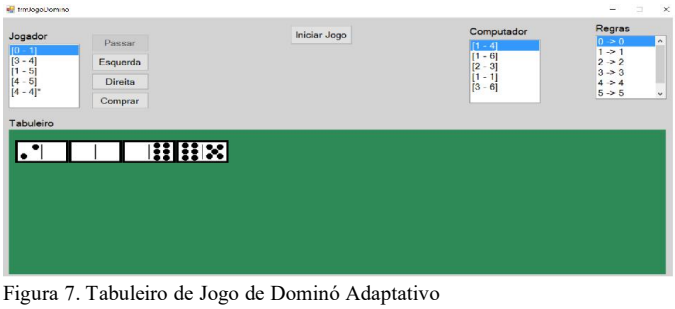

Figura 7. Tabuleiro de Jogo de Dominó Adaptativo

Para realização de testes, as pedras do computador são visíveis, as quais as jogadas são realizadas automaticamente de acordo com as regras vigentes.
As pedras que possuírem um asterisco,significa que ela é uma pedra adaptativa, com isso, o jogador tem a possibilidade de realizar uma jogada com ela em qualquer lado do tabuleiro, independentemente se é uma combinação válida ou não, caso

a combinação não for válida, o método adaptativo é utilizado e modifica as regras do jogo, as quais irão ficar em vigor até o fim do jogo.

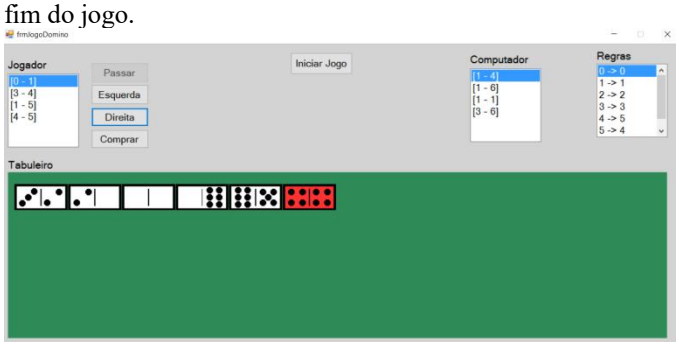

Figura 8. Realização de uma jogada adaptativa.

A figura 8 demonstrou como ficam as regras de encaixe após acontecer uma jogada adaptativa, no caso referente, as pedra que tiverem lado com valor igual a 4 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.

V. CONCLUSÕES

Considerando-se os dados apresentados 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, pois ao se possuir um sistema em que suas classes possuem um baixo acoplamento, o qual a injeção de dependência contribuiu para isto, a implementação de uma nova funcionalidade, como no caso, a tecnologia adaptativa, é realizada de uma maneira simples, ao contrário de um sistema que possuem suas classes com muito acoplamento, o que irá dificulta o processo de implementação da tecnologia adaptativa, devido a provável necessidade de realizar mudanças em várias partes do sistema.

Após a implementação da tecnologia adaptativa no jogo de dominó, foi possível perceber que mudou significativamente a maneira de se jogar, pois o jogador sempre irá possuir regras diferente, fazendo com que o jogo tenha que prestar mais atenção nas regras que mudaram constantemente.

Existem diversos tipos de padrão de projetos além do padrão de injeção de dependência, fica como trabalhos futuros, o estudo da aplicações de 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.

REFERÊNCIAS

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] ALEXANDER, C. A Pattern Language (Estados Unidos: Oxford University Press), 1977.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

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

[17] 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.

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

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

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

[21] ROCHA, R.L.A. Um método de escolha automática de soluções usando tecnologia

Carlos Roberto Rossini Junior Estudante de Bacharel em Ciência da Computação na Fundação Educacional do Município de Assis (FEMA) em Assis, São Paulo, Brasil. Entre os anos de 2015 a 2018. Suas principais áreas de pesquisas são: Orientação a Objetos e desenvolvimento de aplicação .NET.

Almir Rogério Camolesi possui graduação em Processamento de Dados pela Fundação Educacional do Município de Assis (1992), mestrado em Ciência da Computação pela Universidade Federal de São Carlos (2000) e doutorado em Engenharia de Computação e Sistemas Digitais pela Universidade de São

Paulo (2007). Atualmente é professor titular da Fundação Educacional do Município de Assis. Tem experiência na área de Ciência da Computação, com ênfase em Tecnologias Adaptativas, atuando principalmente nos seguintes temas:. tecnologia adaptativa, computação distribuída, teoria da computação, modelagem abstrata, ensino a distância, algoritmos e programação para web.