Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

RAFAEL SEBASTIÃO BERNINI

UTILIZAÇÃO DE TABELA DE DECISÃO ADAPTATIVA EM SISTEMA FINANCEIRO

FEMA

ASSIS 2015

RAFAEL SEBASTIÃO BERNINI

Qualificação do curso de Curso de Ciência da Computação, do Instituto Municipal Educacional do Município de Assis – IMESA e à Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando: Rafael Sebastião Bernini

Orientador: Dr. Almir Rogério Camolesi

Assis

2015

FICHA CATALOGRÁFICA

BERNINI, R. S.

Utilização de Tabela de Decisão Adaptativa em Sistema Financeiro / Rafael Sebastião Bernini, Fundação Educacional do Município de Assis – Assis, 2015. 63p.

Orientador: Dr. Almir Rogério Camolesi

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis

1 –Tecnologia Adaptativa; 2 - Tomada de Decisão Adaptativo; 3 – Sistema Financeiro.

CDD:

Biblioteca da FEMA

DEDICATÓRIA

Dedico esse trabalho a toda minha família, amigos e professores, que acreditaram e apoiaram em todo esses momentos maravilhosos da minha vida.

AGRADECIMENTOS

Agradecer fortemente ao meu orientador e um grande amigo de longa jornada Almir Rogério Camolesi por me orientar, acreditar e tornar todo esse sonho possível.

Agradecer ao 1º avaliador Fabio Éder Cardoso por incentivar avaliar e poder dispor de seu carinho em todo o trabalho produzido.

Aos meus familiares Antônio Bernini, Eloiza Alves Bernini e Jonas Alves Bernini que esteve ao meu lado em todos os meus momentos maravilhosos da minha existência.

Agradecer a minha esposa Caroline Letícia Nascimento por me ceder tantos momentos de felicidades e alegria, e também por me dar o maior e mais iluminado presente da minha vida, meu filho Davi Nascimento Bernini, pois foram eles quem me deram forças, razão por lutar, querer algo cada dia mais e também me mostrar o verdadeiro sentido da vida.

E a todos meu amigos que me apoiaram e foram tão presente nesse percurso e na concretização desse trabalho.

Sumário

Imagens

Tabelas

Fontes

1. Introdução

A tecnologia adaptativa é um conceito constituído de mecanismo científico capaz de buscar novos modelos para auxiliar ou possibilitar soluções de diferentes níveis de complexidade.

Seus métodos muito competitivos como redes neurais, inteligência artificial, algoritmos genéticos, exigem níveis de complexidade em suas classes, e de problemas a serem resolvidos, a tecnologia adaptativa intermediá esse meio de campo para que sua facilidade nesses níveis de complexidade se tornam de fácil uso. Sua operação pode ser descrita como modo incrementado, e de seu modo comportamental, que torna a resolução do problema mais dinâmico a partir dos estímulos causados pelos dados recebidos, obedecendo um conjunto de regras que definem esse tipo de comportamento, que executará em tempo de execução, complementando a si mesmo conforme sua necessidade e através de uma sequência e seus estímulos em sua inserção de dados.

A tecnologia adaptativa pode ser inserida em amplas áreas de atuação e situação, sendo a solução para seguimentos muito complexo de um determinado problema em que exigiria muitas decisões que um algoritmo convencional não resolveria ou necessitária muito da lógica convencional, sendo substituída por uma tomada de decisão tornando mais comportamental em sua tomada de decisão.

As tomadas de decisões são aplicadas dentro da tecnologia adaptativa, ramificando e descrevendo uma sequências complexas em forma de árvore com regras de formações dinâmicas com grandes características flexíveis, contendo dentro de si conjuntos complexos de regras que serão de modo trivial aplicada em uma futura decisão ao seja necessária uma melhor forma comparativa a essa tabela de decisão para o algoritmo.

1.1 Objetivo

O Objetivo deste trabalho é utilizar Tabela de Decisão em um sistema financeiro, estudando os aspectos do uso de uma aplicação comercial para realizar melhorias fiscais, tornando a análise do processo mais inteligente e eficiente, com isso suas rotinas que antes necessitaria de um complexo algoritmo para sua decisão financeira que será substituído por uma tabela de decisão que fará uma análise em tempo de execução.

1.2. Justificativas

A tecnologia adaptativa surge com uma solução inteligente e ágeis, com isso ter um sistema mais inteligente e auxiliando de modo dinâmico em tempo de solução, torna o trabalho menos tortuoso de modo que tenha a capacidade de aprender certo aspecto de rotinas diárias. A partir do momento em que o dado é inserido no programa, sua execução passa de maneira inteligente pelo algoritmo que analisa a requisição e em tempo de execução é capaz de tomar certa decisão otimizando parte da rotina.

1.3 Motivações

O estudo da Tecnologia Adaptativa e Tabela de Decisão Adaptativa, em um futuro próximo venha a ajudar empresas de inovar novas tecnologias, sendo elas mais inteligente, aplicando-as diretamente em seus módulos e as deixando uma solução que tenha capacidade de se portar de maneira mais comportamental na aplicação financeira sendo capaz de tomar suas próprias decisões e fazendo disso um aprendizado para o próprio sistema.

1.4 Estrutura do Trabalho

Este trabalho está organizado em capítulos. Sendo o primeiro capítulo uma introdução sobre o projeto, apresentando os objetivos e as justificativas para as principais tecnologias que contem o trabalho.

O segundo capítulo uma introdução sobre os autômatos e seus principais conceitos e funcionalidades para seguimentos presente no trabalho. O terceiro capítulo apresenta conceitos fundamentais para sobre tecnologia adaptativa destrinchando minunciosamente seus conceitos e estrutura para serem aplicados.

No capítulo quatro, aborda-se o conceito de tabela de decisão adaptativa e suas tomadas de decisão relacionado a sua estrutura. No quinto capitulo apresenta-se computação reflexiva, que aplica seus metadados para desenvolvimento de aplicações dinâmicas e reflexiva.

O capítulo seis demonstra quais as funcionalidade e soluções que irá agregar no desenvolvimento da aplicação. O capítulo sete demonstra a plicação em sua estrutura da aplicação e como foi elaborada.

O sétimo capítulo apresenta a conclusão do trabalho e os planejamento para trabalhos futuros.

E por fim, o último capítulo trás as Referencias Bibliográficas utilizadas para o desenvolvimento do trabalho.

2. Autômatos

MENEZES 1990 diz que as Linguagens Formais destacam-se relacionados às aplicações em análises léxicas e análise sintática das linguagens de programação, modelagem de circuitos lógicos ou redes lógicas, entre outros. Podem-se destacar também aplicações relacionadas a sistemas de animação, hipertextos e hipermídias, também o tratamento de linguagens não-lineares, como linguagens espaciais e linguagens dimensionais, pois são aplicações que se destacaram recentemente.

Para estudar Linguagens Formais, além de muitos outros conceitos, são também necessários conhecimentos em autômato, que em um contexto resumido, é um modelo matemático utilizado para simular uma máquina composta por estados, que são finitos, ou seja, possuem um fim.

TIARAJÚ 1990 nos ensina que um autômato, pode-se entendê-lo como um reconhecedor de uma determinada linguagem para poder modelar uma máquina, ou mais comumente conhecido, um computador simples. Este simples conceito aplica-se a qualquer sistema, como a televisão, por exemplo, seu sistema seria o fato de possuirmos dois estados, o ligado e desligado. Conceitos de sistema e estado estão onipresentes nas máquinas, como perceber-se no exemplo da televisão, chega-se à conclusão que estão em qualquer aparelho desta forma estão onipresentes em nossas vidas também, com isto presumimos a importância do estudo de autômato, pois tendo profundo conhecimento do assunto, poderão ser construídas aplicações para as mais diversas questões, e consequentemente suas soluções, existentes atualmente.

2.1 Autômato Finito

Autômato finito é um sistema de estados finitos (portanto possuem um número finito e predefinido de estados) o qual constitui um modelo computacional do tipo sequencial muito comum em diversos estudos teóricos formais da computação e informática.

Um autômato finito é uma quíntupla definida por M =

Sendo:

Q = conjunto finito de estados Σ = alfabeto

δ = função de transição Q x Σ → Q q0 = estado inicial

F Q = conjunto de estados finais (aceitadores)

Pode ser caracterizado das seguintes formas:

•Determinístico

•Não-determinístico

•Com movimentos vazios

2.2 Autômato Determinístico

Este tipo de autômato caracteriza-se pelo fato de que a depender do estado corrente e do símbolo lido o sistema poderá assumir um único estado de forma que todos os estados estarão bem determinados.

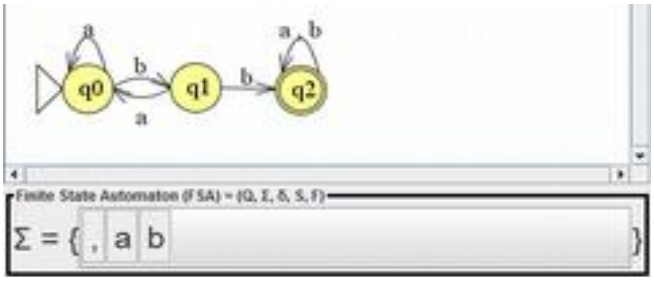

Figura 1 – Exemplo de autômato determinístico (Autoria própria 2015)

Pode-se perceber que em todas as transições existe um próximo estado bem determinado, caracterizando dessa forma o conceito de autômato determinístico.

2.3 Autômato Não-Determinístico

Outro formalismo é caracterizado pelo fato de que a depender do estado corrente e do símbolo lido, o sistema pode assumir um conjunto de valores alternativos, ou seja, para cada estado não necessariamente estará determinado qual será o próximo estado.

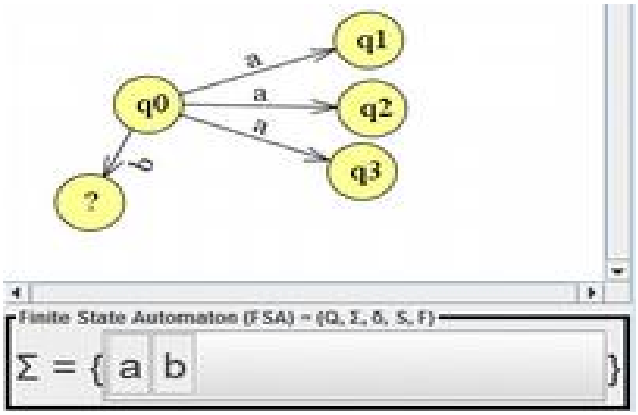

Figura 2 – Exemplo de autômato não-determinístico. Como está bem claro na figura 2, percebe-se que no estado q_n não deixa claro qual será o seu próximo estado caso seja lido a, caracterizando dessa forma o autômato não-determinístico. (Autoria própria 2015)

2.4 Autômato Com Movimentos Vazios

A caracterização deste formalismo dar-se pelo fato de que a depender do estado corrente e sem ler qualquer símbolo o sistema pode assumir um conjunto de estados, logo, também é não-determinístico. Seu diferencial está no fato de que movimentos vazios podem ser vistos como transições encapsuladas nas quais, executando-se por uma eventual mudança de estados, nada mais pode ser observado.

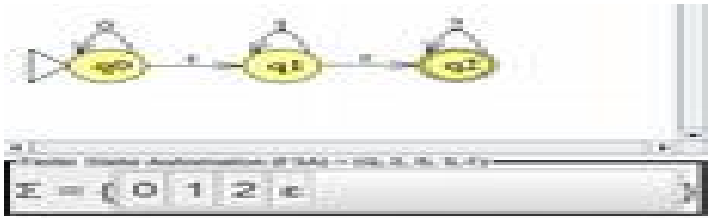

Figura 3 – Exemplo de autômato com movimentos vazios (Autoria própria 2015)

2.5 Autômatos Finitos: Configuração Instantânea

A a expressão [qi, w] de um AF corresponde ao estado qi e cadeia w ainda não processada no dado instante (um elemento de Q x Σ*). A configuração instantânea define o comportamento futuro do AF.

FUNÇÃO DE TRANSIÇÃO ESTENDIDA EM AUTÔMATOS FINITOS

A função de transição estendida de Q x Σ* em Q descreve formalmente a operação de um AF sobre uma cadeia, e é definida por:

Função de transição estendida

$$\hat{\delta}(q,\varepsilon) = q$$

$$\hat{\delta}(q,wv) = \delta(\hat{\delta}(q,w)\ v)$$

Figura 4 – Função de transição estendida (MENEZES 1990)

Uma cadeia x é aceita por M se $\hat{\delta}(q0, x) = p \quad F$.

A linguagem L(M) aceita por M é o conjunto de cadeias em Σ* aceitas por M

Dois AFs que aceitam a mesma linguagem são ditos equivalentes.

2.6 Autômatos Finitos E Máquinas De Estados

Um autômato finito é uma máquina de estados composta por um registrador de estado interno (controle finito) + uma fita segmentada + cabeça de leitura

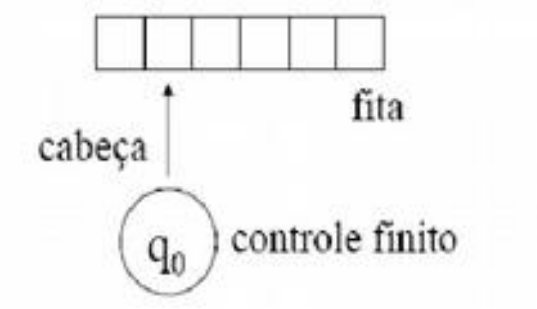

Figura 5 – Exemplo de máquina de estados (MENEZES 1990)

fita: armazena uma cadeia de Σ (1 símbolo/segmento). cabeça: lê segmento da fita (um símbolo da cadeia).

registrador: altera estado de acordo com δ e move a fita um segmento para a esquerda (computação).

uma computação termina quando a cadeia é processada (acaba). cadeia é aceita pelo AF se a computação termina em um estado q Є F e é o último símbolo da fita. cadeia é rejeitada pelo AF se a computação termina em um estado q ∉ F

2.7 Equivalência Entre Autômatos Finitos E Autômatos Finitos Não Determinísticos

Para poder entender a equivalência entre autômatos finitos e autômatos finitos não determinísticos é preciso analisar o seguinte teorema:

Seja L uma linguagem aceita por um AFND. Então existe um AF que aceita L.

Prova: Seja M = (Q, Σ, δ, q0, F) um AFND. Onde a ideia consiste em:

-Construir um AF M' = (Q', Σ, δ', q0', F') que simula computações realizadas por M.

-O AF manterá como "estado" o conjunto de todos os estados que podem resultar após uma dada computação de M.

Seguindo este teorema, poderá ser analisado se os autômatos são equivalentes ou não.

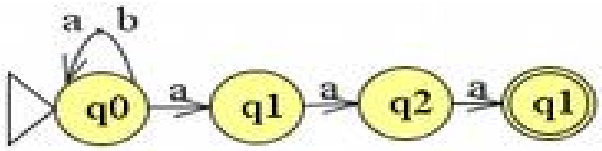

Figura 6 – Autômato finito não-determinístico (Autoria própria 2015)

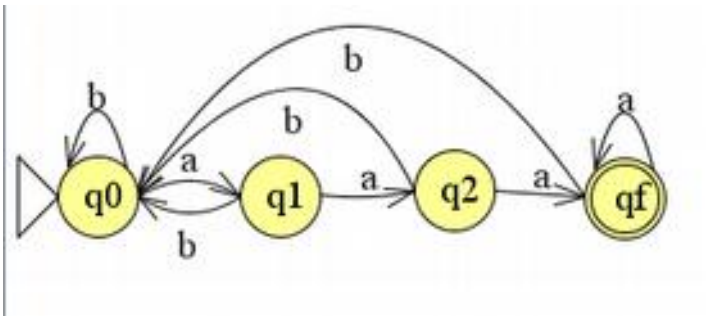

Figura 7 – Autômato finito determinístico (Autoria própria 2015)

As figuras 6 e 7, a exemplo do teorema, mostram respectivamente os autômatos M e M´, não-determinístico e determinístico respectivamente. É provado que eles são equivalentes pela figura 8, pois o autômato mantém como “estado” o conjunto de todos os estados que podem resultar após uma dada computação de M.

δ 6D	Ideia	bb
(q0)	(q0q1)	(q0)
(q0q1)	(q0q1q2)	(q0)
(q0q1q2)	(q0q1q2qf)	(q0)
(q0q1q2qf)	(q0q1q2qf)	(q0)

Tabela 1 – Diagrama de conversão (Autoria própria 2015)

Prova-se que são equivalentes e prova-se da seguinte forma: monta-se o diagrama de estados de cada de estado para cada letra do alfabeto, mostrando sua origem e seu destino. Desta forma seus estados serão determinados, ou seja, o autômato que antes era não determinístico, agora será determinístico.

3. Tecnologia Adaptativa

Com a evolução necessita entender que a inteligência humana tem parâmetros relacionados a problemas cada vez mais complexo impossibilitando que apenas uma única área seja responsável por tais atos existentes.

Yourdon (1990) relatou que o sistema assemelha a um conjunto estruturado ou de certa forma ordenado em devidas partes de um elemento que se completa e que se mantêm integrado a uma ação recíproca em busca de com seguimento de uma demanda muito grande de objetivos, entretanto, tornando de modo mais simples definir um módulo sistemático com um conjunto de elementos interdependentes, ou partes que formam interagindo entre si um unitário complexo e completo, definindo previamente que existem muitas partes menores completando e complexando uma parte maior.

Sendo assim os sistemas devem de modo geral ser construídos para atender a empresas de um modo ao seu todo, com um desenvolvimento de software cada dia mais complexos e melhor adaptado tecnologicamente a grande demanda de exigências fundadas em cada necessidade específica.

Camolesi (2004) cita que as aplicações complexas são caracterizadas necessariamente de componentes e seus aspectos cuja a estrutura comportamental relata-se a adaptar e modificar a sua necessidade especial. Essas aplicações possuem um temperamento inicial definido por um conjunto de ações modificando para suportar uma infinidade de novas possibilidades que possa atender, sendo assim tais modificações decorre a estímulos de entrada de dados ou informações repassadas pelo sistema decorrente de uma função interna acionária. Utilizando se de técnicas projetadas na modelagem comportamental da aplicação, sendo modificável.

Pistori, (2003) cita que a tecnologia adaptativa envolve um dispositivo não-adaptativo (subjacente) já existente em uma camada adaptativa que permite realizar mudanças no comportamento da aplicação definida. Estudos ao qual já foram realizados com o objetivo de se criar uma ferramenta 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, ou seja, a aplicação não sofre modificações em sua total estrutura, e sim utiliza partes de um sistema ao qual trata de auxiliar de modo dinâmica e adaptativa certo processo complexo da aplicação. O programador deve apenas acrescentar novos aspectos ao software e o mesmo se adapta ao código já existente. Porém neste trabalho verificou-se que os conceitos de Tecnologia Adaptativa não são implementados completamente, uma vez que os programas que utilizam Programação Orientada a Aspectos são combinados antes da compilação e os programas produzidos não acabam se tornando adaptativos em tempo de execução.

NETO e MAGALHÃES (1981), foram os pioneiros no estudo de tecnologia adaptativa fornecendo uma visão de como seria tais métodos de análise sintática e de geração de reconhecedores sintáticas.

A elaboração de NETO (1983) mostra como modelo, capacitando uma tradução sintática e desenvolvendo uma versão pratica de algoritmos de conversão de gramática em reconhecimentos sobre compilação. Neto (1988) apresenta um objetivo de facilitar a apresentação de seus respectivos problemas, ao qual, a finalidade de uma classe de máquina de estados finitos com uma memória organizada em pilha que em seu final do recurso exibe seu aprendizado com uma alteração com base nas transições efetuadas pelo tradutor.

NETO (1993), obteve resultados significativos com seu tradutor, sofrendo um aumento no poder representativo em modelos matemáticos, com isso emprega meios de inclusão de recursos em forma de aprendizado em seu tradutor adaptativo, firmando que são capazes de aprenderem sozinho em forma de uma sentença apresentada de um jeito extremamente natural, tornando o tratamento de vários problemas sintáticos alterando adaptativamente a semântica da sintaxe da linguagem proposta. Tais resultados do estudo, mostra que a partir de técnicas clássicas e simples são capazes de interagir e integrar métodos elementares da análise necessárias para a resolução de grandes partes

interessadas, que seria um autômato e um tradutor adaptativo com dispositivos de reconhecimento e tradução sintática.

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), é um simulador de sistemas que utiliza a marcação desenvolvida. Que propiciado uma visão prática da teoria do uso de tecnologia adaptativa, a ferramenta STAD prova a viabilidade de maneira simples e útil a união de tais conceitos representado.

NETO, ALMEIDA e NOVAES (1998) desenvolveu uma aplicação para desenvolvimento e analise a STAD-S, uma ferramenta com um editor Statechart e de um simulador de Statecharts assíncronos, permitindo que um sistema seja em diversos pontos de vista provenientes das características formais apresentadas ao formalismo desenvolvido, com um aspecto de hierarquia e concorrência fundamentada nas características de Statechart, com aspecto de sincronização com base em aprendizado que utiliza de conceitos de automodificação proveniente de Statecharts Adaptativos.

PEREIRA (1997), desenvolveu um “Ambiente de Desenvolvimento de Reconhecedores Sintáticos Baseado em Autômatos Adaptativos”, que foi introduzida ao auxílio de desenvolvimento de analisadores sintáticos para Windows a RWS, tipo de ferramenta são baseadas na teoria de autômato de estados finitos, sua implementação reconhece dados baseados em Autômatos Adaptativos que constitui importantes metas alcançadas por tal ferramenta.

ROCHA e NETO (2000) criou um método de estudo para o desenvolvimento de um modelo constituído no modelo e resolução de problemas mais complexos utilizando tecnologia adaptativa, com fim comparativo entre respectivos autômatos adaptativos, redes neurais, algoritmos genéticos entre outros, coletando dos mesmos dados aos quais permitiram a composição de um modelo de Busca de Soluções por Maquinas Adaptáveis (BSMA). Autômatos Adaptativos para a escolha de solução de problemas e apresentada uma proposta para o uso da tecnologia adaptativa na simulação de Redes Neurais em

ambientes computacionais (ROCHA, 2001; ROCHA, NETO, 2001). Tais estudos serviram para demonstrar a possibilidade de utilização da tecnologia adaptativa em aprendizagem computacional e, de maneira geral, em inteligência artificial.

Freitas (2000) concluiu um estudo com base à aplicação da tecnologia adaptativa no desenvolvimento de ambientes que suportem a tarefas do tipo multilinguagens de programação, uma proposta de implementação de um ambiente que agiliza a aplicação de uma certa tipagem de multilinguagem por meio das definições antiquadas que facilitem a interação entre os diversos segmentos de linguagens que compõem o mesmo. Tais dados estabelecem um mecanismo de gerenciamentos relativos às diferentes variáveis que são importadas ou exportadas entre os módulos componentes da aplicação multilinguagem. Desta forma, ocorre a necessidade de um mecanismo que gerencie o espaço de nomes do ambiente. Neste contexto, Freitas e Neto (2000a) empregaram o autômato adaptativo como analisador de nomes. Embora simples, esta estrutura representa uma alternativa eficiente e elegante para representar as estruturas de dados de armazenamento e busca de cadeias. Para validar a proposta de implementação do ambiente multilinguagem, foi desenvolvido um Ambiente Multilinguagem (AML), no qual as linguagens C++, Prolog, Lisp e Java podem ser utilizadas ao mesmo tempo de outras (FREITAS; NETO, 2000b, 2001).

NETO e SILVA (2005), mostrara uma aplicação adaptativa com uma especificação de uma linguagem, estrategicamente feita para entender certas características de determinada estrutura adaptativa. 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.

PISTORI e NETO (2002) apresentaram o AdapTree - um algoritmo para indução de árvores de decisão que usa técnicas adaptativas - e os primeiros resultados da aplicação de técnicas adaptativas na produção de algoritmos de aprendizagem eficientes. Um protótipo de um sistema cuja interface com o usuário era feita pela direção do olhar utiliza técnicas de baixo custo, fundamentadas em aprendizado computacional e que usa tecnologia adaptativa pode ser encontrado em (PISTORI et al., 2003).

PISTORI (2003) mostra um Autômato de Estados Finitos Adaptativos e mostra como é fundamental uma complementação para a representatividade de certa função e tais ações adaptativas, com uma integração de dispositivos adaptativos discretos, com esse trabalho foram desenvolvidas exemplos com a utilização de tecnologia adaptativa em seu desenvolvimento de aplicação.

LUZ (2004) introduziu uma ação adaptativa em forma de algoritmo de otimização que tem a capacidade de uma forma autônoma de modificar seu comportamento e respostas a partir de uma entrada especifica, procura uma sequência de regras de otimização que se aplica diretamente no código fonte entre várias possibilidades possíveis resultando em uma regra de otimização dinamicamente aplicável.

A utilização de autômatos adaptativos para mapeamento de movimentos de robôs móveis autônomos pode ser apreciada em (SOUSA; HIRAKAWA; NETO, 2004a e 2004b). Inicialmente, o robô possuía um pequeno mapa do ambiente que era ampliado por meio dos caminhos percorridos por ele mesmo. Para tal, foi construído um algoritmo que utilizava técnicas adaptativas que inicialmente adicionavam algumas marcas livres que eram modificadas com informações obtidas pelos sensores. Sousa e Hirakawa (2005) demonstram o funcionamento da navegação do robô utilizando esta estrutura.

3.1 Tecnologia Adaptativa autoadaptação

O trabalho proposto neste projeto tem como base principal a Tecnologia Adaptativa, sendo esta, uma tecnologia com a característica de autoadaptação. Neste sentido espera que o desenvolvimento desta pesquisa possa contribuir com publicações para esta área direcionado para o ramo comercial corporativo a fim de sanar necessidades complexas. O principal diferencial na Tecnologia Adaptativa é a forma razoavelmente simples que pode transformar teorias já existentes, bem como fazer um reaproveitamento e estruturação destas para melhorar sua capacidade de respostas e interação. A 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, 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.

GONÇALVES E CAMOLESI (2012) apresentam o desenvolvimento de uma ferramenta para geração automática de aplicações comerciais a partir do modelo de Banco de Dados projetado para a aplicação. O desenvolvedor utilizando-se de um assistente gera automaticamente, de forma rápida e simples, uma aplicação para realizar manutenção dos dados desejados. A aplicação gerada permite ao usuário da mesma realizar operações de manutenção (inclusão, alteração, recuperação e remoção) dos dados. Além das operações básicas de manutenção a ferramenta permite que relatórios básicos também sejam gerados de forma automática.

3.2 – Mudança de comportamento

Com a evolução crescente da Internet, e das redes de computadores, com facilidade de aquisição de dispositivos móveis com o surgimento da computação em nuvem depara-se com a necessidade de aplicações que estejam todo o tempo conectado em toda a parte e possam ser executadas em diversos ambientes e locais. Os programas geralmente são escritos para trabalhar com dados. Eles em geral, leem, escrevem, manipulam e exibem dados (a geração de Gráficos a partir de bases de dados preexistentes são um exemplo típico de programas desse tipo). Os tipos que você, como o programador, criar e usar é projetado para estes fins, e é você, em tempo de projeto, que deve compreender as características dos tipos que você usa. Para alguns tipos de programas, no entanto, os dados que eles manipulam não são números, textos ou gráficos, mas são as informações sobre programas e tipos de programa. Tais informações são conhecidas como metadados (Uma informação sobre a informação). Os Atributos são um

mecanismo para a adição de tais informações, tais como instruções do compilador e outros dados sobre seus dados, métodos e classes, para o próprio programa. Um programa pode olhar para os metadados de outros conjuntos ou de si mesmo, enquanto ele está executando.

Nos atuais sistemas de informação existentes no mercado apresentam um comportamento pragmaticamente estático com estruturas pré-definidas, nos quais são informados apenas os seus parâmetros inicias e estes são utilizados durante o tempo de execução, mantendo um padrão evolutivo e estativo atualmente. No mundo empresarial o dinamismo é fundamental, novas funcionalidades e estratégias de cálculos surgem no mundo dos negócios o que acarretam em mudanças que devem ser realizadas nos sistemas existentes, e priorizar custo, funcionalidade, tempo e melhoras significativamente o crescimento aplicando sempre novas tecnologias de crescimento fundada e aspectos adaptativos. A análise relacionada ao comportamento de tais sistemas ser estático torna impossível, na maioria dos casos, mudanças automáticas ou sem a necessidade de re codificação de trechos e partes do sistema já desenvolvido. Tais mudanças além de demandar tempo e a alocação de profissionais para a sua realização, pode gerar problemas imprevistos nas aplicações e, até mesmo, terem altos custos para serem realizadas. O uso da Tecnologia Adaptativa neste tipo de aplicação tem por objetivo a atender uma necessidade da aplicação utilizando uma tabela de decisão que possa ter seu comportamento alterado com a obtenção de novas informações comparando-as e apresentando respostas em seu tempo de execução.

3.3 Autômatos Adaptativos

O autômato adaptativo é um seguimento de regras adaptativo, contendo uma camada subjacente constituído de um autômato de pilha ao qual essas camadas são implementadas através de funções adaptativas que definem como tais modificações, como pesquisa, inserção e remoção.

Quando se executa uma função adaptativa seu mecanismo de execução, faz com que os parâmetros de execução seguido por um valor automático para cada valor de parâmetro formal, a esse tal valor é redirecionado a um argumento.

Esse processo é fundamental para a execução de ação adaptativa inicial, porém opcional pode repassar caso necessário os valores recebidos para uma nova variável, para uma nova ou outra função adaptativa. Essas ações adaptativas iniciais não acessam diretamente as variáveis e geradores, por que ainda não foram definidas para tal ponto de execução da função.

Tais regras contém nomes e variáveis e de geradores no lugar de algumas regras, quando executa uma ação o elemento de consulta é responsável por atribuir valores a estas variáveis, sendo assim seu conjunto de regras subjacentes. Quando se atribui valores de cada variável não pode ser mais alterado durante a execução da função adaptativa, pois, seus padrões não conseguem encontrar qualquer regra que possa ser satisfeita com o formato determinado pela ação elementar de consulta e as variáveis permanecem indefinidas por tal ação.

Basicamente o modo de que a regra da camada subjacente contém nomes de variáveis e de geradores no lugar comportamental das regras, com isso suas ações elementares de análise é responsável por atribuir valores a resta variável, essa base contém uma regra de instruções por mecanismo subjacentes sempre atribuídos aso seus respectivos valores e de cada variável não podendo ser alterado durante sua execução da função adaptativa por reconhecer seus padrões não encontrados qualquer regra que contenha essa camada que satisfaça o formato determinando um elemento da requisição e das suas variáveis permanecendo indefinidas

Sendo assim os princípios da execução das ações são com base nos padrões de utilização determinada por valores inseridos pela variável que podem estar presentes nas ações determinadas a partir de seu valor, pois ocorre da execução pre-instanciada ter sido por acaso corresponder a uma camada adjacente ao invés da variável, portanto exige um novo comportamento da regra atribuindo novos símbolos etilizando uma nova camada.

4. Tabela de Decisão Adaptativa

A tomada de decisão tem sido objetos de pesquisa e estudo em diversas áreas da tecnologia, seu funcionamento trouxe dinamismo para a maioria das ações complexas deviso ao seu ato de decisivo implementados diretamente no software, aplicando metodologias técnicas de Tecnologia Adaptativa solucionando vários tipos de problemas de decisão.

4.1 – Tomada de decisão

Uma decisão que necessita um processo lógico racional, e seguindo essa ideia elabora-se estudos para métodos e modelos que se aplica de forma dinâmica com as tomadas de decisões necessárias, levando em conta a obediência de determinados critérios analisando cada nível de complexidade para que cada decisão seja tomada de maneira correta.

Sua adaptabilidade tem uma estrutura que se alto modifica relativo ao ambiente que requisita sua operação, não convencional as demais estruturas de decisão tradicional fora da tecnologia adaptativa, através de um método não determinístico, existe uma probabilidade imensa de se adaptar e modificar em diferentes números e modelos e de adequar a melhor opção, usando uma métrica comum com conceitos complexos. A árvore de decisão, se adapta de uma forma incrível tornando seu modo adaptativo capaz de criar seus próprios atributos e métodos que estendem a sua capacidade para a solução de coisas simples, tornando-a diferente.

Com a grande evolução computacional a capacidade de armazenamento afeta diretamente a produtividade tornando necessária a confecção de utilização de ferramentas de apoio, e quanto mais inteligente for melhor resolve problemas de processos, com regras e tomada de decisões que tornaria mais prático e funcional, além de inteligente a organização desses dados, que no caso da árvore de decisão faria com que esses dados se comportassem de modo mais

organizado, deliberadamente diferenciando partes realmente uteis e não descartáveis necessário para casos futuros baseados em um mecanismo capaz de representar modificações funcionais, organizando uma estrutura por níveis de prioridade, entre outros.

Sabendo que a tecnologia adaptativa é uma alternativa mais prática a alguma solução havendo uma parceria em suas tomadas de decisões, ou seja, ela auxilia a devidos problemas através de uma Tabela de Decisão Adaptativa que resulta na combinação de algoritmos e métodos em diversos critérios de um modo estratégico que envolve multicritérios nem sempre preciso ou completos.

TCHEMRA E CAMARGO (2008) falam que as complexidades das decisões exigem do desenvolvedor uma grande demanda de estudos gigantesca na área teórica da decisão quanto nas áreas técnicas de desenvolvimento de software. Tais métodos de feitos para auxiliar as decisões para determinados problemas envolve uma gama de necessidades especiais, e exige que tenha uma gama de decisões computacionais aliados a evolução computacional e seus respectivos casos de estudo.

O apoio a esses métodos computacionais de serviços de apoio soma uma grande quantidade de dados utilizados para pesquisa, adquiridas na maioria das vezes de previas de informações de banco de dados, com isso é possível criar um cenário que permite uma série de análises sobre um determinado problema, que será simulado um ambiente e a partir desse ponto aplicar determinadas tomadas de decisões, sempre analisando entrada de processos e assemelhando a realidade atual.

Com a inteligência artificial, desenvolve-se ferramentas que auxiliam execuções de tarefas, sendo parte delas e complementos de estudos aplicações e conceitos para tomada de decisões que assimilam a capacidade humana, contendo capacidade de aprender, raciocinar, solucionar problemas e ter a capacidade de tomar qualquer tipo de decisão ao qual melhor se enquadrar para tal problema proposto.

A tabela de Decisão adaptativa tem características que facilitam a sua utilização e compreensão em relação aos dados inseridos, sua constituição é bem

simples, ela necessita de ações que sejam facilmente identificados pelo usuário sendo critérios ou alternativas, para que possa ser analisado e decidido.

O propósito da tabela é simular um autômato adaptativo para reconhecer definições da linguagem utilizada no contexto computacional e comportamental da aplicação, NETO (2003) complementa que a função adaptativa, e suas ações adaptáveis, são capazes após ser executada, de modificar um conjunto complexo de regras de uma tabela através de inclusões e exclusões de regras, dependente de regras recebidas través de uma autônoma e meramente dinâmica.

4.2 – Tomada de Decisão

É um processo decisivo que leva em relação os fatores apresentados anteriormente, nesse ponto o é tomada a decisão por uma alternativa, dois alternativos ou muito mais para que no final seja atingido seu objetivo esperado. Dependendo da complexidade implantada, da natureza do problema e de seus objetivos a serem traçados como meta, a decisão pode ser tomada individualmente sem requerer a participação de um grupo ou seu processo tem que ser levado em conta demais elementos pertencentes a um objetivo que depende de valores ou julgamentos que implica em analisar determinados valores, modelos e partes sistemas distintos para tal decisão.

Existem casos muito complexo para tomada de decisões sendo meramente encontradas em planejamento estratégico de uma organização, devido ao método estratégico planejado que determina medidas a tais processos de decisões que vão conter valores e formarão ações escolhidas para que seu sistema seja extremamente eficaz para objetivos empresariais sejam fortemente alcançados.

TCHEMRA (2009) afirma que tais modelo englobam dados que representam as variáveis de controle que determinarão quais são as ações que podem ser substituídas ou não, bem como as variáveis incontroláveis que são relevantes ao problema em questão, e os critérios de decisão que podem levar a melhor ação dentre as necessidades apresentadas na inserção dos dados.

Á vários modelos que necessitam de decisões programadas que, geralmente são definidas por regras, procedimentos ou métodos quantitativos muito bem especificados e estruturados. Sendo assim, existem também decisões não-programadas, mais difíceis de quantificar e exigidas em situações extremas. São problemas desestruturados, onde a aplicação de regras ou de procedimentos nem sempre são bem especificados. E são encontrados problemas de decisão semiestruturados que visualizam características estruturadas e outras não estruturadas.

5. Computação Reflexiva

O sistema de raciocinar sobre si mesmo, havendo a capacidade de solucionar problemas em tempo de execução, os quais somente poderiam ser resolvidos com módulos complementar de novos códigos na programação.

Forman & Forman 2008 diz que uma classe pode acessar outras classes no seu tempo de execução do processo da ação, sem conhecer sua definição no momento da compilação. Relativo à esta definição, como seus construtores, métodos e atributos, podem ser facilmente acessados através de métodos de reflexão. Tais classes externas à aplicação devida aplicação, que não foram compiladas junto a mesma, podem ser instanciadas para utilização de seus recursos. Os recursos de reflexão oferecidos em sua maioria dos casos, são utilizados para prover extensão de funcionalidades a aplicações, desenvolvimento de ferramentas de debug e aplicativos que permitem a navegação no conteúdo de classes compiladas.

Segundo Guerra (1998) o desenvolvimento da reflexão computacional trouxe pela primeira vez um conceito de linguagens procedurais a partir do ano de 1982, consolidando o conceito de reflexão à definindo como um processo que pode modificar sua própria estrutura e comportamento, através deste meio a reflexão ainda é utilizada como meta programação e esse conceito é capaz de realizar computações a respeito dele mesmo

Reflexão é comumente usado por programas que exigem a capacidade de examinar ou modificar o comportamento de tempo de execução de aplicativos em execução na máquina virtual Java. Esta é uma característica relativamente avançada e deve ser usado somente por desenvolvedores que têm uma forte compreensão dos fundamentos da linguagem. Com essa advertência em mente, a reflexão é uma técnica poderosa e pode permitir que os aplicativos para executar operações que de outra forma seria impossível.

Barth (2000), a reflexão pode ser definida como qualquer ação executada por um sistema computacional sobre si próprio, pode-se dizer que reflexão é a capacidade de um programa reconhecer detalhes internos em tempo de execução que não estavam disponíveis no momento da compilação do programa.

Reflexão Computacional tem dois significados distintos. Um é introspecção, que se refere ao ato de examinar a si próprio, o segundo é intercessão, onde está ligado ao redirecionamento da luz, ou seja, o termo reflexão computacional na área da informática tem a capacidade de examinar ou conhecer a si mesmo ou estrutura, podendo fazer alterações no seu comportamento através do redirecionamento ou de interceptação de operações efetuadas.

Segundo Barth (2000), a reflexão Computacional define em uma nova arquitetura de software. Este modelo de arquitetura é composto por um meta nível, onde se encontram estruturas de dados e ações a serem realizadas sobre o sistema objeto, localizadas no nível base. Neste contexto, a arquitetura de meta nível é explorada para expressar propriedades não funcionais do sistema, de forma independente do domínio da aplicação.

Requisitos funcionais não apresentam nenhuma função a ser realizada pelo software, e sim comportamentos que este software deve utilizar.

5.1 – Recursos da Reflexão

Um aplicativo pode fazer uso de classes externas, definidas pelo usuário, criando instâncias de objetos de extensibilidade usando seus nomes totalmente qualificados.

Um navegador da classe precisa ser capaz de enumerar os membros de classes. Ambientes de desenvolvimento visuais podem beneficiar de fazer uso de informações de tipo disponíveis na reflexão para ajudar o desenvolvedor de escrever código correto.

Depuradores precisa ser capaz de examinar os membros privados em classes. Equipamentos de teste pode fazer uso de reflexão para chamar sistematicamente um detectáveis APIs conjunto definido em uma classe, para garantir um elevado nível de cobertura de código em uma suíte de teste.

A dois modelos de Reflexão Computacional, o primeiro é a reflexão estrutural, pode ser definida por qualquer atividade exercida em uma meta classe, a reflexão estrutural permite em tempo de execução fazer alterações em seus componentes no processo de execução ou compilação (Sallem 2007).

Reflexão estrutural tem as funcionalidades de realizar algumas transformações básicas, tais como: informações sobre a classe, suas instâncias, modificar atributos e métodos de uma classe, alterar campos (Barth 2000).

A Segunda Reflexão seria a reflexão comportamental, que leva mais a fundo sobre o comportamento de objetos, sem modificar a estrutura do objeto.

5.2 – Arquitetura Reflexiva

Com a reflexão permite que o código para executar operações que seriam ilegais em código anti-reflexo, tal como aceder campos e métodos particulares, o uso da reflexão pode resultar em efeitos secundários inesperados, que possam tornar código disfuncional e podem destruir a portabilidade. Código reflexivo quebra abstrações e, portanto, pode mudar o comportamento com atualizações da plataforma (Forman & Forman 2004).

Reflexão envolve tipos que são resolvidas dinamicamente, certos máquina virtual Java otimizações não pode ser executada. Consequentemente, as operações reflexivas ter um desempenho mais lento do que suas contrapartes não-reflexivas, e deve ser evitado em seções de código que são chamados com freqüência em aplicações sensíveis ao desempenho.

Reflexão requer uma permissão de execução, que não pode estar presente quando rodando sob um gerente de segurança. Esta é uma consideração importante no código que tem de ser executado em um contexto de segurança restrito, tal como em um applet.

A reflexão Computacional define conceitualmente uma arquitetura em níveis, denominada arquitetura reflexiva. Em uma arquitetura reflexiva, um sistema computacional é visto como incorporando dois componentes: um representando os objetos (domínio da aplicação), e outro a parte reflexiva (domínio reflexivo ou autodomínio) (Yamaguti, 2002).

Forman & Forman 2004, diz que no nível-base são encontradas as funções dos objetos, essas computações dos objetos têm a funcionalidade de resolver problemas e retornar informações sobre o domínio externo, enquanto o nível reflexivo encontra-se no meta-nível, resolvendo os problemas do nível-base, e retorna informações sobre a computação do objeto, podendo adicionar funcionalidades extras a este objeto.

Em uma arquitetura reflexiva, tem uma visão que um sistema computacional tem incorporado uma parte objeto e outra parte reflexiva. Na parte da computação objeto é realizadas funções para resolver problemas e retornar informações sobre um domínio externo, e quanto ao nível reflexivo tem a funcionalidade de resolver problemas e retornar informação sobre a computação do objeto.

6. O uso de Tabela de Decisão Adaptativa no desenvolvimento de aplicações comerciais.

A tecnologia adaptativa na área comercial ainda não tem um nicho focado especificamente nisso, com isso a pretensão de se mostrar o uso de tecnologia adaptativa aplicada em uma aplicação comercial, com uma funcionalidade em nuvem, isso provará o quanto a tomada de decisão utilizando uma devida estrutura tornará muito mais funcional e inteligente tais ações tratadas como muito complexas e pouco funcionais.

A proposta é utilizar de Tomada de Decisões Adaptativos no processo de funções financeiras, executando ações básicas nas suas rotinas e trabalhos diários que sofram inserções de dados, sendo assim um apoio nas operações financeiras, otimizando e deixando de maneira mais compreensiva seu funcionamento adaptativo e suas tomadas de decisões. Baseando se em combinações de estratégias corporativas, dando a liberdade de que o sistema tome uma decisão mais eficaz e melhore a competitividade e seus respectivos processos pouco dinâmico, reforçando sempre o método estratégico competitivo, trazendo melhor condição de trabalho com o sistema, tornando sem si mais lucrativos, e não se preocupando com tais processos lentos e maçantes, com isso temos um modelo de negócio mais ágil, inteligente e decisivo sem muitas complexidades em regras de negócio, deixo isso totalmente para aplicação, relacionando quais das melhores decisões deve-se tomar sem afetar qualquer coisa que possa ser prejudicial para a empresa.

Segundo o Simples Nacional 2015, os mercados de trabalho adotam várias e diversas estratégias em questão de negócios, sendo competitivos e sempre querendo prioridade quando a questão é negócios, e tomar decisões nesse meio é de veras, complicado e requer um estudo prático, que levaria tempo, exigiria uma série de regras e necessidades de análise. O objetivo da tecnologia é tornar essas ações mais práticas, e a inteligência artificial para deixá-las mais inteligente, o foco em utilizar tecnologia adaptativa e tabelas de decisões é unificar esses conceitos e deixar uma aplicação tomar decisões sérias e muitos mais funcionais, levando em conta uma série de regras e ações pré-

determinadas, focando principalmente em qualidade, agilidade, confiabilidade e produtividade, fazendo com que o sistema proposto possa conter essas qualidades e de poder manter a capacidade financeira produzindo e desenvolvendo com maior dinamismo e flexibilidade.

Sendo assim os mecanismos de Tabela de Decisão Adaptativa os diversos critérios para tais tomada de decisão, deriva-se de tabelas de decisão convencionais com ações adaptativas e procedimentos de métodos multicritério. É mostrado como esses dispositivos, pelo seu caráter genérico e com aplicabilidade ampla, podem apoiar na tomada de decisão, diante de critérios estabelecidos a cada ciclo do processo decisório, embora o modelo da tomada de decisão proposto seja uma ferramenta para a tomada de decisões em um módulo financeiro, as decisões ainda devem ser tomadas pelos profissionais responsáveis pela análise e são, portanto, sujeitas aos julgamentos de decisões humanas.

6.1 Desenvolvimento

A tecnologia que para se criar a aplicação é muito importante, pois é com ela que tais conceitos apresentados será provado e permitirá o usuário ter acesso de modo mais simplificado e funcional. A opção foi de se utilizar apenas tecnologia Open Sources, que são as tecnologias que pode se utilizar livremente sem nenhum custo para o programador.

Utiliza-se da linguagem Java para desenvolvimento Web, uma das ferramentas gratuitas mais utilizadas no mercado atual, o motivo obvio além da liberdade e do poder de desenvolvimento é que possui uma gama muito grande de materiais disponíveis nos principais meios, como livros, fóruns, blogs, vídeos, e em diversos formatos, sendo de uso multiplataforma, podendo ser desenvolvido em qualquer sistema operacional, Windows, Linux, Mac ou Unix e utilizado como aplicação nas mesmas.

6.2 Utilização do Simples Nacional

O Simples Nacional é um programa do governo federal que visa simplificar a burocracia das empresas.

As empresas enquadradas no programa possuem as rotinas e obrigações mensais facilitadas, além de uma carga tributária reduzida e unificada (um imposto único).

Embora o programa propõe uma guia única de imposto, as alíquotas diferem de acordo com a atividade exercida. As alíquotas iniciais variam de 4,5% até 16,93% sobre o valor bruto faturado.

Cada atividade permitida no Simples Nacional está enquadrada dentro de 1 dos 6 anexos do programa.

Por isso, é possível que uma empresa que possua mais de uma atividade tenha que pagar diferentes alíquotas de imposto.

Por exemplo, vamos supor uma empresa que possui:

Atividade primária: 6209-1/00 – Suporte Técnico, Manutenção e outros serviços em tecnologia da informação

Atividade secundária: CNAE 6201-5/00-Desenvolvimento de programas de computador sob encomenda

A atividade primária de Suporte Técnico está enquadrada no Anexo 3 e, portanto, tem alíquota inicial de 6% sobre o valor faturado. Já a atividade de Desenvolvimento de Programas de Computador está enquadrada no Anexo 5, com alíquota inicial de 17,5% + 2% de ISS (totalizando 19,5%).

Toda a vez que esta empresa emitir nota fiscal referente a atividade primária de suporte técnico, ela pagará 6% de imposto sobre o valor faturado.

No Simples Nacional, os impostos federais, estaduais e municipais são pagos em um único boleto. Além disso, a carga tributária das empresas participantes será reduzida em até 40%.

Entretanto, com as alterações, sancionadas no último dia 7, houve um aumento de atividades inseridas no regime tributário, e, desse modo, há uma série de intervalos de alíquotas de tributos.

Por exemplo, o setor de Comércio terá alíquotas de imposto que variam de 4%, quando o faturamento é de até R$ 180 mil; a 11,61%, se o faturamento da companhia em questão for de R$ 3,42 milhões. Já para o setor industrial, as alíquotas de imposto oscilam entre 4,5% e 12,11%. Confira as tabelas de alíquotas para o comércio e para a indústria.

Se a função exercida for relacionada à agência terceirizada de correios, de viagem e turismo; centro de formação de condutores; agências lotéricas; serviços de instalação, de reparos e de manutenção em geral; transportes interestaduais de cargas e intermunicipais (de cargas ou passageiros); escritórios de serviços contábeis; produções cinematográficas, audiovisuais, artísticas e culturais as faixas de tributação irão de 6% até 17,42%.

Para os serviços advocatícios; de construção de imóveis; de obras de engenharia em geral; de execução de projetos; de paisagismo; de decoração de interiores; de vigilância, limpeza ou conservação, as taxas de imposto cobradas vão de 4,5% até 16,85%.

As atividades de medicina (laboratorial, enfermagem e veterinária); odontologia; psicologia; psicanálise, terapia ocupacional; acupuntura; podologia; fonoaudiologia; ligadas a clínicas (de nutrição e de vacinação) e bancos de leite; serviços de comissária de despachantes; tradução; interpretação; engenharia; arquitetura; medição, cartografia, topografia, geologia, geodésia; testes, suportes e análises técnicas e tecnológicas; pesquisas; design, desenho; agronomia; de representação comercial; de perícia, de leilão e avaliação; auditoria; economia; consultoria; gestão, organização, controle e administração; jornalismo; publicidade; agenciamento, exceto de mão de obra; e serviços que tenham por finalidade o exercício de atividade intelectual têm alíquotas que variam entre 16,93% e 22,45%.

TABELA DO SIMPLES NACIONAL

Receita Bruta em 12 meses (em R$)	Alíquota	IRPJ	CSLL	Cofins	PIS/Pasep	CPP	ICMS
Até 180.000,00	4,00%	0,00%	0,00%	0,00%	0,00%	2,75%	1,25%
De 180.000,01 a 360.000,00	5,47%	0,00%	0,00%	0,86%	0,00%	2,75%	1,86%
De 360.000,01 a 540.000,00	6,84%	0,27%	0,31%	0,95%	0,23%	2,75%	2,33%
De 540.000,01 a 720.000,00	7,54%	0,35%	0,35%	1,04%	0,25%	2,99%	2,56%
De 720.000,01 a 900.000,00	7,60%	0,35%	0,35%	1,05%	0,25%	3,02%	2,58%
De 900.000,01 a 1.080.000,00	8,28%	0,38%	0,38%	1,15%	0,27%	3,28%	2,82%
De 1.080.000,01 a 1.260.000,00	8,36%	0,39%	0,39%	1,16%	0,28%	3,30%	2,84%
De 1.260.000,01 a 1.440.000,00	8,45%	0,39%	0,39%	1,17%	0,28%	3,35%	2,87%
De 1.440.000,01 a 1.620.000,00	9,03%	0,42%	0,42%	1,25%	0,30%	3,57%	3,07%
De 1.620.000,01 a 1.800.000,00	9,12%	0,43%	0,43%	1,26%	0,30%	3,60%	3,10%
De 1.800.000,01 a 1.980.000,00	9,95%	0,46%	0,46%	1,38%	0,33%	3,94%	3,38%
De 1.980.000,01 a 2.160.000,00	10,04%	0,46%	0,46%	1,39%	0,33%	3,99%	3,41%
De 2.160.000,01 a 2.340.000,00	10,13%	0,47%	0,47%	1,40%	0,33%	4,01%	3,45%
De 2.340.000,01 a 2.520.000,00	10,23%	0,47%	0,47%	1,42%	0,34%	4,05%	3,48%
De 2.520.000,01 a 2.700.000,00	10,32%	0,48%	0,48%	1,43%	0,34%	4,08%	3,51%
De 2.700.000,01 a 2.880.000,00	11,23%	0,52%	0,52%	1,56%	0,37%	4,44%	3,82%
De 2.880.000,01 a 3.060.000,00	11,32%	0,52%	0,52%	1,57%	0,37%	4,49%	3,85%
De 3.060.000,01 a 3.240.000,00	11,42%	0,53%	0,53%	1,58%	0,38%	4,52%	3,88%
De 3.240.000,01 a 3.420.000,00	11,51%	0,53%	0,53%	1,60%	0,38%	4,56%	3,91%
De 3.420.000,01 a 3.600.000,00	11,61%	0,54%	0,54%	1,60%	0,38%	4,60%	3,95%

Tabela 1: Tabela do Simples Nacional (Planalto Vigência a Partir de 01.01.2012)

A princípio, a maioria das Micro e Pequenas Empresas, que está dentro dos parâmetros da Lei Geral para Micro e Pequenas Empresas, pode fazer parte do Simples.

Resumidamente, nesse grupo estão:

Microempresa - pessoa jurídica que fatura até R$ 240 mil ao ano

Pequena Empresa - pessoa jurídica que fatura mais de R$ 240 mil até R$ 2,4 milhões ao ano

O Simples, no entanto, prevê também outros parâmetros, de acordo com o Produto Interno Bruto (PIB) Estadual. Assim:

Estados (e seus respectivos municípios) com participação em até 1% do PIB (atualmente RO, AC, RR, AP, TO, MA, PI, RN, PB, AL, SE) poderão adotar o limite de R$ 1,2 milhão para as pequenas empresas;

Estados (e seus respectivos municípios) com participação em até 5% do PIB (atualmente AM, PA, CE, PE, BA, ES, SC, MT, MS, GO e DF) poderão adotar o limite de R$ 1,8 milhão para pequenas empresas;

Estados (e seus respectivos municípios) com participação acima de 5% do PIB (atualmente MG, RJ, SP, PR e RS) terão o limite de R$ 2,4 milhões para as pequenas empresas.

Esta limitação, no entanto, depende da regulamentação em cada um dos Estados. Alguns Estados já tem suas leis de incentivo a micro ou pequenas empresas, principalmente em relação ao ICMS, como Alagoas, Bahia, Ceará, Minas Gerais, Paraná, Pernambuco, Rio Grande do Sul, Rio de Janeiro, Santa Catarina e São Paulo. A adaptação vai depender do trabalho do Comitê Gestor de Micro e Pequenas Empresas (Receita 2015).

Alguns destes perfis, no entanto, podem se encaixar no Simples se a empresa obedecer alguns desses critérios:

Mesmo que a empresa esteja nessa última lista, é importante calcular ou pedir para o seu contador calcular direitinho e saber se realmente vale a pena aderir ao sistema. Outro detalhe: se a empresa tem uma atividade que pode ser

contemplada no Simples e outra que não pode, ela fica sem o direito de inclusão no benefício.

O Simples unifica oito impostos:

Imposto de Renda da Pessoa Jurídica (IPRJ)

Imposto sobre Produtos Industrializados (IPI)

Contribuição Social sobre o Lucro Líquido (CSLL)

Contribuição para o Financiamento da Seguridade Social (Cofins)

PIS/Pasep

Contribuição para o INSS

Imposto sobre Serviços de Qualquer Natureza (ISS)

Imposto sobre Operações Relativas à Circulação de Mercadorias e Sobre Prestações de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação (ICMS) (Receita 2015).

É bom lembrar que estes dois últimos impostos são de responsabilidade dos governos estadual e municipal, o que, como já foi dito, vai depender de legislação específica para serem incluídos no Simples, apesar de alguns estados já terem essa legislação. Além da unificação dos impostos, quem estiver no sistema não é obrigado a pagar contribuições instituídas pelas entidades de serviço social autônomo como Sebrae, Senac ou Sesc, além de contribuição sindical patronal ou qualquer imposto sindical previsto na CLT.

Outros impostos, no entanto, vão continuar a ser cobrados a parte, entre eles:

Imposto sobre Operações de Crédito, Câmbio e Seguros (IOF)

Imposto de Importação

PIS, Cofins e IPI na importação

Imposto de Exportação

Imposto Sobre a Propriedade Territorial Rural (IPTR)

INSS relativo ao empregado e ao empresário

Contribuição Provisória sobre a Movimentação ou Transmissão de Valores e de Créditos e Direitos de Natureza Financeira (CPMF)

Fundo de Garantia por Tempo de Serviço (FGTS) para os empregas

ICMS e ISS em regime de substituição tributária e na importação.

A primeira grande mudança entre o Simples, de 1996, e o Simples, de 2007, é a base de cálculo das alíquotas do imposto. No modelo anterior, as empresas calculavam a alíquota do imposto em cima da receita bruta acumulada durante o ano em exercício. Ou seja, em maio daquele ano, a alíquota do imposto era calculada pela receita bruta desde janeiro. Agora, a porcentagem de imposto a pagar deve ser medida em cima da receita acumulada nos últimos doze meses. Ou seja, em maio, calcula-se a alíquota em cima da receita de junho de um ano a maio do ano corrente.

Tal medida já provoca uma certa controvérsia entre os contadores. Para alguns, essa medida aumenta o imposto a ser pago. Outros acreditam que a medida não implica necessariamente em aumento de imposto já que as alíquotas são diferentes do modelo anterior. Mais uma vez, cada caso é um caso e deve ser olhado de maneira particular.

As alíquotas do Simples serão medidas de acordo com o serviço e setor que a empresa está atuando. O que chamamos nos últimos parágrafos de receita bruta se refere à:

Revenda de mercadorias;

Venda de mercadorias industrializadas pelo contribuinte;

Prestação de serviços, bem como da locação de bens móveis;

Venda de mercadorias sujeitas à substituição tributária;

Exportação de mercadorias, inclusive as vendas realizadas por meio de comercial exportadora ou do consórcio previsto na lei geral.

Comércio, indústria e serviços pagam impostos diferentes de acordo com o Simples. Além, obviamente, da variação de acordo com a receita, cada um tem

uma tabela diferente, que aqui estão enumeradas de 1 a 5. Atenção: algumas empresas deverão usar mais de uma tabela.

Veja qual a alíquota da sua empresa:

Para o comércio, as alíquotas variam de 4 a 11,61%, conforme a tabela 2.

Para indústria, as alíquotas variam de 4,5 a 12,11%, conforme tabela 3.

TABELA DO SIMPLES NACIONAL– INDUSTRIA

Receita Bruta em 12 meses (em R$)	ALÍQUOTA	IRPJ	CSLL	COFINS	PIS/PASEP	CPP	ICMS	IPI
Até 120.000,00	4,50%	0,00%	0,00%	0,00%	0,00%	2,75%	1,25%	0,50%
De 120.000,01 a 240.000,00	5,97%	0,00%	0,00%	0,86%	0,00%	2,75%	1,86%	0,50%
De 240.000,01 a 360.000,00	7,34%	0,27%	0,31%	0,95%	0,23%	2,75%	2,33%	0,50%
De 360.000,01 a 480.000,00	8,04%	0,35%	0,35%	1,04%	0,25%	2,99%	2,56%	0,50%
De 480.000,01 a 600.000,00	8,10%	0,35%	0,35%	1,05%	0,25%	3,02%	2,58%	0,50%
De 600.000,01 a 720.000,00	8,78%	0,38%	0,38%	1,15%	0,27%	3,28%	2,82%	0,50%
De 720.000,01 a 840.000,00	8,86%	0,39%	0,39%	1,16%	0,28%	3,30%	2,84%	0,50%
De 840.000,01 a 960.000,00	8,95%	0,39%	0,39%	1,17%	0,28%	3,35%	2,87%	0,50%
De 960.000,01 a 1.080.000,00	9,53%	0,42%	0,42%	1,25%	0,30%	3,57%	3,07%	0,50%
De 1.080.000,01 a 1.200.000,00	9,62%	0,42%	0,42%	1,26%	0,30%	3,62%	3,10%	0,50%
De 1.200.000,01 a 1.320.000,00	10,45%	0,46%	0,46%	1,38%	0,33%	3,94%	3,38%	0,50%
De 1.320.000,01 a 1.440.000,00	10,54%	0,46%	0,46%	1,39%	0,33%	3,99%	3,41%	0,50%
De 1.440.000,01 a	10,63%	0,47%	0,47%	1,40%	0,33%	4,01	3,45%	0,50%

1.560.000,00						%		
De 1.560.000,01 a 1.680.000,00	10,73%	0,47%	0,47%	1,42%	0,34%	4,05 %	3,48%	0,50%
De 1.680.000,01 a 1.800.000,00	10,82%	0,48%	0,48%	1,43%	0,34%	4,08 %	3,51%	0,50%
De 1.800.000,01 a 1.920.000,00	11,73%	0,52%	0,52%	1,56%	0,37%	4,44 %	3,82%	0,50%
De 1.920.000,01 a 2.040.000,00	11,82%	0,52%	0,52%	1,57%	0,37%	4,49 %	3,85%	0,50%
De 2.040.000,01 a 2.160.000,00	11,92%	0,53%	0,53%	1,58%	0,38%	4,52 %	3,88%	0,50%
De 2.160.000,01 a 2.280.000,00	12,01%	0,53%	0,53%	1,60%	0,38%	4,56 %	3,91%	0,50%
De 2.280.000,01 a 2.400.000,00	12,11%	0,54%	0,54%	1,60%	0,38%	4,60 %	3,95%	0,50%

Tabela 3 - Planilha de Simples Nacional – Indústria (Planalto Vigência a Partir de 01.01.2012)

A substituição tributária, é um mecanismo tributário que possibilita o pagamento dos impostos de toda a cadeia produtiva e comercial de uma só vez por um dos integrantes da cadeia. Por exemplo, ao sair da fábrica, uma garrafa de refrigerante já paga o PIS, Cofins, ICMS não só dela mas do comerciante que vai vender o produto. O Simples prevê que esses impostos sejam abatidos, assim como no caso da exportação, da tabela de impostos principal. Se parte da sua receita veio de exportações ou de substituição tributária, é possível subtrair os impostos que já foram pagos por outrem. Os impostos que podem ser substituídos são PIS, Cofins, ICMS e IPI (Simples Nacional 2015).

Além de todos esses detalhes, é importante ficar atento se sua empresa não vai acabar ultrapassando os limites previstos na lei geral. Por exemplo, se a receita ultrapassar os R$ 2,4 milhões anuais, a parcela que exceder estará sujeita às alíquotas máximas de cada tabela, somando ainda mais 20%. Apesar do nome, é fácil perceber que o Simples não é tão simples assim. A principal facilidade do sistema é a unificação dos impostos, mas a adesão ou não ao sistema vai depender de uma análise cuidadosa de sua empresa. Se ela, por exemplo, não for microempresa, é possível que não seja tão simples assim (Simples Nacional 2015).

7 – Desenvolvimento do Estudo de Caso

A Aplicação consistem em um Sistema Integrado de Pagamento de Impostos e Contribuições das Microempresas e Empresas de Pequeno Porte (Simples) é um regime tributário diferenciado, simplificado e favorecido, aplicável às pessoas jurídicas consideradas como microempresas (ME) e empresas de pequeno porte (EPP). Constitui-se em uma forma simplificada e unificada de recolhimento de tributos, por meio da aplicação reflexiva com metadados de percentuais favorecidos e progressivos, incidentes sobre uma única base de cálculo, a receita bruta.

O programa constitui de classes java que são:

Classe ClienteDAO, classe responsável por passar todos os parâmetros e fazer comunicação com a reflexão e classe principal, essa classe ela se estende a todas outras para utilização dos seus meta dados e também responsável por manter os dados do cliente.

```
package bcc.fema.edu.dao;

import java.io.Serializable;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Date;

import bcc.fema.edu.annotations.AdaptativoAnotation;

@AdaptativoAnotation(nome = "cliente")
public class ClienteDAO extends Cliente implements Serializable, ClienteInterface {

	@AdaptativoAnotation(nome = "codigoRegistro")
	private long cliCodigo;
	@AdaptativoAnotation(nome = "nome-completo", hasCData = true)
	private String nome;
	@AdaptativoAnotation(nome = "sexo")
	private String sexo;
	@AdaptativoAnotation(nome = "dataNascimento", hasCData = true, mascara =
"dd/MM/yyyy HH:mm:ss", tipo = Date.class)
	private java.util.Date dataNascimento;
	@AdaptativoAnotation(nome = "situacao")
	private boolean situacao;
	@AdaptativoAnotation(nome = "servico-cliente", isClasse = true)
	private ServicoDAO servico;
	@AdaptativoAnotation(nome = "total", hasCData = true, mascara = "#.00", tipo =
BigDecimal.class)
	private BigDecimal total;
	@AdaptativoAnotation(nome = "totalPorcentagem", hasCData = true, tipo = float.class)
	private Float totalPorcentagem;
```

```
	public ClienteDAO() {
		super();
	}

	public ClienteDAO(long codigo, String nome) {
		super();
		this.cliCodigo = codigo;
		this.nome = nome;
	}

	public ClienteDAO(long ra, String nome, String sexo, Date dataNascimento, boolean
ativo, ServicoDAO servico,
				float totalPorcentagem) {
		super();
		this.cliCodigo = ra;
		this.nome = nome;
		this.sexo = sexo;
		this.dataNascimento = dataNascimento;
		this.situacao = ativo;
		this.servico = servico;
		this.totalPorcentagem = totalPorcentagem;
	}
```

Fonte1 – ClasseDAO.java

EmpresaDAO, responsável pela situação da empresa, se está habilitada pela Simples Nacional.

```
package bcc.fema.edu.dao;

import java.io.Serializable;
import java.util.Date;

import bcc.fema.edu.annotations.AdaptativoAnotation;

@AdaptativoAnotation(nome = "empresa")
public class EmpresaDAO implements Serializable {

	@AdaptativoAnotation(nome = "contrato")
	private long contrato;
	@AdaptativoAnotation(nome = "nome", hasCData = true)
	private String nome;
	@AdaptativoAnotation(nome = "tipo-empresa", hasCData = true)
	private String tipo;
	@AdaptativoAnotation(nome = "empTipo")
	private String empTipo;
	@AdaptativoAnotation(nome = "dataContrato", hasCData = true)
	private java.util.Date dataContrato;
	@AdaptativoAnotation(nome = "ativo")
	private boolean ativo;

	public EmpresaDAO() {
		super();
	}
```

```
	public EmpresaDAO(long contrato, String nome) {
		super();
		this.contrato = contrato;
		this.nome = nome;
	}

	public EmpresaDAO(long codigo, String nome, String tipo, String sexo, Date
dataContrato, boolean ativo) {
		super();
		this.contrato = codigo;
		this.nome = nome;
		this.tipo = tipo;
		this.empTipo = sexo;
		this.dataContrato = dataContrato;
		this.ativo = ativo;
	}
```

Fonte2 – EmpresaDAO.java

ServicoDAO, classe responsável pela analise dos valores da empresa.

```
package bcc.fema.edu.dao;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.List;

import bcc.fema.edu.annotations.AdaptativoAnotation;

@AdaptativoAnotation(nome = "servico")
public class ServicoDAO implements Serializable {

	@AdaptativoAnotation(nome = "codigo")
	private long codigo;
	@AdaptativoAnotation(nome = "descricao-completa", hasCData = true)
	private String descricao;
	@AdaptativoAnotation(nome = "sigla", hasCData = true)
	private String sigla;
	@AdaptativoAnotation(nome = "tipo", isClasse = true)
	private List<TipoDAO> tipo;
	@AdaptativoAnotation(nome = "total", hasCData = true, tipo = BigDecimal.class)
	private BigDecimal total;

	public ServicoDAO() {
		super();
	}

	public ServicoDAO(long codigo, String descricao) {
		super();
		this.codigo = codigo;
		this.descricao = descricao;
	}

	public ServicoDAO(long codigo, String descricao, String sigla, List<TipoDAO> servico) {
		super();
```

```
		this.codigo = codigo;
		this.descricao = descricao;
		this.sigla = sigla;
		this.tipo = servico;
	}
```

Fonte 3 – ServicoDAO.java

TipoDAO, classe responsável pela descrição do serviço.

```
package bcc.fema.edu.dao;

import java.io.Serializable;

import bcc.fema.edu.annotations.AdaptativoAnotation;

@AdaptativoAnotation(nome = "tipo")
public class TipoDAO implements Serializable {

	@AdaptativoAnotation(nome = "codigo")
	private long codigo;
	@AdaptativoAnotation(nome = "descricao-completa", hasCData = true)
	private String descricao;
	@AdaptativoAnotation(nome = "sigla", hasCData = true)
	private String sigla;
	@AdaptativoAnotation(nome = "qtde-servico")
	private int qtdeServico;
	@AdaptativoAnotation(nome = "empresa", isClasse = true)
	private EmpresaDAO empresa;

	public TipoDAO() {
		super();
	}

	public TipoDAO(long codigo, String descricao) {
		super();
		this.codigo = codigo;
		this.descricao = descricao;
	}

	public TipoDAO(long codigo, String descricao, String sigla, int qtdeServico, EmpresaDAO
empresa) {
		super();
		this.codigo = codigo;
		this.descricao = descricao;
		this.sigla = sigla;
		this.qtdeServico = qtdeServico;
		this.empresa = empresa;
	}
```

Fonte 4 – TipoDAO.java

ClassReflection, classe que analisa todas as outras, para ver o tempo de execução, quais o processos de bytecode e comportamento previsto na execução, isso aplica-se a reflexão diretamente na execução do comportamento da aplicação.

```
package bcc.fema.edu.principal;

import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.math.BigDecimal;
import java.text.SimpleDateFormat;
import java.util.Date;

import bcc.fema.edu.dao.ClienteDAO;
import bcc.fema.edu.dao.ServicoDAO;
import bcc.fema.edu.dao.ClienteInterface;
import bcc.fema.edu.dao.EmpresaDAO;

public class ClassReflection {

	public static void main(String[] args) {
		try {
			ClienteDAO cliente1 = new ClienteDAO(1L, "Nome do Usuário Para
Teste");
			cliente1.setDataNascimento(new java.util.Date());

			Class<?> klassInterface = ClienteInterface.class;

			System.out.println(klassInterface.getPackage());

			// Class<?> klass1 = aluno1.getClass();
			Class<?> klass2 = ClienteDAO.class;
			Class<?> klass3 = Class.forName("bcc.fema.edu.dao.ClienteDAO");

			System.out.println("Name: " + klass2.getName());
			System.out.println("Simple Name: " + klass2.getSimpleName());
			System.out.println("Package: " + klass2.getPackage());

			System.out.println("Modifier: " + klass2.getModifiers());
			System.out.println("Modifier: " + Modifier.toString(klass2.getModifiers()));

			if (klass2.getInterfaces() != null) {
				for (Class<?> classInterface : klass2.getInterfaces()) {
					System.out.println("Interface: " +
classInterface.getName());
				}
			}

			Class<?> superclasse = klass2.getSuperclass();

			System.out.println(superclasse.getSimpleName());
```

```
System.out.println(superclasse.getPackage());
System.out.println(Modifier.toString(superclasse.getModifiers()));

System.out.println(klass2.isInstance(cliente1));
// bcc instanceof ClienteDAO

// Field[] fields = klass2.getFields();
Field[] fields = klass2.getDeclaredFields();
// Method[] methods = klass2.getMethods();
Method[] methods = klass2.getDeclaredMethods();
// Constructor<?>[] constructors = klass2.getConstructors();
Constructor<?>[] constructors = klass2.getDeclaredConstructors();

if (fields != null) {
	for (Field field : fields) {
		System.out.println(field);
	}
}

if (methods != null) {
	for (Method method : methods) {
		System.out.println(method);
	}
}

if (constructors != null) {
	for (Constructor<?> constructor : constructors) {
		System.out.println(constructor);
	}
}

System.out.println();

// ################################

Field fieldNome = klass2.getDeclaredField("nome");

fieldNome.setAccessible(true);

System.out.println(fieldNome.get(cliente1));

fieldNome.set(cliente1, "Nome do cliente para Teste");

System.out.println(fieldNome.get(cliente1));

Field fieldDataNascimento = klass2.getDeclaredField("dataNascimento");

fieldDataNascimento.setAccessible(true);

System.out.println(fieldDataNascimento.get(cliente1));

fieldDataNascimento.set(cliente1, new
SimpleDateFormat("dd/MM/yyyy").parse("16/06/1995"));

System.out.println(fieldDataNascimento.get(cliente1));

ServicoDAO srv = new ServicoDAO(1001L, "Tipo do Serviço");
```

```
                        Field fieldServico = klass2.getDeclaredField("servico");

                        System.out.println(fieldServico.getType());
                        System.out.println(fieldServico.getModifiers());
                        System.out.println(Modifier.toString(fieldServico.getModifiers()));
                        System.out.println(fieldServico.getDeclaringClass());

                        fieldServico.setAccessible(true);

                        fieldServico.set(cliente1, srv);

                        System.out.println(fieldServico.get(cliente1));

                        // ################################

                        Method methodCalcularMedia =
klass2.getDeclaredMethod("calculaTotal", BigDecimal.class, BigDecimal.class);
                        // var args = varargs

                        // aluno1.calcularMedia(bg1, bg2);
                        BigDecimal total = (BigDecimal) methodCalcularMedia.invoke(cliente1,
new BigDecimal("8.7"),
                                        new BigDecimal("9.5"));

                        System.out.println(total);

                        System.out.println(methodCalcularMedia.getModifiers());

        System.out.println(Modifier.toString(methodCalcularMedia.getModifiers()));

                        System.out.println("Retorno: " + methodCalcularMedia.getReturnType());

                        Class<?>[] parametersType =
methodCalcularMedia.getParameterTypes();

                        if (parametersType != null) {
                                for (Class<?> parameterType : parametersType) {
                                        System.out.println(parameterType);
                                }
                        }

                        Class<?>[] exceptionsType = methodCalcularMedia.getExceptionTypes();

                        if (exceptionsType != null) {
                                for (Class<?> exceptionType : exceptionsType) {
                                        System.out.println(exceptionType);
                                }
                        }

                        // ################################

                        Constructor<?> construtorCliente =
klass2.getDeclaredConstructor(long.class, String.class);

                        System.out.println(construtorCliente);

                        System.out.println(construtorCliente.getModifiers());
                        System.out.println(Modifier.toString(construtorCliente.getModifiers()));
```

```
                        // parametersType (method são equivalentes ao constructor)
                        // exceptionsType (method são equivalentes ao constructor)

                        parametersType = construtorCliente.getParameterTypes();

                        if (parametersType != null) {
                                for (Class<?> parameterType : parametersType) {
                                        System.out.println(parameterType);
                                }
                        }

                        exceptionsType = construtorCliente.getExceptionTypes();

                        if (exceptionsType != null) {
                                for (Class<?> exceptionType : exceptionsType) {
                                        System.out.println(exceptionType);
                                }
                        }

                        Object novoCliente = (Object) construtorCliente.newInstance(400l,
"Nome do Cliente para do construtor");
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

Fonte 5 - ClasseReflection.java

A Classe Principal é responsável por utilizar de todas as outras classes aplicando os conceitos desenvolvido em todo o processo. Essa classe aproveita de todas as demais para um único resultado em tempo de exceção. Essa classe utiliza-se de todas as outras estanciando uma nova classe e importando todas as demais.

Com a criação do uma nova instância, passa-se todos os valores de cada classe, ao qual foi estendida entre as classes anteriores. Após a inserção dos dados, a estrutura cria um serviço e reflete os dados para a classe que identifica as DAO's, a cada criação de serviços ele utiliza a reflexão para a criação de meta dados e gerando uma Lista para representar o resultado de forma dinâmica a nível de execução, tornando o comportamento reaproveitado assim que a chamada do processo for executada.

```
package bcc.fema.edu.principal;

import java.io.BufferedWriter;
import java.io.File;
```

```
import java.io.FileWriter;
import java.math.BigDecimal;
import java.util.LinkedList;
import java.util.List;

import com.sun.org.apache.xerces.internal.util.XMLSymbols;

import bcc.fema.edu.dao.ClienteDAO;
import bcc.fema.edu.dao.ServicoDAO;
import bcc.fema.edu.dao.TipoDAO;
import bcc.fema.edu.util.ExportaXML;
import bcc.fema.edu.dao.EmpresaDAO;

public class Principal {


	public static Float totalPorcentagem;

	public static void main(String[] args) {

		ClienteDAO cliente = new ClienteDAO();
		cliente.setCliCodigo(000001);
		cliente.setNome("Nome do Cliente 1");
		cliente.setDataNascimento(new java.util.Date());
		cliente.setSexo("M");
		cliente.setServico(null);
		cliente.setTotal(new BigDecimal("12.000"));
		cliente.setTeste("Porcentagem do ICMS: ", totalPorcentagem = 12.000f);

		ServicoDAO srv = new ServicoDAO();
		srv.setCodigo(001);
		srv.setDescricao("Serviços 1");
		srv.setSigla("SRV");
		srv.setTipo(null);

		cliente.setServico(srv);

		EmpresaDAO Empresa1 = new EmpresaDAO();
		Empresa1.setContrato(001l);
		Empresa1.setNome("Empresa 1");
		Empresa1.setTipo("1");
		Empresa1.setDataContrato(new java.util.Date());
		Empresa1.setEmpTipo("S");
		Empresa1.setAtivo(true);

		TipoDAO configuracaoServidor = new TipoDAO();
		configuracaoServidor.setCodigo(1l);
		configuracaoServidor.setDescricao("Servidor Windows Server");
		configuracaoServidor.setSigla("ServeWin");
		configuracaoServidor.setEmpresa(Empresa1);
		configuracaoServidor.setQtdeServico(2);

		TipoDAO vendaSite = new TipoDAO();
		vendaSite.setCodigo(0l);
		vendaSite.setDescricao("Site PHP");
		vendaSite.setSigla("WebPHP");
		vendaSite.setEmpresa(Empresa1);
		vendaSite.setQtdeServico(1);
```

```
                // Teste 02

                ClienteDAO cliente2 = new ClienteDAO();
                cliente2.setCliCodigo(000002);
                cliente2.setNome("Nome do Cliente 2");
                cliente2.setDataNascimento(new java.util.Date());
                cliente2.setSexo("M");
                cliente2.setServico(null);
                cliente2.setTotal(new BigDecimal("200.000"));
                cliente.setTeste("Porcentagem do ICMS: ", totalPorcentagem = 200.000f );
//              cliente.setTeste(" % ",total2);

                ServicoDAO srv2 = new ServicoDAO();
                srv2.setCodigo(2l);
                srv2.setDescricao("Servi�os 1");
                srv2.setSigla("SRV");
                srv2.setTipo(null);

                cliente2.setServico(srv2);

                EmpresaDAO Empresa2 = new EmpresaDAO();
                Empresa2.setContrato(002l);
                Empresa2.setNome("Empresa 2");
                Empresa2.setTipo("2");
                Empresa2.setDataContrato(new java.util.Date());
                Empresa2.setEmpTipo("S");
                Empresa2.setAtivo(true);

                TipoDAO configuracaoServidor2 = new TipoDAO();
                configuracaoServidor2.setCodigo(2l);
                configuracaoServidor2.setDescricao("Servidor Linux");
                configuracaoServidor2.setSigla("ServeWin");
                configuracaoServidor2.setEmpresa(Empresa2);
                configuracaoServidor2.setQtdeServico(2);

                TipoDAO vendaSite2 = new TipoDAO();
                vendaSite2.setCodigo(2l);
                vendaSite2.setDescricao("Site PHP");
                vendaSite2.setSigla("WebPHP");
                vendaSite2.setEmpresa(Empresa2);
                vendaSite2.setQtdeServico(1);

//              List<TipoDAO> tipo2 = new LinkedList<TipoDAO>();
//
//              tipo2.add(configuracaoServidor2);
//              tipo2.add(vendaSite2);
//
//              srv2.setTipo(tipo2);

                // Teste 03

                ClienteDAO cliente3 = new ClienteDAO();
                cliente3.setCliCodigo(000003);
                cliente3.setNome("Teste 3");
                cliente3.setDataNascimento(new java.util.Date());
```

```
                cliente3.setSexo("F");
                cliente3.setServico(null);
                cliente3.setTotal(new BigDecimal("500.000"));
                cliente.setTeste("Porcentagem do ICMS: " , totalPorcentagem = 500.000f );
//              cliente.setTeste(" % ",total3);

                ServicoDAO srv3 = new ServicoDAO();
                srv3.setCodigo(3l);
                srv3.setDescricao("Serviços 5");
                srv3.setSigla("SRV-5");
                srv3.setTipo(null);

                cliente3.setServico(srv3);

                EmpresaDAO Empresa3 = new EmpresaDAO();
                Empresa3.setContrato(003l);
                Empresa3.setNome("Empresa 3 Sobre Desenvolvimento Web");
                Empresa3.setTipo("3");
                Empresa3.setDataContrato(new java.util.Date());
                Empresa3.setEmpTipo("S");
                Empresa3.setAtivo(true);

                TipoDAO sistemaCShap3 = new TipoDAO();
                sistemaCShap3.setCodigo(3l);
                sistemaCShap3.setDescricao("Sistema Publico C#");
                sistemaCShap3.setSigla("Pub C#");
                sistemaCShap3.setEmpresa(Empresa3);
                sistemaCShap3.setQtdeServico(4);

                TipoDAO vendaSite3 = new TipoDAO();
                vendaSite3.setCodigo(3l);
                vendaSite3.setDescricao("Site ASPX");
                vendaSite3.setSigla("WebASPX");
                vendaSite3.setEmpresa(Empresa3);
                vendaSite3.setQtdeServico(1);

//              List<TipoDAO> tipo3 = new LinkedList<TipoDAO>();
//
//              tipo3.add(sistemaCShap3);
//              tipo3.add(vendaSite3);
//
//              srv3.setTipo(tipo3);


                List<TipoDAO> tipo = new LinkedList<TipoDAO>();
                tipo.add(configuracaoServidor);
                tipo.add(vendaSite);
                tipo.add(configuracaoServidor2);
                tipo.add(vendaSite2);
                tipo.add(sistemaCShap3);
                tipo.add(vendaSite3);

                srv.setTipo(tipo);

        }
        }
```

Fonte 6 - Principal.java

8 – Conclusão e Trabalhos Futuros

Havendo capacidade intuitiva decorrente de sua experiência, pode criar um ato mais direcionada ao seu objetivo no atendimento das necessidades de suprimento de forma eficiente e eficaz, sendo visionário em diminuir o impacto das incertezas, através da utilização de processos sistemáticos de tomada de decisão.

Tal modelo de projeto apresentado em sua fase de desenvolvimento experimental, tem comportamentos confiáveis em seu modelo adaptativo que deverá ser alcançado atendendo as exigências matemáticas proposta com maior agilidade, destacando em uma maior economia de código, visando as áreas comerciais explorando novos meios tecnológicos com isso, conseguimos desfrutar de uma maior agilidade em seus processos e uma nova experiências aplicadas diretamente ao software.

Desta maneira, acredita-se que o modelo da Tabela de Decisão Adaptativa e seus multicritérios para tomada de decisão trará contribuições de grande responsabilidade e um retorno generosamente satisfatório, tanto nos aspectos práticos de aplicação da Tecnologia Adaptativa quanto nos processos de tomada de decisão, abrindo um grande leque para estudos mais detalhado para quem em um futuro próximo possa ser aprofundado mais diretamente um novo e melhor uso dessa tecnologia.

Referência Bibliográfica.

A. H. Tchemra, R. Camargo - **Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão:Uma Abordagem Estratégica na Seleção de Fornecedores -** WTA 2008 – Segundo Workshop de Tecnologia Adaptativa

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.

BARTH, F.J. **Utilização da Reflexão Computacional para implementação de aspectos não funcionais em um gerenciador de arquivos distribuídos.** Trabalho de Conclusão de Curso, Universidade Regional de Blumenau. 2000

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.; 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.

DEITEL, H. M.; DEITEL, P. J. **Java: Como Programar**. 6. Ed. São Paulo: PearsonEducation, 2005.

Forman, Ira R,; Forman, Nate - **Java Reflection in Action (In Action series)** - Manning Publications Co. Greenwich, CT, USA ©2004

FREITAS, A.V. ; NETO, J.J. **Uma Ferramenta para Construção de Aplicações Multilinguagens de Programação**. CACIC'2001 - Congreso Argentino de Ciencias de la Computación, 15-20 Outubro, El Calafate, Argentina, 2001.

FREITAS, A.V. **Aspectos do projeto e implementação de ambientes multilinguagens**

de programação. Dissertação de Mestrado, USP, São Paulo, 2000.

FREITAS, A.V.; NETO, J.J. **Ambiente Multilinguagem de Programação - Aspectos do Projeto e Implementação.** Boletim Técnico PBT/PCS/0109, ISSN 1413, 215X, Escola Politécnica, São Paulo, 2000b.

FREITAS, A.V.; NETO, J.J. **Aspectos da Implementação de um Ambiente Multilinguagem de Programação.** Anais do CACIC2000 - VI Congreso Argentino de Ciencias de la Computación. Ushuaia, Argentina, 2000a.

GONÇALVES, J.S. CAMOLESI, A.R. **O uso de programação reflexiva para o desenvolvimento de aplicações comerciais adaptativas.** Relatório Final de Projeto de Iniciação Científica, FEMA, Assis/SP, 2012.

GUERRA EDUARDO - **Componentes Reutilizável em Java com Reflexão e Anotaçãoes** - Casa do Código, BR, 2004

LIMA, J. C.S. Um **Estudo Sobre a Reconfiguração da Função de Compras em Empresas do Setor Automotivo**. Tese de Doutorado - Escola Politécnica da USP, 2004.

LUZ, J.C. **Tecnologia adaptativa aplicada à otimização de código em compiladores**. Dissertação de Mestrado, USP, São Paulo, 2004.

LUZ, J.C.; NETO, J.J. **Tecnologia adaptativa aplicada à otimização de código em compiladores.** IX Congreso Argentino de Ciencias de la Computación, La Plata,

MENEZES, Paulo Blauth, DIVERIO, Tiarajú Asmuz; Universidade Federal do Rio Grande do Sul (UFRGS). **Teoria da computação**. Porto Alegre: UFRGS, 1990.

NETO J.J. **Introdução a Compilação**. EDITORA: LTC, Rio de Janeiro, 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. **Cross-Assemblers para Microprocessadores** - Geração Automática Através do SPD. I CONAI - Congresso Nacional de Automação Industrial, pp. 501-509, São

Paulo, 1983.

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. **Uma Solução Adaptativa para Reconhecedores Sintáticos**. Anais EPUSP - Engenharia de Eletricidade - série B, vol. 1, pp. 645-657, São Paulo, 1988.

NETO, J.J.; ALMEIDA Jr.J.R.; NOVAES, J.M. **Synchronized Statecharts for Reactive Systems**. Proceedings of the IASTED International Conference on Applied Modelling and Simulation, pp.246-251, Honolulu, Hawai, 1998.

NETO, J.J.; IWAI, M.K. **Adaptive Automata for Syntax Learning**. CLEI 98 - XXIV Conferencia Latinoamericana de Informatica, MEMORIAS. pp. 135-149, Quito, Equador, 1998.

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.

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.

PISTORI, H.; NETO, J.J. AdapTree - **Proposta de um Algoritmo para Indução de Árvo-**

res de Decisão Baseado em Técnicas Adaptativas. Anais Conferência Latino Americana de Informática - CLEI 2002, Montevideo, Uruguai, Novembro, 2002.

PISTORI, H.; NETO, J.J.; COSTA, E.R. **Utilização de Tecnologia Adaptativa na Detecção da Direção do Olhar**. SPC Magazine, v.2., n.2, pp.22-29, Lima, Perú, Maio, 2003.

ROCHA, R.L.A. **Um método de escolha automática de soluções usando tecnologia adaptativa.** Tese de Doutorado, USP, São Paulo, 2000.

ROCHA, R.L.A. **Uma proposta de uso de tecnologia adaptativa para simulação de redes neurais em um dispositivo computacional**. In: IX Encuentro Chileno de Computación 2001, Punta Arenas. Proceedings of the Encuentro Chileno de Computación. Punta Arenas: Universidad de Magallanes, v. CD-ROM, pp. 1-9, 2001.

ROCHA, R.L.A.; NETO, J.J. **Construção e Simulação de Modelos Baseados em Autômatos Adaptativos em Linguagem Funcional**. Proceedings of ICIE 2001 - International Congress on Informatics Engineering, Buenos Aires: Computer Science Department - University of Buenos Aires, v. CD-ROM, pp. 509-521, 2001.

ROCHA, R.L.A.; NETO, J.J**. Uma proposta de método adaptativo para a seleção automática de soluções**. Proceedings of ICIE Y2K - International Congress on Informatics Engineering, Buenos Aires, 2000.

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.

SOUSA, M.A.A.; HIRAKAWA, A.R.; NETO, J.J. **Adaptive automata for mobile robotic mapping**. Proceedings of VIII Brazilian Symposium on Neural Networks - SBRN'04. São Luís/MA - Brazil. September 29 - October 1, 2004a.

SOUSA, M.A.A.; HIRAKAWA, A.R.; NETO, J.J. **Adaptive automata for mapping unknown environments by mobile robots**. Lecture Notes in Artificial Intelligence. C. Lemaître, C. A. Reyes, J. A. González (Eds.): Advances in Artificial Intelligence - IBERAMIA 2004, Springer-Verlag, pp 562-571, 2004b.

STAD. **State-Charts Adaptativos**. Disponível em http://lta.poli.usp.br/lta/, visitado em Março de 2015.

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

YAMAGUTI, M. H; **Uma Arquitetura Reflexiva Baseada na Web para Ambiente de Suporte a Processo. Dissertação de Mestrado**. Universidade Federal do Rio Grande do Sul 2002

YOURDON, E. **Análise Estruturada Moderna**. Editora Campus, 1990.

Documentação da Tabela Simples Nacional – Alíquotas e Partilhas do Simples Nacional – Comércio <http://www.normaslegais.com.br/legislacao/simples-nacional-anexoI.html> Acesso Maio 2015

Simples Nacional <http://idg.receita.fazenda.gov.br/@@busca?SearchableText=tabela+simples+nacional> Acesso Junho 2015

Planalto Simples Nacional <http://www.planalto.gov.br/ccivil_03/leis/LCP/Lcp116.htm> Acesso Setembro 2015

Planalto Simples Nacional <http://www.planalto.gov.br/ccivil_03/leis/LCP/Lcp123.htm#art21§4> Acesso Setembro 2015

Planalto Simples Nacional < http://www.planalto.gov.br/ccivil_03/leis/LCP/Lcp123.htm> Acesso Setembro 2015

Receita Fazenda – Simples Nacional <http://www8.receita.fazenda.gov.br/SimplesNacional/> Acesso Setembro 2015

Receita Fazenda – Simples Nacional <http://www8.receita.fazenda.gov.br/SimplesNacional/Documentos/Pagina.aspx?id=3> Acesso Setembro 2015