CENTRO UNIVERSITÁRIO UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE SISTEMAS DE INFORMAÇÃO

BRUNO DADALT ZAMBIAZI

ANÁLISE DE FERRAMENTAS PARA GESTÃO DE REGRAS DE NEGÓCIO EM SISTEMAS DE INFORMAÇÃO

Lajeado
2013

BRUNO DADALT ZAMBIAZI

ANÁLISE DE FERRAMENTAS PARA GESTÃO DE REGRAS DE NEGÓCIO EM SISTEMAS DE INFORMAÇÃO

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas do Centro Universitário UNIVATES, como parte dos requisitos para a obtenção do título de bacharel em Sistemas de Informação.
Área de concentração: Engenharia de Software

ORIENTADOR: Fabrício Pretto

Lajeado

2013

BRUNO DADALT ZAMBIAZI

ANÁLISE DE FERRAMENTAS PARA GESTÃO DE REGRAS DE NEGÓCIO EM SISTEMAS DE INFORMAÇÃO

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Sistemas de Informação e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Fabrício Pretto, UNIVATES

Mestre pela PUCRS – Porto Alegre, Brasil

Banca Examinadora:

Prof. Fabrício Pretto, UNIVATES

Mestre pela PUCRS – Porto Alegre, Brasil

Prof. Pablo Dall'Oglio, UNIVATES

Mestre pela UNISINOS – São Leopoldo, Brasil

Prof. Vilson Cristiano Gärtner, UNIVATES

Mestre pela UNISINOS – São Leopoldo, Brasil

Coordenador do Curso de Sistemas de Informação :_____

Prof. Evandro Franzen

Lajeado, dezembro de 2013.

RESUMO

As regras de negócio se tornaram um importante aspecto no âmbito do desenvolvimento de sistemas de informação, visto que representam o conhecimento que guia as decisões e operações de uma organização. Atualmente, a maior parte dos sistemas de informação existentes possui as regras de negócio embutidas diretamente em seu código-fonte, representadas como programas, o que as torna visíveis apenas por pessoas com conhecimento técnico em linguagens de programação. Ao passo que as regras de negócio costumam mudar com muita frequência, seja por leis e regulamentações externas, ou, então, por necessidades e políticas internas, torna-se essencial que os sistemas de informação que as implementam deixem-nas acessíveis e passíveis de modificação para um novo público: os responsáveis pelo negócio. Este trabalho visa à análise e comparação de duas ferramentas que possibilitam a gestão de regras de negócio a partir de sua externalização do código-fonte de sistemas de informação.

Palavras-chave: brms, regras de negócio, sistemas de gerenciamento de regras de negócio, sistemas de informação.

ABSTRACT

Business rules have become an important aspect in information systems development scope since they represent the knowledge that drives business decisions and operations. Nowadays, the major part of existing information systems have business rules embedded into their source code, represented as code programs, which makes them visible just for people with technical knowledge in programming languages. While business rules often change, either by external laws and regulations or by internal requirements and policies, information systems that implement them need to turn them accessible and subject to change for a new audience: the business people. This work aims at analyzing and comparing two tools that enable the business rules management through their source code externalization in information systems.

Keywords: brms, business rules, business rules management systems, information systems.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API –	Application Programming Interface
BPM –	Business Process Management
BPMN –	Business Process Model and Notation
BRMS –	Business Rules Management System
DRL –	Drools Rule Language
DSL –	Domain Specific Language
ECA –	Evento-Condição-Ação
IA –	Inteligência Artificial
JCR –	Java Content Repository
LHS –	Left Hand Side
RHS –	Right Hand Side
RMI –	Remote Method Invocation
RPC –	Remote Procedure Call
SAD –	Sistema de Apoio à Decisão
SBC –	Sistema Baseado em Conhecimento
SGBD –	Sistema de Gerenciamento de Bancos de Dados
SI –	Sistema de Informação
SOA –	Service Oriented Architecture
SOAP –	Simple Object Access Protocol
TI –	Tecnologia da Informação
XML –	eXtensible Markup Language
WS –	Web Service
WSDL –	Web Services Description Language

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de informação representam, atualmente, elementos fundamentais na manutenção de empresas e organizações de sucesso. Para Rezende (2005), "sistemas de informação são todos os sistemas que produzem ou geram informações, que são dados trabalhados […] para execução de ações e para auxiliar processos de tomada de decisões". A utilização destes sistemas no apoio à execução de aspectos de negócio das organizações traz consigo uma nova perspectiva: a necessidade de atualização constante dos sistemas para possibilitar a adaptação das organizações ao mercado dinâmico e competitivo da atualidade.

O desenvolvimento de sistemas de informação para auxiliar processos de negócio em áreas como contabilidade, financeira, logística, *marketing*, entre outras, faz com que as alterações exigidas, na maior parte dos casos, variem de leis e regulamentações externas a necessidades de mercado e políticas internas. Tudo isso é conhecido como regras de negócio, conforme ressaltado por Ross (2000): "as regras de negócio são o que guiam sua organização nas operações do dia a dia".

A maioria dos sistemas de informação desenvolvidos na atualidade é construída conforme um modelo conhecido como Arquitetura em Três Camadas, na qual se dividem as funcionalidades conforme o objetivo de cada camada, geralmente havendo uma para apresentação, outra para a lógica de negócio e uma última para os dados. Em sistemas com um bom nível de separação, as regras de negócio são frequentemente desenvolvidas de forma centralizada na camada de lógica de negócio, embora seja comum encontrá-las espalhadas – muitas vezes duplicadas – em outras camadas.

Além disso, muitas organizações se deparam com a necessidade de manter diversos sistemas de informação para diferentes fins, tanto sistemas mais novos como sistemas legados. Isso significa que muitas regras de negócio acabam sendo replicadas entre diferentes aplicações, o que, no momento de uma atualização, aumenta os custos de manutenção e exige um grande esforço das equipes de desenvolvimento, visto que pode ser necessária a alteração da regra em diferentes pontos de um mesmo sistema ou, ainda pior, em diferentes sistemas.

Outro importante aspecto referente às regras de negócio diz respeito a quem detém seu conhecimento. Os envolvidos com o desenvolvimento dos sistemas ou com a tecnologia da informação, em geral, não costumam saber como e quando uma regra de negócio mudará, nem quais impactos sua implementação causará. Segundo Bajec e Krisper (2005a), "regras de

negócio não pertencem aos sistemas de informação; regras de negócio pertencem ao negócio e devem ser mantidas e gerenciadas pelo negócio".

Para que a manutenção das regras de negócio possa ser feita pelas pessoas vinculadas à área de negócio, é fundamental que elas sejam retiradas do código-fonte da aplicação e façam parte de algum repositório central que permita seu gerenciamento completo. Foi com este objetivo que surgiram os sistemas de gerenciamento de regras de negócio, ou Business Rules Management Systems (BRMS), que visam possibilitar que usuários detentores do conhecimento sejam os responsáveis pelas regras que guiam os sistemas, retirando essa atribuição da área de tecnologia da informação e proporcionando um ambiente eficiente para o controle das regras de uma organização e de seus sistemas de informação.

1.1 Justificativa

De acordo com Graham (2007, p. 25), estima-se que aproximadamente 90% dos gastos com a tecnologia da informação seja atribuído à manutenção de sistemas existentes, ao invés do seu desenvolvimento. Isso pode ser explicado, ao menos em partes, devido à replicação de regras de negócio que ocorre entre diferentes sistemas de uma mesma organização ou em diferentes pontos de um mesmo sistema.

Na inexistência de um repositório que centralize as regras de negócio, cada local de implementação se torna um possível ponto de falha, pois, quando se torna necessária uma alteração, algum ponto pode ser esquecido ou, simplesmente, alterado de forma errada. Todavia, a simples centralização de regras de negócio não resolve todos os problemas. Isso porque, em tese, o conhecimento de uma área de negócio deveria ficar a cargo das pessoas envolvidas com o negócio, tornando estas as responsáveis pelas suas regras, retirando tal atribuição dos desenvolvedores e analistas de sistemas.

Dessa forma, grande parte das atualizações nos sistemas de informação não precisaria mais ser feita apenas por pessoas com conhecimentos técnicos, uma vez que a manutenção das regras que guiam o funcionamento do negócio – e, consequentemente, dos sistemas que as implementam – poderia ser feita quase que em sua totalidade pelas pessoas que detêm o conhecimento: as pessoas do negócio.

> Não é mais desejável embutir as regras em especificações ou códigos de programas onde elas fiquem trancadas, necessitando de intervenções custosas e elevadas para efetuar mudanças. Você não pode mais entregar as regras em um formato inacessível e não entendível à audiência do negócio. Você não pode mais deixar as regras de

negócio em um local onde elas se percam (HALLE, 2002).

1.2 Objetivos

O objetivo deste trabalho é a análise e comparação de duas ferramentas que se propõem ao gerenciamento de regras de negócio. Estas tarefas ocorrem a partir do exemplo de um hipotético sistema de comércio eletrônico, o qual teve um conjunto básico de regras de negócio especificado, sendo estas regras posteriormente implementadas em cada uma das ferramentas, possibilitando, assim, obter dados e ciência suficiente para compará-las em termos de abrangência, funcionalidades e limitações.

Como conclusão, será indicado se as ferramentas estudadas fornecem meios satisfatórios para a criação e manutenção das regras de negócio de um sistema de informação, por usuários sem conhecimentos técnicos em linguagens de programação, assim como indicam as principais referências bibliográficas da área.

Este trabalho não visa ao estudo e nem à utilização de Business Process Management (BPM) e Business Process Model and Notation (BPMN), que, embora relacionadas com regras e modelos de negócio, são áreas voltadas à parte de processos.

1.3 Metodologia

Para o desenvolvimento deste trabalho, foi utilizada a metodologia de pesquisa de natureza exploratória, tendo como procedimentos a pesquisa bibliográfica e o estudo de caso.

A pesquisa exploratória possui o objetivo de "proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito […] de modo que possibilite a consideração dos mais variados aspectos relativos ao fato estudado" (GIL, 2002). A pesquisa bibliográfica é elaborada a partir de materiais já publicados, visando embasar o conteúdo do trabalho, e "em suma, […] leva ao aprendizado sobre uma determinada área" (CRUZ; RIBEIRO, 2003). O estudo de caso, por fim, permite o estudo detalhado e amplo conhecimento sobre determinado objeto de estudo (GIL, 2002).

Para que os objetivos deste trabalho fossem alcançados, foi realizada a análise e comparação entre ferramentas que possibilitam o gerenciamento de regras de negócio. A análise levou em conta um estudo de caso referente a um conjunto básico de regras de negócio envolvido com um hipotético sistema de comércio eletrônico.

1.4 Organização

O presente trabalho está organizado da seguinte forma: o segundo capítulo apresenta embasamento e referencial teórico sobre os assuntos relacionados ao trabalho; o terceiro capítulo explica o funcionamento básico dos softwares utilizados no estudo; o quarto capítulo demonstra o estudo de caso realizado através da comparação das ferramentas escolhidas; o quinto e último capítulo apresenta a conclusão do trabalho.

2 REFERENCIAL TEÓRICO

Neste capítulo, são apresentados os conceitos relevantes para a proposta deste trabalho, passando por uma visão geral das regras de negócio, seu ciclo de vida, formas de identificação e especificação, bem como a definição do que são sistemas de gerenciamento de regras.

2.1 Regras de negócio

Nas últimas décadas, o termo regras de negócio se popularizou no ambiente de Tecnologia da Informação (TI) devido ao fato de que utilizar uma abordagem voltada a sua implementação, no desenvolvimento de sistemas, capacita as aplicações a se tornarem mais flexíveis e propensas a modificações (BAJEC; KRISPER, 2005a). Porém, apesar da grande atenção recebida nos últimos anos, tanto de profissionais quanto de pesquisadores, ainda não existe uma definição única, clara e objetiva sobre o que as regras de negócio são e o que exatamente representam (BAJEC; KRISPER, 2005b). Morgan (2002) complementa citando que regras de negócio é um termo que muitas pessoas utilizam, embora poucas consigam definir.

Ao longo dos anos, muitos autores propuseram diversas definições para o termo. Dentre as quais, pode-se destacar:

- “Uma regra de negócio é uma declaração explícita estipulando uma condição que deve existir no ambiente de informação de negócios, para a informação extraída do ambiente ser consistente com a política da empresa.” (APPLETON, 1984);
- “Uma regra de negócio é uma regra afirmando algo que impacta nos interesses do negócio, e a interpretação da regra pode ter grande impacto na qualidade do sistema de informação que será desenvolvido.” (SELVEITH, 1991);
- “Uma regra de negócio é uma declaração que define ou restringe algum aspecto do negócio. Pretende afirmar a estrutura, ou controlar, ou influenciar o comportamento do negócio.” (HAY; HEALY, 2000);
- “Regras de negócio [...] servem como um sistema de orientação que influencia o comportamento coletivo das pessoas de uma organização e os sistemas de informação.” (HALLE, 2002).

De acordo com Ross (2000), todos os envolvidos em uma organização sabem, ao menos implicitamente, o que são as regras de negócio. Elas representam o que se utiliza para conduzir a empresa, guiando seus colaboradores nas operações e decisões do dia a dia. Se elas não existissem, as decisões seriam tomadas com base na intuição, conforme análise das situações caso a caso, o que, em algum momento, produziria resultados inconsistentes e insatisfatórios. Segundo o autor, as empresas não conseguem mais sobreviver agindo dessa forma, e, portanto, as que têm algum processo empresarial organizado possuem – e conhecem – regras de negócio.

No âmbito empresarial, as decisões tomadas pelos gestores para definir o rumo das empresas se baseiam em fatos e restrições. Para que suas decisões sejam de qualidade e precisão, é necessário que as regras que auxiliam na tomada das decisões também sejam de qualidade. Halle (2002) cita estas regras como a definição ideal para o termo regras de negócio, afirmando, ainda, que elas representam a expressão formal do conhecimento que leva a empresa para a direção desejada.

Outro importante aspecto relacionado ao tema diz respeito à modelagem de negócios de uma organização. Kovacic (2004) explica que um modelo de negócios é uma visão simplificada dos processos de negócio da empresa, que mostra como seus componentes são relacionados entre si para operarem em conjunto e que enfatiza os aspectos mais importantes de determinado contexto, deixando de lado detalhes irrelevantes. O objetivo, segundo o autor, é prover uma imagem clara do estado atual da empresa e determinar sua visão para o futuro, servindo como base para a extração de regras de negócio que podem, posteriormente, serem implementadas em um sistema de informação. Eriksson e Penker apud Kovacic (2004) citam algumas vantagens para justificar a produção de modelos de negócio:

- Possibilitam a melhoria no entendimento do negócio como um todo, facilitando a comunicação entre os envolvidos com os processos empresariais;
- Mostram uma imagem clara da situação atual do negócio, servindo como a base das melhorias na estrutura do negócio e suas operações;
- Identificam as partes mais importantes do negócio, permitindo que as de menor importância possam ser delegadas a terceiros;
- Podem servir como base para a criação de um sistema de informação.

Para complementar o conceito de regras de negócio, Ross (2003) indica que as regras

representam a base do conhecimento do negócio, são motivadas pelas metas e objetivos empresariais e, justamente por isso tudo, devem ser definidas por pessoas que estejam diretamente envolvidas com a área de negócios da organização. Para finalizar, o autor cita que "se alguma coisa não pode ser expressa, então ela não é uma regra".

2.1.1 Taxonomia

A classificação das regras de negócio é algo que ainda não possui uma padronização e definição clara. Fica a cargo de cada autor, portanto, agrupá-las da forma como considera mais adequado.

Shao e Pound (1999), por exemplo, propõem separar as regras de negócio em quatro tipos:

a) Regras estruturais: especificam detalhes sobre definições utilizadas no contexto de negócio da empresa, como, por exemplo, o que significa um "cliente ouro". Exemplo:
   - *É considerado "cliente ouro" todo cliente que realizar mais de 20 compras no período de 6 meses.*

b) Regras derivativas: são as regras derivadas de outros fatos, como cálculos matemáticos, inferência ou dedução. Exemplo:
   - *O valor total da venda deve ser calculado conforme somatório do valor dos itens e subtração do desconto fornecido pelo gerente.*

c) Regras restritivas: especificam as condições que devem ser atendidas para que determinada operação possa ser feita. Exemplo:
   - *Os descontos acima de 10% do valor total da venda devem ser autorizados pelo setor comercial.*

d) Regras eventuais: possuem indicativos de tempo ou quantidade de ações para a ocorrência de algum fato. Exemplo:
   - *Três é o limite de tentativas de acesso a uma conta até que ela seja automaticamente bloqueada.*

Morgan (2002), por outro lado, indica a existência de três categorias principais de regras de negócio. São elas:

a) Regras de estruturas: descrevem as restrições e relacionamentos entre vários

elementos de um modelo ou sistema. Exemplo:

- *Um pedido preenchido deve ser associado a um empacotador específico.*

b) Regras de comportamento: definem ações a serem seguidas conforme uma situação em particular, além do reconhecimento e resposta a determinados eventos de negócio. No geral, são executadas a partir da mudança de estado de algum dos elementos envolvidos. Exemplo:

   - *Um pedido classificado como urgente deve ser encaminhado ao supervisor de empacotamento para expedição imediata.*

c) Regras de definição: proveem a definição sobre um termo ou o relacionamento quantitativo/qualitativo entre alguns termos. Exemplo de regra:

   - *O valor total de um pedido é a soma entre o valor dos itens mais o imposto sobre a venda.*

Entretanto, a taxonomia mais utilizada diz respeito a uma subcategorização específica sobre o que, de fato, compõe uma regra de negócio. Hay e Healy (2000), Ross (2000, 2003), Santos (2010), entre outros, declaram que uma regra de negócio representa a composição de quatro elementos/divisões: termos, fatos, restrições e derivações.

Geralmente expressos como glossários, os termos representam substantivos, expressões ou sentenças que definem o que são determinados elementos relevantes ao contexto do negócio (SANTOS, 2010). Exemplo: *define-se como cliente ouro qualquer cliente que fizer mais de 20 compras em um período de 6 meses*.

Os fatos são proposições que ligam dois ou mais termos, visando à formação de uma sentença que exprima a importância de algo para o negócio (HAY; HEALY, 2000). Como exemplo, pode-se considerar o *desconto especial* que poderia ser dado a cada *cliente ouro* – sendo que, neste caso, "desconto especial" e "cliente ouro" são dois termos previamente definidos.

Restrições representam condições que devem ser atendidas para que determinada situação ou evento possa ocorrer (SANTOS, 2010). Por exemplo: a *compra através de cartão de crédito deve ser aprovada pela operadora responsável.*

As derivações, por sua vez, são a obtenção de novas regras a partir de outras regras já existentes (HAY; HEALY, 2000) ou a utilização do conhecimento de uma regra através de outra (SANTOS, 2010). Exemplo: *o valor total da venda é representado pelo somatório do*

valor dos itens com o cálculo do imposto.

Tabela 1 - Taxonomia mais utilizada para classificação de regras de negócio

Tipo	Significado
Termos	Expressões que definem o que são/representam os elementos utilizados nas regras de negócio.
Fatos	Composição de termos para formar declarações que demonstram a importância de algo para o negócio.
Restrições	Pré-condições que precisam ser atendidas para que determinada ação seja realizada.
Derivações	Obtenção de novas regras a partir de outras já existentes.

Fonte: elaborado pelo autor.

2.1.2 Análise de negócio

As etapas de levantamento e especificação das regras de negócio se tornaram uma etapa crucial para a entrega de softwares que atendam às necessidades do cliente (HALLE, 2002). Do ponto de vista empresarial, muitas restrições que devem estar presentes nos sistemas de informação são coisas óbvias e fáceis de serem implementadas. Isso implica na existência de regras de negócio que não são explicitamente formuladas como requisitos, por serem consideradas triviais (HERBST et al., 1994). Com isso, a análise de negócio se manifesta como a área que deve ser combinada à análise de sistemas para permitir que este objetivo – a entrega do software de qualidade – seja alcançado, envolvendo a capacidade de percepção de problemas tanto no âmbito estratégico quanto no técnico (CARKENORD, 2008).

Carkenord (2008) caracteriza a análise de negócio como:

> Conjunto de tarefas e técnicas usadas para trabalhar como uma ligação entre as partes interessadas, com o objetivo de entender a estrutura, políticas e operações de uma organização, e recomendar soluções que habilitem a empresa a atingir seus objetivos.

A análise de negócio envolve aspectos e tarefas como: identificar os problemas e oportunidades relacionados ao negócio; perceber e analisar as necessidades e restrições advindas das partes interessadas em determinado processo ou problema – visando sua definição como requisitos de uma solução; validar soluções; gerenciar o escopo dos requisitos (CARKENORD, 2008).

A importância da análise de negócio fica evidente, segundo Hay e Healy (2000), pois

os analistas de sistemas tendem a não perceber muitas das regras de negócio envolvidas nos processos que guiam uma empresa. Segundo os autores, embora tenham capacidade de descrever a organização em termos de estruturas de dados, os analistas de sistemas acabam negligenciando algumas das restrições sobre as quais a empresa opera, muitas vezes devido à natureza de seu conhecimento e de sua área de especialização mais técnica.

Atualmente, as mudanças contínuas são um fator central para o negócio e as técnicas utilizadas para realizar o levantamento e análise do negócio devem ser baseadas no fato de que as regras mudarão e, muito provavelmente, com bastante frequência. Assim, a melhor solução de negócio é a que atende às mudanças fazendo isso de forma amigável às pessoas e analistas envolvidos diretamente com o negócio (ROSS; LAM, 2011).

2.1.3 Identificação e levantamento

O processo de levantamento das regras de negócio pode ser baseado em dois tipos de conhecimento: explícito e implícito. Enquanto o primeiro é um tipo de conhecimento fácil de expressar na forma de princípios, procedimentos, fatos e diagramas, o segundo, geralmente, baseia-se em operações sem representação formal. Ou seja, o conhecimento implícito acaba sendo baseado na compreensão, experiência e intuição de pessoas envolvidas em determinada tarefa, sendo algo altamente pessoal e subjetivo, normalmente obtido através de entrevistas e conversas informais. A extração deste tipo de regra, e sua consequente transformação em declarações claras e entendíveis, é um problema muito comum e geralmente complicado de resolver (BAJEC; KRISPER, 2005b).

Por outro lado, operações baseadas em procedimentos já padronizados e bem propagados dentro da organização, tornam-se regras de negócio possíveis de serem documentadas por se basearem no chamado “conhecimento explícito”. Como muitas vezes são obtidos através de modelos já existentes e bem documentados, tais regras são mais facilmente automatizadas por sistemas de informação pois sua definição costuma ser mais clara e fácil de compreender (BAJEC; KRISPER, 2005b).

Hay e Healy (2000) mencionam que a identificação das regras de negócio costuma começar pelo conhecimento da política da organização. Porém, mesmo esta sendo formal e exclusiva, é tipicamente descrita de forma superficial, sendo uma representação, aos olhos dos colaboradores, de tarefas específicas que eles devem realizar. Além disso, muitas regras são

capturadas pelas “divagações” dos funcionários, sendo originadas a partir de operações realizadas em seu dia a dia – o conhecimento implícito. Assim, em alguns casos as regras identificadas são claras, mas, muitas vezes, acabam sendo ambíguas e contendo mais de uma ideia expressa em uma única declaração.

Ainda segundo a fonte, a partir da coleta de todas as regras possíveis, é tarefa do analista responsável decompô-las e identificar sua estabilidade, ou seja, se são regras fundamentais à operação do negócio da empresa, ou se são vagas demais para serem consideradas regras. Conforme Ross (2003), qualquer declaração que não se consegue expressar não pode ser considerada uma regra de negócio.

Hay e Healy (2000) também afirmam que, após as etapas citadas, a próxima tarefa é identificar todas as declarações com a taxonomia correta – classificando-as em termos, fatos, restrições ou derivações. Enquanto termos, fatos e alguns tipos de restrições podem ser representados diretamente em modelos gráficos, as derivações e restrições restantes devem ser expressas com algum tipo maior de formulismo, como documentações, declaração em linguagem natural, entre outras.

2.1.4 Definição e especificação

Uma regra de negócio em particular pode estar envolvida dentro de diversos processos de negócio, sendo utilizada – ou replicada – em diversos subsistemas. Há diversas maneiras de desenvolvê-las, desde implementações em nível de código-fonte a *triggers* de bancos de dados. Porém, para que seja possível identificar as implicações de alterá-las no futuro, bem como entender seu significado nos sistemas que as implementam, é vital que elas sejam claramente documentadas durante todo o seu ciclo de vida, da criação às mudanças posteriores (BAJEC; KRISPER, 2005a).

Ter as regras de negócio bem documentadas e organizadas faz uma grande diferença, podendo proporcionar vantagem competitiva frente aos concorrentes, visto que elas são, na realidade, o que realmente direciona o funcionamento da organização. Regras bem documentadas e escritas claramente também ajudam no processo de obtenção de resultados. Isso porque sua expressão é muito mais próxima a formas comuns de expressão de usuários que não trabalham diretamente com TI, ou seja, deixa as regras que governam a aplicação mais acessíveis a seus usuários e operadores (GOUGEON, 2003).

Para que isso tudo seja possível, a definição e a especificação das regras de negócio devem ser feitas com muito cuidado. Segundo Gottesdiener (1997), uma regra deve ser escrita de forma que represente uma única e completa ideia, tornando-se um componente atômico que não pode ser dividido. Para Morgan (2002), a clareza da regra é fundamental para seu entendimento, devendo ser escrita em um formato que o responsável pelo negócio consiga imediatamente entendê-la e aceitá-la – como lógica – ou rejeitá-la – como inválida.

Segundo Date (2000), é importante entender que a escrita das regras de negócio deve ser independente de software ou hardware. Conforme Ross (2000), as regras de negócio não são um sistema, embora, geralmente, sejam implementadas dentro – ou através – de um. Para Gottesdiener (1997), as regras precisam ser independentes de paradigmas ou plataformas técnicas, pois devem ser definidas, gerenciadas e orientadas para e pelo negócio.

O conjunto de regras de toda a aplicação, pertinente ao negócio em questão, deve constituir um modelo de negócio completo. De fato, em virtude das regras serem compartilhadas entre diferentes aplicações, a atividade de definição das regras, no geral, deve ser vista não apenas como um processo de desenvolvimento de aplicações individuais, mas, sim, como o desenvolvimento de uma aplicação inteiramente integrada (DATE, 2000).

Referente à especificação das regras, deve-se ter cuidado com suas sentenças. Conforme Ross (2009), uma sentença de regra de negócio é um modelo de escrita, geralmente em linguagem natural, que serve para expressar determinado tipo de regra. Segundo o autor, a utilização de padrões específicos para sua definição possibilita que as regras sejam mais facilmente entendidas por diferentes pessoas envolvidas no processo, pois permite que expressem suas ideias de forma semelhante.

Para Morgan (2002), a forma ideal para expressar as regras de negócio é mediante variações do modelo mais simples de sentença: <*sujeito*> *deve* <*restrição*>. Exemplo: *"Para que a venda seja efetuada, o cliente deve ser considerado adimplente"*. Segundo o autor, essa forma de representação induz as regras ao conceito que considera fundamental: sempre retornar uma expressão verdadeira.

Atualmente, entretanto, existem algumas formas padronizadas para a escrita de regras de negócio. Um das mais utilizadas, Rule Speak® Sentence Forms, criada e citada por Ross (2009), define que é importante que as regras não sejam escritas de acordo com alguma linguagem de programação – algo muito comum quando se envolve no processo alguém da área de TI.

2.2 Regras de negócio na engenharia de software

A engenharia de software é uma matéria da engenharia que se refere a todos os aspectos de produção e manutenção de um software, visando ao desenvolvimento de aplicações de grande qualidade e de forma produtiva.

> A engenharia de software é um rebento da engenharia de sistemas e de hardware. Ela abrange um conjunto de três elementos fundamentais – métodos, ferramentas e procedimentos – que possibilita ao gerente o controle do processo de desenvolvimento do software e oferece ao profissional uma base para a construção de software de alta qualidade produtivamente (PRESSMAN, 1995, p. 31).

Segundo Sommerville (2007), uma das maiores dificuldades encontradas no desenvolvimento de grandes sistemas é o que está envolvido com a engenharia de requisitos. Para o autor, os requisitos de um sistema indicam o que ele deve fazer e de como deve se comportar, incluindo a descrição de suas funcionalidades e serviços oferecidos, bem como regras e restrições operacionais. "O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado de engenharia de requisitos" (SOMMERVILLE, 2007, p. 79). Análise e especificação de requisitos são tarefas da engenharia de software que ocorrem antes da etapa de desenvolvimento, proporcionando a representação do que o software deve realizar e os critérios para avaliação da qualidade do que será construído (PRESSMAN, 1995).

O desenvolvimento de um software pode ser feito de acordo com diversas metodologias diferentes, como o modelo em cascata[1], o desenvolvimento evolucionário[2] e o baseado em componentes[3]. Para Sommervile (2007), embora exista diversos processos de software, há atividades comuns e fundamentais a todos eles (TABELA 2).

1 No modelo em cascata, todas as etapas (definição, implementação, testes, etc) são feitas de forma sequencial, ou seja, uma só começa quando a anterior for considerada concluída (SOMMERVILLE, 2007).

2 O desenvolvimento evolucionário prevê a criação inicial de uma software simples, posteriormente evoluindo-o até que se torne o produto final adequado. É uma abordagem que intercala as atividades de especificação, desenvolvimento e validação (SOMMERVILE, 2007).

3 Abordagem na qual o processo de desenvolvimento de software foca na integração de componentes previamente existentes e reutilizáveis, sem ter de desenvolvê-los do zero (SOMMERVILLE, 2007).

Tabela 2 - Atividades fundamentais no desenvolvimento de softwares

Atividade	Descrição
Especificação de software	A funcionalidade do software e as restrições sobre sua operação devem ser definidas.
Projeto e implementação de software	O software que atenda à especificação deve ser produzido.
Validação de software	O software deve ser validado para garantir que ele faça o que o cliente deseja.
Evolução de software	O software deve evoluir para atender às necessidades mutáveis do cliente.

Fonte: SOMMERVILLE, 2007, p. 43.

A atividade de especificação do software, descrita no primeiro item da Tabela 2, que inclui o processo de engenharia de requisitos, é considerada, segundo Sommerville (2007), um estágio crítico no desenvolvimento do software, uma vez que os erros cometidos nessa etapa conduzem a inevitáveis problemas no momento da implementação. Em virtude disso, levar em conta as regras de negócio da organização é fundamental para que se obtenha êxito no processo de levantamento e especificação de requisitos – e, posteriormente, no desenvolvimento do software (GUEDES, 2009).

2.3 Regras de negócio em Sistemas de Informação

Nos últimos anos, houve grande crescimento na adoção de abordagens voltadas à utilização de regras de negócio no contexto de desenvolvimento de Sistemas de Informação (SI), tanto em termos de importância quanto de popularidade. Bajec apud Kovacic (2004) considera tal fato um reconhecimento das regras de negócio como peça fundamental no desenvolvimento de aplicações flexíveis e aptas a mudanças constantes.

De acordo com Bajec e Krisper (2005a), a origem das regras de negócio vem dos Sistemas Baseados em Conhecimento (SBC), aplicações da área da Inteligência Artificial (IA) nas quais o conhecimento e o raciocínio humanos são expressos na forma de complexas redes de regras. Geralmente, tais regras são escritas de uma forma que não exigem ordem sequencial e fluxo de controle, salvas em uma base de regras e posteriormente processadas por um componente chamado motor de inferências. O motor é responsável por avaliar as condições das regras e decidir quais – e quando – devem ser disparadas.

Em SIs, Gottesdiener (1997) cita diversas vantagens na adoção de abordagens voltadas

à implementação de regras de negócio: independência técnica, agilidade no desenvolvimento de sistemas, requisitos de melhor qualidade, um balanço claro entre flexibilidade e controle centralizado e facilidade de mudanças nas regras. Esta última característica, aliás, é citada por diversos autores (HALLE, 2002; KOVACIC, 2004; BAJEC e KRISPER, 2005a) como uma vantagem crucial na utilização desse tipo de abordagem.

Willmor e Embury (2006), por exemplo, citam que as mudanças nas regras de negócio costumam acontecer conforme novas oportunidades de mercado, como o desejo de aumentar o número de clientes ou maximizar as vendas, ou por regulamentos que obrigam a conformidade com determinadas leis. Já Kovacic (2004) explica que mudanças rápidas e constantes, muito comuns no mundo dos negócios, afetam não somente a área de negócios, mas também os SIs utilizados. Dessa forma, segundo o autor, exige-se que estes sistemas sejam preparados para se adaptarem frequentemente às novas regras do negócio.

Uma abordagem voltada à utilização de regras de negócio em SIs é importante, também, para proporcionar uma maior aproximação entre as áreas do negócio e da TI. Segundo Bajec e Krisper (2005b), o alinhamento destas áreas pode ser considerado um dos maiores desafios em qualquer empresa, porém, conforme Gottesdiener (1997), as regras de negócio colocam as pessoas do negócio no centro da atividade que envolve o software, criando, assim, uma integração real entre elas.

2.3.1 Perspectiva do negócio x perspectiva dos Sistemas de Informação

Conforme já explicado, o conceito de regras de negócio é muito amplo. Assim, de acordo com Gottesdiener (1997), as regras têm diferentes conotações dependendo do contexto em que são utilizadas.

Segundo Hay e Healy (2000), por exemplo, é necessário separar o conceito de regras em duas perspectivas: do negócio e da TI. No âmbito do negócio, as regras se referem a quaisquer restrições aplicadas ao comportamento das pessoas na empresa, da impossibilidade de fumar a procedimentos de preenchimento de uma ordem de compra; sob a visão dos SIs, referem-se aos fatos que são guardados como dados e às restrições específicas para criação, manutenção e remoção destes dados. Bajec e Krisper (2005b) complementam:

> Para as pessoas do negócio, as regras de negócio são diretivas que pretendem influenciar ou guiar o comportamento do negócio. Desenvolvedores, por outro lado, tendem a enxergar as regras de negócio como partes atômicas da lógica de programas que estão tipicamente no formato 'SE-ENTÃO'.

Pela perspectiva do negócio, regras podem ser definidas como proposições que restringem o comportamento de uma organização, existindo em diversas formas que variam de simples a complexas e dinâmicas (BAJEC; KRISPER, 2005a). Segundo a mesma fonte, as regras podem ser definidas com base interna ou externa, dependendo do contexto da organização: enquanto as internas são propostas dentro da empresa, geralmente advindas de elementos estratégicos como missão, visão de mercado e fatores de sucesso, as regras externas, por outro lado, provêm do ambiente externo à empresa, geralmente surgidas devido a regulamentações específicas do ramo de atuação daquela organização.

Quanto à utilização de regras de negócio nos SIs, Halle (2002) elenca diversas vantagens para ambas as áreas. Para a de negócios, destacam-se:

- O processo de modificação das regras não é mais tão custoso e demorado;
- As pessoas do negócio ficam mais próximas das especificações do sistema;
- As regras se tornam um mentor para qualquer usuário do sistema quando estão documentadas e acessíveis via repositório, e não mais escondidas no código-fonte. Assim, as pessoas sabem onde encontrá-las e entendem como o processo funciona;
- Conclusões encontradas por processos de *data warehousing*[4] e/ou *data mining*[5] fazem mais sentido quando associadas a regras de negócio ativas.

Pela perspectiva dos SIs, todas as regras que tendem a apresentar mudanças constantes requerem atenção especial. Assim, acabam sendo consideradas como regras de negócio, pelos desenvolvedores e até mesmo analistas, não somente as regras que derivam especificamente do negócio, mas, também, todas as outras que surgem dentro do processo de desenvolvimento de uma aplicação (BAJEC; KRISPER, 2005b).

Os mesmos autores também citam que um dos equívocos mais comuns em relação ao tema, este tipicamente cometido por desenvolvedores, é achar que todas as regras e restrições que governam uma aplicação representam, de fato, regras de negócio. Regras que se referem exclusivamente à parte visual da aplicação, como, por exemplo, a indicação da cor com a qual deve ser exibido determinado valor em um balancete financeiro, não representam regras de negócio, pois, para os gestores e pessoas envolvidas com os processos de negócio, o que importa é o valor da informação – e não sua apresentação.

4 *Data warehousing* é uma tecnologia que armazena uma grande quantidade de dados, visando à apresentação deles em informações estratégicas úteis para o processo de tomada de decisões (PONNIAH, 2001).

5 *Data mining* é um processo de análise de uma grande quantidade de dados "dispersos", visando à coleta/descoberta de novas informações (PONNIAH, 2001).

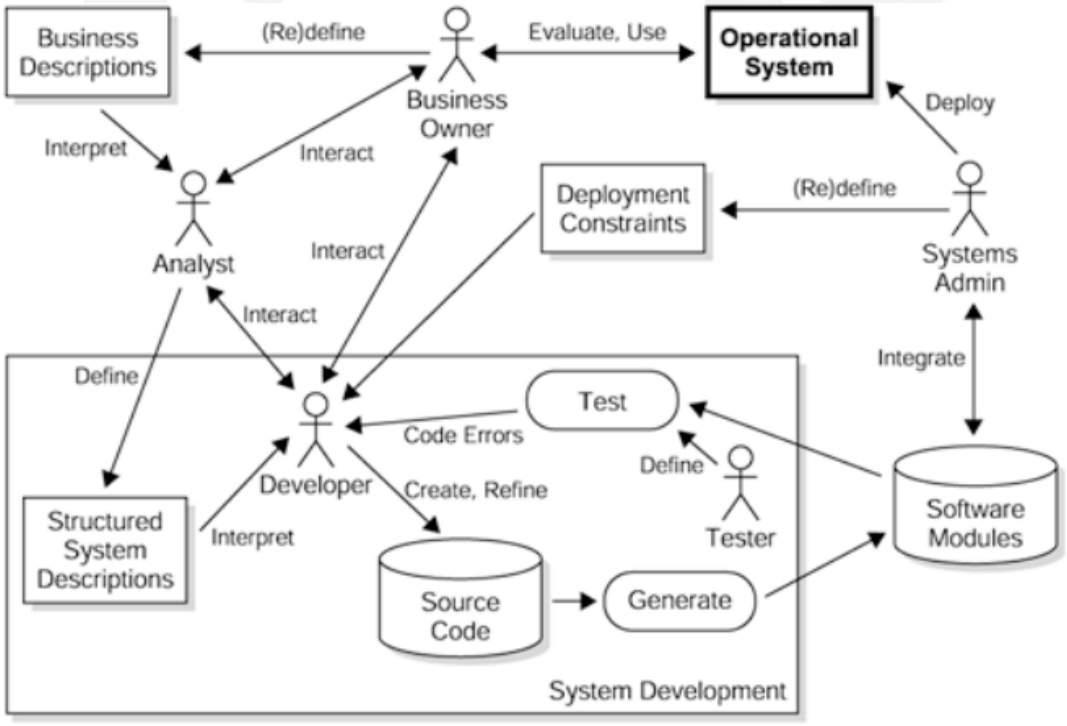
Halle (2002) também cita vantagens para os SIs na utilização de uma abordagem voltada às regras de negócio:

- Diminuição do tempo de desenvolvimento de uma aplicação pois, geralmente, há menos coisas a serem feitas;
- Os sistemas estão preparados para mudanças;
- Utilizando-se ferramentas para a gestão das regras de negócio, é diminuída a distância entre os processos de requisitos, análise e implementação;
- O desenvolvimento de um sistema baseado em regras tende a possuir um custo mais efetivo do que trabalhar sobre customizações.

2.3.2 Modelos de desenvolvimento de Sistemas de Informação

Para entender a importância das regras de negócio no contexto dos SIs, é necessário, primeiramente, entender como funciona o processo de desenvolvimento utilizado atualmente na maior parte das empresas. A Figura 1 apresenta um modelo criado por Morgan (2002) para demonstrar os atores e o fluxo de informação gerado por eles neste processo.

Figura 1 - Modelo atual de desenvolvimento de sistemas



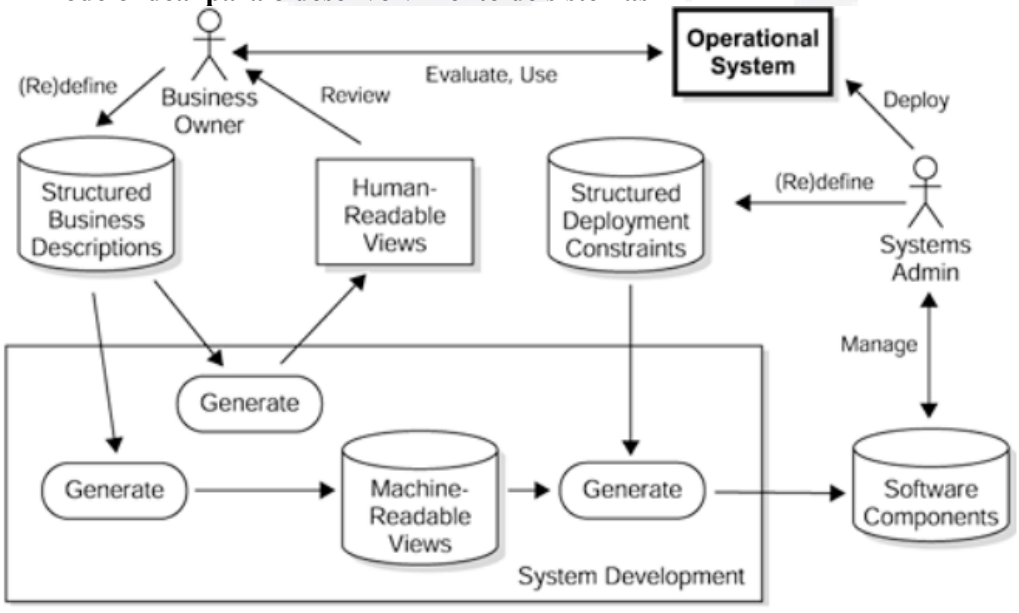
Fonte: MORGAN, 2002, p. 26.

Segundo a fonte, há muitas coisas com o que se preocupar neste modelo. Uma das principais é o fato do responsável pelo negócio (*Business Owner*), ou seja, a pessoa que precisa do sistema e que provavelmente está pagando por ele, estar fora do processo de desenvolvimento do software. Isso indica que, após uma primeira fase de elicitação de requisitos, a sua atuação fica bastante limitada, uma vez que os artefatos gerados pelos desenvolvedores (*Source Code > Generate*) são difíceis de serem entendidos por quem não possui conhecimentos técnicos e/ou treinamento adequado.

Outro ponto a ser notado é que o modelo se baseia na premissa de que o código gerado conterá erros, o que, invariavelmente, adiciona uma quantidade exagerada de testes. Assim, subentende-se que qualquer alteração no software poderá introduzir novos erros, o que torna as oportunidades de melhorias no sistema bastante limitadas.

Além disso, o processo acaba dependendo de materiais específicos que devem ser interpretados por diversas pessoas até se tornar, de fato, um produto em desenvolvimento. Basicamente, os analistas (*Analysts*) interpretam os requisitos do negócio (*Business Descriptions*) para produzirem um documento de especificações para desenvolvimento (*Structured System Descriptions*); este documento deve ser interpretado pelos desenvolvedores (*Developers*) para que, somente então, seja gerado o código-fonte (*Source Code*). Essa sequência acaba introduzindo muitos pontos de falha por questões de mal entendidos, situações que podem demorar a ser detectadas.

Figura 2 - Modelo ideal para o desenvolvimento de sistemas



Fonte: MORGAN, 2002, p. 29.

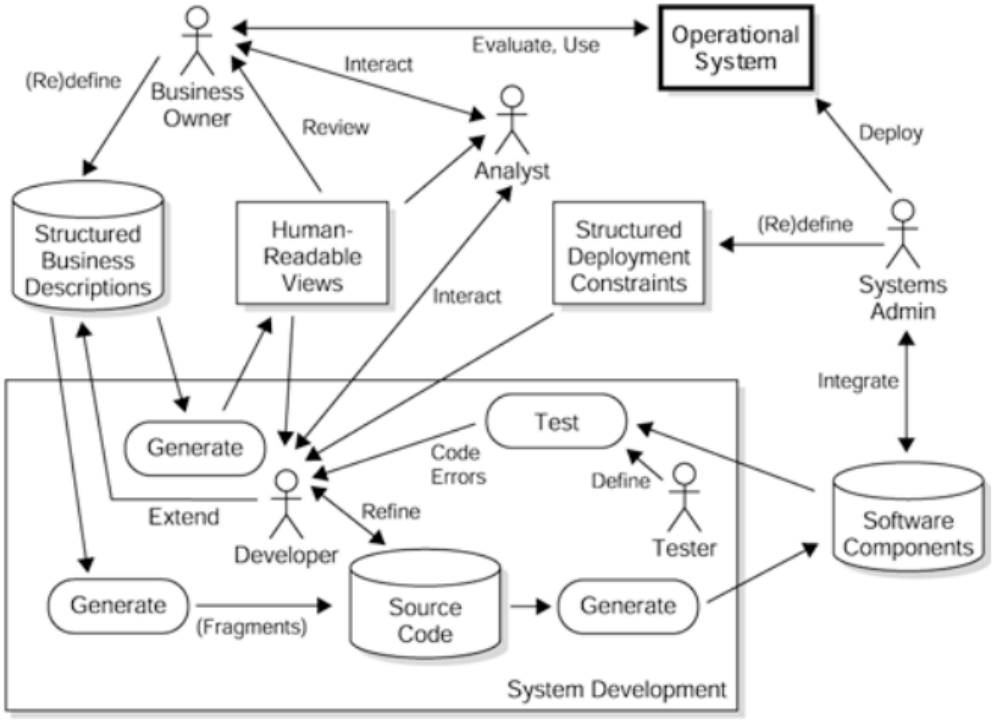
A Figura 2, também criada por Morgan (2002), apresenta o modelo indicado como ideal para a criação de sistemas. De forma geral, a única mudança ocorre na etapa de desenvolvimento (*System Development*), que passa a ser bastante simplificada, e o grande diferencial fica por conta da automação das etapas antes atribuídas aos desenvolvedores.

Para que fosse implementável, um modelo destes exigiria um repasse de conhecimento aos responsáveis pelo negócio, deixando-os aptos a descreverem as necessidades, regras e requisitos do sistema de uma forma estruturada (*Structured Business Descriptions*), seguindo alguns padrões pré-definidos. Dessa forma, seria possível a utilização de poder computacional para analisar as estruturas geradas e, posteriormente, oferecer uma visão de alto nível (*Human-Readable Views*) sobre o que foi coletado a partir das fontes informadas. Isso permitiria que o responsável percebesse problemas antes de prosseguir, de certa forma estabelecendo um contrato de acordo entre o que deve e o que será feito.

A próxima etapa, então, seria a automação do processo de geração de código-fonte. A partir das declarações de negócio definidas e estruturadas pelo responsável, uma ferramenta seria encarregada de gerar o código (*Machine-Readable Views*) automaticamente. Tudo isso levaria ao aumento da qualidade do software, pois as possibilidades de erro humano seriam praticamente eliminadas pelo processo de geração automática do código.

Embora o modelo apresentado possa ser visto como ideal, sua aplicação prática é bastante limitada. O autor cita algumas barreiras para sua implementação: inexistência de um modelo rico o suficiente para expressar todas as informações necessárias para geração de código confiável; inexistência de um consenso sobre como expressar necessidades do negócio de acordo com algumas limitações técnicas impostas pela arquitetura utilizada; necessidade de alterações pontuais para prover pequenas vantagens como, por exemplo, aumento de desempenho. Sendo improvável a implementação de tal modelo, porém, Morgan (2002) indica a possibilidade de trabalhar sobre o modelo atualmente utilizado no desenvolvimento de sistemas, indicado na Figura 3, adaptando-o e melhorando-o nos pontos mais críticos do processo.

Figura 3 - Modelo aplicável ao desenvolvimento de sistemas



Fonte: MORGAN, 2002, p. 33.

Conforme pode ser observado na Figura 3, novamente as mudanças são feitas, em sua maioria, na etapa de desenvolvimento do sistema. Basicamente, a ideia seria que o desenvolvedor utilizasse o conhecimento do negócio – passo que poderia ser mantido conforme o modelo ideal, apresentado na Figura 2 – para gerar pequenas porções e fragmentos (*Generate* > *Fragments*) de código, que na verdade seriam apenas acoplados ao código-fonte desenvolvido. Tal acoplamento poderia ser feito, por exemplo, através da chamada de um Web Service (WS)[6] que solicitaria a execução de determinada regra.

2.3.3 Aplicação das regras de negócio

Embora as abordagens voltadas às regras de negócio concebam vantagens em praticamente qualquer tipo de SI com razoável nível de complexidade, Halle (2002) cita diversos cenários do mundo empresarial como os mais apropriados para a implementação de sistemas baseados em regras. Dentre estes cenários, destacam-se:

- O negócio precisa mudar, mas o sistema é uma barreira às mudanças: muitos softwares

6 Os WSs representam componentes de uma aplicação que se comunicam utilizando protocolos abertos baseados em eXtensible Markup Language (XML) para a troca de mensagens, provendo, assim, um grande nível de interoperabilidade entre aplicações desenvolvidas em quaisquer linguagens de programação (SUDA, 2003).

entregues atualmente são vistos como caixas-pretas nas quais as regras de negócio estão embutidas e que requerem considerável esforço quando necessitam alterações. Estas, na maioria das vezes, são demoradas e têm um custo elevado;

- Criação de novas legislações que exigem aderência dos sistemas ou abrem caminho para novas oportunidades de negócio: a adesão a novas leis e regulamentações, geralmente, implica na modificação de sistemas existentes ou na criação de novos sistemas, muitas vezes com prazos a serem cumpridos. Em alguns casos, também dão a oportunidade para a implantação de novas formas de negócio se a organização percebê-las com antecedência;
- Produtos e serviços emergentes na internet: a implantação de serviços e lançamento de novos produtos em um ambiente tão volátil e ágil, como a internet, exige que as regras de negócio evoluam a todo momento;
- Fusões e aquisições de outras empresas: quando isso ocorre, faz-se necessária a consolidação de informações, base de usuários e de dados, sendo que tudo isso requer validação de políticas, práticas e regras já existentes.

2.4 Gerenciamento de regras de negócio

As organizações que possuem regras de negócio bem definidas e estruturadas, estabelecem de forma correta o conhecimento dos processos empresariais. Mas, ao mesmo tempo que isso traz vantagens, também traz mais responsabilidades: enquanto sua existência provê um meio centralizado de alterações mais ágeis, o contraponto é que se tornam mais um recurso a ser gerenciado (ROSS, 2000). No entanto, o interesse pela manipulação explícita das regras de negócio está em franco crescimento, e as pesquisas nessa área visam a um objetivo principal: identificar formas que suportem a propagação automática de atualizações das regras de negócio nos SIs que as implementam (ANDREESCU; MIRCEA, 2009).

Para Santos (2010), o gerenciamento das regras de negócio é necessário devido a aspectos internos e externos. Os primeiros dizem respeito a fatores como dificuldade em localizar e compreender as regras existentes, inconsistência ou conflito nas regras e conhecimento do negócio associado a poucas pessoas que se tornam essenciais para o funcionamento da empresa. Já os aspectos externos se referem a constantes mudanças, concorrência acirrada, relacionamento com clientes e problemas de conformidade com

regulamentações e regimentos obrigatórios.

O gerenciamento das regras de negócio, porém, é uma atividade que não deve ser feita pela TI, uma vez que, conforme Bajec e Krisper (2005b), elas não pertencem à TI. Segundo os autores, "regras de negócio são definidas pelo negócio e devem ser, portanto, gerenciadas pelo negócio". Para Gerrits (2012), considerar as regras uma responsabilidade ou um dos artefatos da TI, é colocar em risco a aplicação como um todo, podendo ter as seguintes consequências:

- Muitas regras de negócio podem ser ignoradas e consequentemente nunca implementadas no software por serem consideradas muito complexas ou complicadas de automatizar. As regras de negócio estão em todos os lugares (na mão das pessoas, em procedimentos, políticas e contratos) e são ativos valiosos da empresa;
- As regras se tornarão meros requisitos de sistema, o que pode ser visto como uma lógica de negócio que só fará sentido dentro de um contexto com objetivo particular;
- As regras acabam ficando menos reutilizáveis, pois os desenvolvedores não conhecem o contexto exato em que elas estão inseridas.

Em seu trabalho, Jacob e Froscher (1990) propõem um modelo de gerenciamento de regras que provisione seu agrupamento por afinidade. Os autores focam sua proposta na divisão das regras dentro de uma hierarquia de grupos, tendo como princípio geral, para decidir se duas regras devem ser colocadas no mesmo grupo, a complexidade de sua alteração em relação a outra regra. Ou seja: se uma alteração for feita em uma regra, até que ponto a outra será afetada?

Para Bajec e Krisper (2005b), a implementação de regras de negócio nos SIs requer funcionalidades como levantamento, especificação, modelagem, implementação, manutenção, versionamento e monitoramento das mesmas. Porém, ao passo que existe uma grande variedade de ferramentas que proveem essas facilidades aos desenvolvedores, no lado do negócio essa afirmação não é verdadeira, pois são limitadas as ferramentas que dão o suporte necessário para que pessoas sem conhecimento técnico consigam gerenciar as regras de negócio de forma apropriada.

2.4.1 Separação de regras de negócio e código-fonte

Regras de negócio representam declarações sobre como o negócio é feito, ou seja, sobre diretrizes e restrições associadas aos estados e processos de uma organização

(HERBST; MYRACH, 1997). Embora estejam relacionadas ao contexto de negócio da empresa, nos SIs, geralmente, elas costumam aparecer embutidas diretamente em seu código-fonte (WILLMOR; EMBURY, 2006).

Essa forma de desenvolver sistemas, no entanto, traz à tona um problema em particular: o usuário que opera o sistema pode não saber como funciona a regra de determinada operação e até mesmo os desenvolvedores, em muitos casos, não sabem exatamente como se comporta alguma parte ou funcionalidade do sistema. Tais afirmações ficam claras na ideia de Halle (2002):

> Infelizmente, as regras de negócio são frequentemente inacessíveis ou, pior, desconhecidas. Este é o caso em que regras de negócio estão ocultas no código legado, para o qual existe pequena ou nenhuma documentação. É assustador pensar na execução de regras, que interessam ao negócio, que permanecem escondidas daqueles que utilizam o sistema e daqueles que querem fazer mudanças em sua lógica. Quando tais regras são inacessíveis ou desconhecidas, as pessoas (incluindo os desenvolvedores do sistema) fazem suposições sobre elas que podem estar incorretas ou inconsistentes. Tais suposições levam a um comportamento (humano ou eletrônico) que não é bem orquestrado, nem efetivamente focado em objetivos comuns, e certamente incapaz de mudanças fáceis e adaptabilidade.

Conforme Kovacic (2004), os SIs desenvolvidos na forma tradicional não estão aptos a acompanharem as mudanças requeridas pelo mundo empresarial, visto que as regras de negócio costumam ser atualizadas constantemente. Para Bajec e Krisper (2005a), mudanças raramente acontecem de forma espontânea, sendo, geralmente, o resultado de decisões provenientes da cúpula administrativa ou de fontes externas como normas governamentais, que determinam novas leis ou mudanças de tributação, por exemplo. Nesses casos, o impacto, geralmente, ocorre nas regras de negócio e sua implementação, que precisam ser reavaliadas e modificadas para atenderem aos novos objetivos, políticas e valores.

Os envolvidos com a TI, porém, geralmente não possuem o conhecimento necessário para saber como e quando uma regra mudará, tampouco o impacto que tal alteração terá sobre outras regras. Segundo Narayanan (2009), tais fatos comprovam a necessidade de manter as regras de negócio fora da aplicação, permitindo que analistas de negócio, por exemplo, possam gerenciá-las de forma contínua sempre que necessário.

Para exemplificar o que foi mencionado, pode-se imaginar um sistema de vendas online de uma empresa de varejo. Considerando-se a competitividade acirrada do setor, devido à grande variedade e quantidade de opções disponíveis, a empresa que deseja alcançar êxito nesse ramo precisa, constantemente, lançar novas promoções sobre determinados tipos de produtos e/ou público-alvo. A definição de tais ofertas, porém, ocorre em âmbito gerencial,

na maioria das vezes pela equipe comercial. Ter estas regras acopladas no código-fonte da aplicação, porém, prejudica a dinamicidade de um sistema desse porte, pois a implementação das funcionalidades depende exclusivamente da TI.

De acordo com Morgan (2002), a codificação das regras diretamente no sistema é um dos maiores erros cometidos durante o desenvolvimento das aplicações. O resultado dessa ação é que as atualizações de regras requerem a rescrita do software, o que, consequentemente, torna sua realização lenta e mais cara. Para Halle (2002), ter as regras embutidas numa caixa-preta que exige intervenções de especialistas para futuras modificações, torna os SIs menos receptíveis por empresas que buscam atualizar seu negócio constantemente. A autora afirma que "você não pode mais entregar as regras em um formato inacessível e não entendível à audiência do negócio. Você não pode mais deixar as regras de negócio em um local onde elas se percam".

Regras de negócio, essenciais para o funcionamento de uma organização, devem ser automatizadas sempre que possível, embora sua implementação em SIs seja uma das partes mais críticas no processo de desenvolvimento de sistemas (ROSS, 2013). Empregando a abordagem correta para seu gerenciamento, porém, é necessário suportar suas constantes modificações sem que seja preciso alterar a aplicação, provendo a externalização das regras – pelo menos as mais críticas – em relação ao sistema que as utiliza (NARAYANAN, 2009).

As regras de negócio podem ser consideradas como a parte principal de um software, entretanto, implementá-las dentro de um sistema pode acarretar em redundância ou incompletude. Além disso, mesmo quando programadas de forma consistente, as regras – sua definição ou seu conhecimento – podem se perder conforme o passar do tempo ou com o excesso de modificações (HERBST et al., 1994). Embora não seja algo novo, a separação entre regras de negócio e código-fonte promove vantagens como facilidade de atualização, reutilização de regras e escalabilidade. Permitir que as regras de negócio sejam definidas e gerenciadas separadamente à aplicação pode ser visto como o estado da arte no âmbito do desenvolvimento de SIs (GOTTESDIENER, 1997).

2.4.2 Sistemas de gerenciamento de regras de negócio

A implementação de uma única regra de negócio aparecerá, tipicamente, em muitos programas diferentes, podendo vir a ocorrer mais de uma vez dentro do mesmo programa,

geralmente feita de diversas formas. A correta aplicação das regras de negócio nos SIs, porém, é vital para muitas organizações, em especial àquelas nas quais as regras são definidas, na sua maioria, por regulamentos estatutários e nas quais as multas e penalidades podem ser aplicadas em casos de violação. Sempre que possível, as regras de negócio devem ser centralizadas em repositórios de regras e executadas por meio de motores de regras (WILLMOR; EMBURY, 2006).

Conhecidos como BRMS, os sistemas de gerenciamento de regras de negócio ganharam muita atenção da indústria de desenvolvimento de sistemas nos últimos anos. Tendo como principal objetivo a separação das regras visando prover um local centralizado para atualizações, essa nova categoria de software permite solucionar um clássico problema entre as áreas de negócio e TI: a alta frequência com que as regras de negócio precisam ser alteradas, exige que os SIs que as contemplam acompanhem o mesmo ritmo de mudanças (ANDREESCU; MIRCEA, 2009).

2.4.2.1 Sistemas especialistas e o motor de inferências

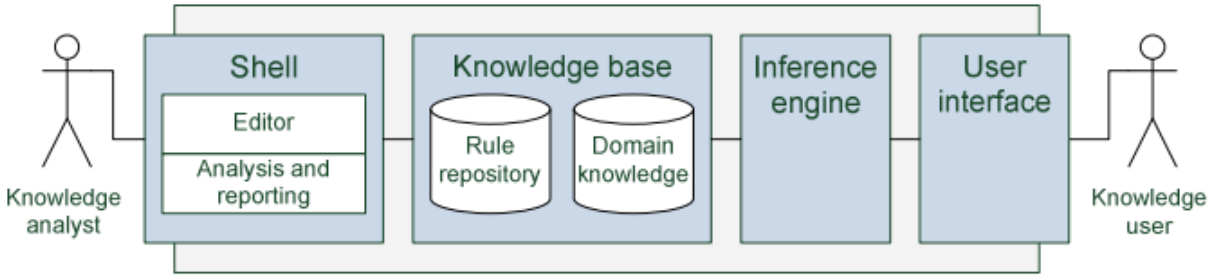
Embora o interesse e a popularidade de sistemas BRMS tenha crescido apenas recentemente, sua origem data do início da década de 1980, quando do lançamento de uma ferramenta chamada EMYCIN. Baseado em um sistema especialista da área médica conhecido como MYCIN[7], o EMYCIN propôs o primeiro modelo de gerenciamento de regras a partir de sua retirada do código-fonte e posterior externalização. Além disso, a ferramenta permitia a utilização de qualquer domínio de informações – podendo ser utilizado para qualquer área de interesse – e possibilitava a interação com o usuário, baseando-se nas respostas para estabelecer os melhores resultados e conclusões possíveis (GRAHAM, 2007).

A Figura 4 apresenta o modelo de arquitetura por trás de um sistema especialista – também conhecido como sistema baseado em conhecimentos – semelhante ao EMYCIN. Pode-se observar a existência de um componente específico (*knowledge base*) que contém tanto as regras de negócio (*rule repository*) quanto o domínio de informações referente ao modelo do sistema (*domain knowledge*). As informações desse componente são manutenidas por uma pessoa que possui conhecimento técnico para tal (*knowledge analyst*), a partir da utilização de uma ferramenta que o auxilie nessa tarefa (*shell*). Na outra ponta da arquitetura,

7 Um dos primeiros sistemas especialistas desenvolvidos, lançado em 1976 com o objetivo de diagnosticar doenças infecciosas no sangue (GRAHAM, 2007), o MYCYN não pode ser considerado um BRMS porque suas regras estavam diretamente acopladas ao código-fonte da aplicação.

o usuário do sistema (*knowledge user*) utiliza uma interface específica (*user interface*) que consulta o componente mais complexo do modelo: o motor de inferências (*inference engine*).

Figura 4 - Arquitetura de um sistema especialista



Fonte: KOORNNEEF, 2006, p. 19.

O motor de inferências pode ser visto como o mecanismo que aplica o conhecimento sobre o conjunto de dados, com o objetivo de obter conclusões válidas de acordo com premissas (GRAHAM, 2007). Tipicamente, é o responsável por disparar as regras quando forem detectados determinados padrões de equivalência (ROSENBERG; DUSTDAR, 2005). Conforme Zandipour (2011), o motor de inferências é composto por dois componentes: um *pattern matcher* (combinador de padrões) e uma agenda. O *pattern matcher* é o elemento que realiza as comparações entre os fatos e as regras, enquanto a agenda se comporta como uma fila de regras a serem disparadas – apenas as que forem julgadas apropriadas pelo primeiro.

Ochem (2008) indica que o mecanismo de funcionamento do combinador de padrões se baseia em *forward chaining* (encadeamento dianteiro) ou *backward chaining* (encadeamento retrógrado), padrões utilizados para a verificação de estruturas de dados em relação a diversas condições, ambos os conceitos provenientes da IA. O autor define os dois padrões da seguinte forma:

> *Forward chaining* verifica as condições das regras até encontrar uma que corresponda, e então utiliza esta regra para fazer deduções em outras regras que têm as mesmas condições. *Backward chaining* faz o processamento em ordem inversa, iniciando de conclusões até os fatos antecedentes.

Para Graham (2007), a utilização de *forward chaining* é indicada em situações nas quais a quantidade de resultados costuma ser pequena, embora a coleta e processamento das informações seja uma tarefa custosa. O *backward chaining*, por outro lado, realiza o processo contrário, sendo uma opção mais indicada para os casos em que a quantidade de informações resultante é muito grande. Segundo o autor, ainda existe um terceiro paradigma baseado num modelo misto, identificado como *opportunistic chaining,* que, de fato, é o que costuma ser empregado nas principais ferramentas de BRMS da atualidade.

2.4.2.2 BRMS

Koornneef (2006) e Ochem (2008) caracterizam os sistemas de gerenciamento de regras de negócio (BRMS) como uma evolução dos sistemas especialistas, criados especificamente para trabalhar com conceitos vinculados ao negócio da organização, e que, geralmente, são utilizados dentro de arquiteturas de Sistemas de Apoio à Decisão (SAD)[8]. Para Graham (2007), um BRMS constitui a forma mais prática e econômica de implantação de uma abordagem de sistemas voltada às regras de negócio.

A Tabela 3 apresenta um modelo de características e responsabilidades atribuído a um BRMS.

Tabela 3 - Características e responsabilidades de um BRMS

Conceito	Descrição
Armazenamento e manutenção	Prover o armazenamento e a manutenção de regras de negócio em um repositório centralizado, responsável por conter informações pertinentes ao funcionamento da empresa.
Externalização	Permitir que as regras de negócio e sua lógica se mantenham separadas do código-fonte da aplicação.
Integração	Proporcionar a integração com outros sistemas da organização, para que as regras possam ser utilizadas em todos os processos de tomada de decisões.
Conjuntos e inferência	Formalizar regras dentro de conjuntos independentes que podem ser acoplados, realizando inferências sobre estes visando à obtenção de novas regras.
Facilidade de utilização	Permitir que pessoas sem conhecimento técnico em TI, como muitos analistas de negócio e até mesmo usuários do sistema, consigam realizar a manutenção de regras com o mínimo de aprendizado necessário.
Inteligência e naturalidade	Possibilitar a criação de aplicações inteligentes, a partir da interação com o usuário de forma natural e lógica, com o objetivo de automatizar e facilitar os processos de negócio.

Fonte: do autor, adaptado de Graham (2007).

Os primeiros e mais simples BRMSs ofereciam, basicamente, apenas um ambiente para definição e gerenciamento de regras para usuários com conhecimento técnico, ou seja, as regras eram escritas em uma linguagem de difícil entendimento para as pessoas do negócio. O avanço destes sistemas, porém, proporcionou a oferta de ferramentas mais avançadas que incluem interfaces gráficas para a criação de regras e integração com planilhas, softwares tipicamente utilizados pelas pessoas do negócio. A adoção de um BRMS também traz diversas

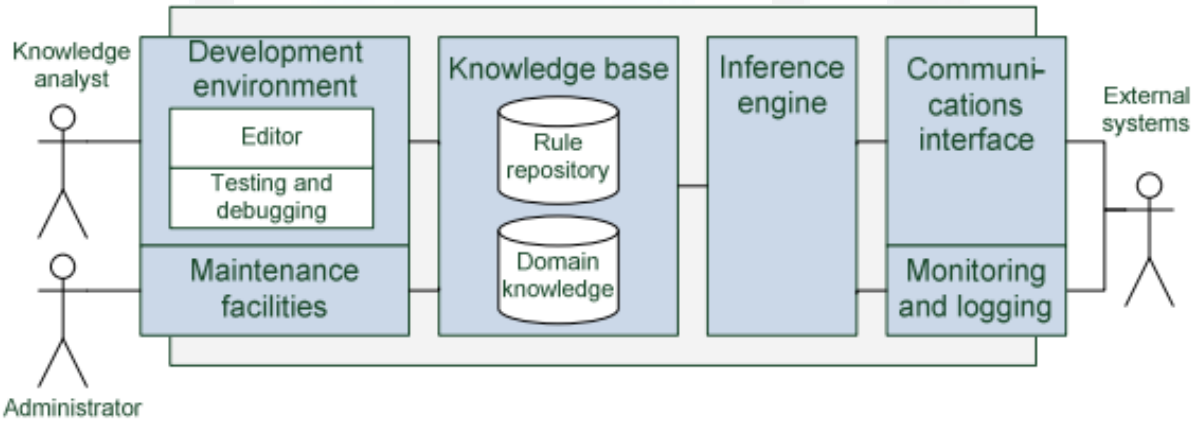
8 Um SAD pode ser visto como um sistema computacional capaz de ajudar o usuário no julgamento e escolha de atividades ou decisões a serem tomadas (DRUZDEL; FLYNN, 2002).

vantagens à organização, como a diminuição da dependência do departamento de TI, implantação acelerada de novas regras nos seus SIs e possibilidades de simular os efeitos decorrentes da alteração de alguma regra (ZANDIPOUR, 2011).

A arquitetura de um BRMS, conforme Graham (2007), é composta por quatro componentes principais: o ambiente tecnológico no qual o sistema está inserido, que inclui linguagem de programação, compiladores, estruturas de dados, etc; a estrutura da base de conhecimentos, onde ficam armazenadas as informações pertinentes ao domínio do negócio; o motor de inferências, parte responsável por avaliar as regras e obter conclusões válidas a partir de sua verificação, e, por último, o repositório de regras, que deve gerenciar, versionar e compartilhar as regras de negócio.

A Figura 5 apresenta a arquitetura de um BRMS contendo todos os componentes citados, bem como os atores envolvidos no processo de utilização do sistema. Basicamente, existe um componente (*Knowledge base*) que contém os elementos referentes à representação do conhecimento, ou seja, o domínio do negócio (*Domain knowledge*) e o repositório de regras (*Rule repository*). Estes são utilizados pelo analista de negócio (*Knowledge analyst*), para manutenção e gerenciamento das regras através do ambiente de desenvolvimento provido (*Development environment*), e pelo motor de inferências (*Inference engine*). O ambiente tecnológico pode ser visto como a área que circunda todos os componentes citados.

Figura 5 - Arquitetura de um BRMS



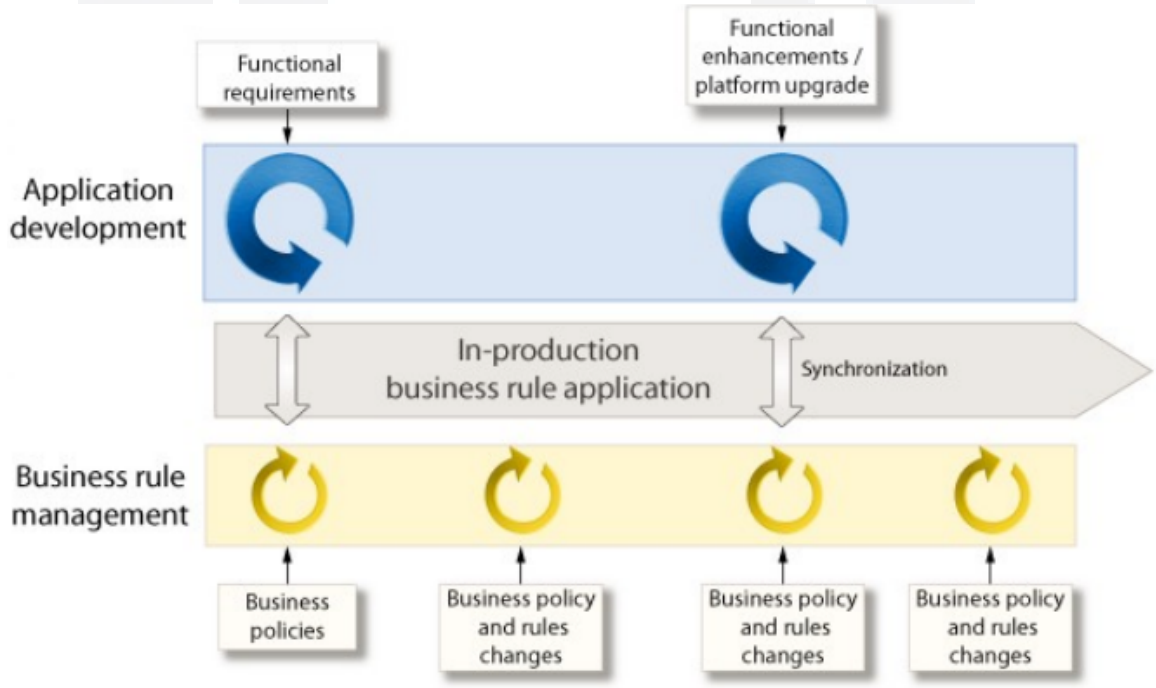
Fonte: KOORNNEEF, 2006, p. 30.

Para Koornneef (2006), a arquitetura de um BRMS é muito semelhante à de um sistema especialista, o que pode ser comprovado pela comparação das Figuras 4 e 5. Conforme o autor, a grande diferença reside no fato de que os BRMSs permitem a exposição

de suas funcionalidades via WS ou Application Programming Interface (API)[9], possibilitando a utilização de suas regras por sistemas externos e, portanto, substituindo o modelo orientado a conversações com o usuário proveniente dos sistemas especialistas. Segundo Andreescu e Mircea (2009), este é um dos principais motivos para que os sistemas especialistas não tivessem a adoção esperada no mundo dos negócios, ao passo que os BRMSs foram especialmente projetados para prover serviços que automatizam aplicações envolvendo o negócio da organização.

Segundo Zandipour (2011), um importante benefício provido por um BRMS é a separação entre os ciclos de vida da aplicação e de suas regras de negócio, conforme ilustra a Figura 6. Para o autor, as regras mudam com uma frequência consideravelmente maior do que a aplicação que as utiliza, o que significa ser possível terminar com a necessidade de recompilar ou parar o sistema inteiro a todo momento. Com a utilização de um BRMS, as regras são alteradas enquanto a aplicação está rodando, sem a necessidade de alterá-la.

Figura 6 - Separação dos ciclos de vida de um sistema e suas regras de negócio, através da utilização de um BRMS



Fonte: ZANDIPOUR, 2011, p. 31.

Para que se alcance êxito na implantação e utilização de um BRMS dentro de uma organização, é recomendável a existência de cinco papéis vinculados tanto ao contexto do

9 Uma API serve como uma interface que define como um software deve se comportar para acessar as funcionalidades de outro que implementa esta API (3 SCALE, 2012).

sistema quanto da organização como um todo, assim como mostra a Tabela 4.

Tabela 4 - Papéis envolvidos com um BRMS

Papel	Descrição
Analista de negócios	É quem realiza o intermédio entre as áreas do negócio e da TI. Deve ter muito conhecimento sobre o negócio e, pelo menos, o básico de TI, para conseguir traduzir as políticas da empresa em um modelo entendível aos desenvolvedores.
Desenvolvedor	É o profissional da informática responsável por mapear o vocabulário criado pelo analista de negócio, em uma estrutura subjacente ao sistema.
Gerenciador de políticas	É quem define as políticas da organização. Geralmente, não possui conhecimentos técnicos de TI, porém, costuma ser um profissional com bastante experiência em ferramentas auxiliares como planilhas.
Administrador do ambiente	É o profissional que gerencia e configura o ambiente e a instalação do BRMS.
Arquiteto de regras	É quem faz o trabalho de otimizar as regras criadas e garante que as mesmas pertencem ao conjunto de regras apropriado.

Fonte: do autor, adaptado de Zandipour (2011).

Em seu trabalho, Graham (2007) afirma que os sistemas de BRMS devem possuir algumas características específicas para que atinjam um estado de maturação. Halle (2002), por outro lado, cita aspectos de tratamento necessários às regras para que um BRMS possa ser considerado apto a servir adequadamente ao negócio e a seus gestores. As Tabelas 5 e 6, respectivamente, ilustram essas características e aspectos.

Tabela 5 - Características recomendadas a um BRMS

Característica	Descrição
Casos de sucesso	Possuir casos de sucesso em termos de softwares comerciais.
Disponibilização	Permitir que as regras sejam disponibilizadas como serviços, possibilitando sua utilização por outras aplicações.
Facilidade	Possibilitar que analistas de negócio criem e gerenciem as regras.
Logs e relatórios	Prover facilidades em termos de geração de *logs* e relatórios.
Suporte	Possuir uma equipe de suporte profissional.
Usabilidade[10]	Possuir boa usabilidade, de forma que os usuários consigam descobrir a ferramenta enquanto a utilizam.

Fonte: do autor, adaptado de Graham (2007).

10 A usabilidade é uma característica que define a facilidade de utilização de determinado software ou ferramenta, sendo, atualmente, algo essencial para o sucesso das aplicações que visam à conquista de um grande público (KRUG, 2006).

Tabela 6 - Aspectos de tratamento de regras em um BRMS

Aspecto	Descrição
Separação	Permitir a separação da regra de negócio do código-fonte e da lógica de implementação, tornando-as reutilizáveis.
Rastreabilidade	Facilitar a análise de impacto sobre mudanças, possibilitando o rastreamento de todos os envolvidos com a regra.
Externalização	Prover acesso para todas as pessoas que precisam conhecer as regras, fazendo com que estas não sejam mais um produto interno da TI.
Preparação para mudanças	Possibilitar mudanças nas regras a qualquer momento, com o mínimo de esforço necessário.

Fonte: do autor, adaptado de Halle (2002).

Shpigel (2011) resume os BRMSs da seguinte forma:

> Um BRMS salva as regras em um repositório de regras e compartilha-as por múltiplas aplicações e processos dentro da organização. As decisões dos BRMSs são baseadas na avaliação de conjuntos de várias regras, que podem estar amarradas a outras regras. Além disso, as regras são aplicadas a qualquer aplicação que as invoca, portanto, uma vez feita uma alteração, esta é aplicada a todos os locais que a utilizam.

No que se refere à representação das regras de negócio, sistemas BRMS geralmente suportam e possibilitam a utilização de regras em diversos formatos. Os modelos mais citados por autores, entre eles Graham (2007), Andreescu e Mircea (2009) e Zandipour (2011), são Evento-Condição-Ação (ECA) e Tabelas de Decisão. A Tabela 7 apresenta um resumo de cada modelo.

Tabela 7 - Modelos comuns para representação de regras de negócio em BRMSs

Modelo	Descrição
Evento-Condição-Ação (ECA)	Também conhecido como "SE-ENTÃO", "QUANDO-ENTÃO" ou regras de produção, o modelo ECA é bastante simples e semelhante a condições *if-else* de linguagens de programação. Exemplo: *"Se a forma de pagamento for boleto bancário, então um desconto de 10% deve ser concedido"*. Em um BRMS, esta regra de exemplo poderia ser representada de forma semelhante ao apresentado na Listagem 1.
Tabelas de Decisão	Modelo para representação de regras em tabelas, nas quais cada linha é uma regra de negócio. Uma de suas vantagens é o formato mais simples que pode ser criado com aplicativos como Microsoft Excel ou LibreOffice Calc. A Figura 7 ilustra um exemplo de tabela de decisão.

Fonte: elaborado pelo autor.

Listagem 1 - Exemplo de regra no formato "quando-então", conforme utilizado pelo Drools

```
rule "Conceder desconto para um pagamento com boleto bancário."
    when
        $venda.tipoPagamento == 'BB'
```

```
        then
            $venda.valor = ( $venda.valor * 0.9 )
    end
```

Fonte: elaborador pelo autor.

Figura 7 - Exemplo de tabela de decisão

Desconto no valor do seguro do carro		
Idade do cliente		Desconto
Mínima	*Máxima*	
18	30	7,50%
31	60	10,00%
61	80	5,00%

Fonte: elaborado pelo autor.

A implementação e escrita das regras de negócio em SIs, independentemente da ferramenta BRMS escolhida, sempre ocorrerá de forma diferente do que em linguagens de programação tradicionais. Para completar o tema, Browne (2009) resume da seguinte forma:

> Linguagens de programação tradicionais são mais como um conjunto de instruções: faça a etapa 1, faça a etapa 2, repita 5 vezes, e assim por diante. Regras são diferentes; elas permitem que você faça sentenças individuais sobre o que você sabe ser verdadeiro e então deixe o computador decidir se estas regras se aplicam (ou não) à situação atual. Isto é semelhante à maneira com que a mente humana trabalha.

3 FERRAMENTAS

O foco do presente trabalho é a análise e comparação de diferentes ferramentas para a gestão de regras de negócio. Neste capítulo, é dada uma visão geral acerca dos BRMSs escolhidos para tal, apresentando suas principais características e funcionamento básico.

As ferramentas utilizadas nesta análise exploratória foram o JBoss Drools e o OpenL Tablets, ambas gratuitas e de código-fonte aberto. A escolha foi feita, principalmente, levando-se em consideração a abrangência da primeira e a simplicidade da segunda, além da utilização e popularidade das duas no atual mercado de desenvolvimento de sistemas corporativos[11]. A escolha das ferramentas, aliás, foi uma das dificuldades iniciais do trabalho, visto que os projetos de pesquisa encontrados levam em consideração apenas os BRMSs pagos, tornando mais difícil o processo de análise por não haver outro trabalho semelhante com o qual fosse possível comparar os resultados obtidos.

Cabe ressaltar que as duas ferramentas apresentadas a seguir serão exploradas com mais detalhes no Capítulo 4. A escolha delas levou em consideração alguns aspectos como: código-fonte livre, documentação, popularidade, consolidação, simplicidade, abrangência.

3.1 JBoss Drools

JBoss Drools é uma ferramenta mantida pelas empresas JBoss e Red Hat. O projeto é totalmente escrito em linguagem Java, gratuito e com código-fonte aberto, sendo licenciado sob a Licença Apache 2.0[12], e sua primeira versão foi lançada em 2001. Atualmente na versão 5.5, lançada em 13/11/2012, o JBoss Drools é uma plataforma única e integrada para manutenção de regras, processos de negócio e processamento de eventos.

A ferramenta possui código-fonte livre, vasta documentação e comunidade ativa na internet, além de constituir uma solução bastante abrangente. O Drools fornece a possibilidade de gerência, testes, auditoria e versionamento de regras através de linguagem específica ou interface *web*, execução de regras de forma remota, disponibilização das regras

11 Devido à inexistência de um estudo que indique a utilização de BRMSs gratuitos, o critério de utilização levou em consideração o número de resultados encontrados sobre as ferramentas na *internet*, tanto em *sites* quanto em fóruns de tecnologia da informação. O único estudo oficial encontrado – conhecido como Worldwide Business Rules Management Systems Vendor Shares – considera apenas softwares que são comercializados. A versão de 2011 dele pode ser vista em ftp://public.dhe.ibm.com/software/websphere/odm/2011_BRMS_MarketShare_Report.pdf.

12 A Licença Apache é uma licença de software de autoria da Apache Software Foundation, que indica que o código-fonte do software pode ser distribuído e modificado desde que haja a inclusão do aviso de *copyright*.

via WSs, tabelas de decisão, integração com diversos *frameworks* Java, entre outras.

O Drools possui diversos módulos criados de forma separada para facilitar o trabalho de implantação e manutenção da ferramenta (TABELA 8). Este trabalho faz o estudo de dois dos seus componentes, Drools Expert e Drools Guvnor, que são diretamente relacionados entre si e caracterizam as funcionalidades essenciais de um BRMS.

Tabela 8 - Componentes do JBoss Drools

Componente	Descrição
Drools Expert	Considerado o núcleo do projeto, representa o motor de regras/inferência e é usado para especificar, manter e executar as regras de negócio.
Drools Guvnor	É reconhecido como o gerenciador das regras de negócio, servindo como o repositório central das bases de conhecimento da ferramenta.
Drools Fusion	Módulo para processamento de eventos complexos que visa identificar tarefas significativas a partir do processamento de um grande número de requisições.
Drools Flow	Também conhecido como jBPM 5, permite representar os processos de negócio visualmente, visando ao entendimento por parte tanto do negócio quanto da TI.
Drools Planner	Módulo para agendamento e execução dinâmica de regras.

Fonte: elaborado pelo autor.

3.1.1 Drools Expert

O Drools Expert é o módulo principal do projeto, servindo para especificar, manter e executar as regras de negócio. É considerado, assim, o motor de regras da ferramenta – também chamado de motor de inferências, explicado no item 2.4.2.1 do Capítulo 2. É denominado o núcleo do projeto, sendo parte essencial do mesmo, pois, sem ele, não existe a base de conhecimentos do sistema. Todo o trabalho feito diretamente sobre o Drools Expert requer o conhecimento acerca de três tipos diferentes de arquivos, que indicam como deve ser feita a declaração das regras através da ferramenta (TABELA 9).

Tabela 9 - Tipos de arquivos do Drools Expert

Extensão	Descrição
DRL	Armazena as regras escritas com a linguagem específica do Drools.
DSL	Armazena sentenças e definições específicas do domínio do sistema, visando a um melhor entendimento das regras.
DSLR	Armazena as regras que utilizam as sentenças e definições DSL.

Fonte: elaborado pelo autor.

Basicamente, uma regra escrita em linguagem nativa do Drools, chamada de Drools Rule Language (DRL), exige apenas quatro componentes (LISTAGEM 2): o nome de um pacote, que a identifique dentro de um grupo; um nome único; condições para sua execução, conhecidas como Left Hand Side (LHS), e as consequências/ações que devem ser realizadas, conhecidas como Right Hand Side (RHS). A Listagem 3 apresenta um exemplo de regra que verifica se um cliente é inválido através da verificação de seu telefone e endereço.

Listagem 2 - Formato básico de um arquivo DRL

```
package br.com.pacote
rule "nome"
    when
        LHS
    then
        RHS
end
```

Fonte: elaborado pelo autor.

Listagem 3 - Exemplo de regra no formato DRL

```
package com.brunozambiazi.tcc
rule "Verificar se o cliente é inválido"
    when
        $cliente : Cliente( endereco == null || telefone == null )
    then
        System.out.println( "O cliente "+ $cliente + " é inválido
pois não possui endereço ou telefone." );
end
```

Fonte: elaborado pelo autor.

Para a interpretação das partes LHS e RHS, o Drools Expert trabalha com estruturas de dados que podem ser adicionadas aos arquivos DRL, acima da parte de regras, embora o mais indicado seja a existência de um único arquivo contendo todas elas, deixando as regras em arquivos separados. A Listagem 4 apresenta um exemplo de definição dentro de um DRL, no qual é definido o pacote ao qual a estrutura pertence, o nome dela e seus atributos, com um identificador e seu tipo correspondente a uma classe Java.

Listagem 4 - Estrutura de dados Servidor no formato DRL

```
package com.brunozambiazi.tcc
declare Cliente
   nome : String
```

```
    idade : Integer
    endereco : String
    telefone : String
end
```

Fonte: elaborado pelo autor.

Para a execução das regras, o Drools Expert disponibiliza uma API que pode ser acessada tanto em ambientes com a linguagem de programação Java quanto em ambientes .NET. A Listagem 5 apresenta um trecho de código Java que utiliza esta API e realiza as seguintes operações: cria uma nova base de conhecimentos (2) a partir do arquivo regras.drl (1); cria um novo fato com base numa estrutura de dados do tipo Cliente, instanciando o objeto (3) e setando suas informações (4); insere o Cliente na sessão e dispara as regras (5).

Listagem 5 - Trecho de código Java que utiliza a API do Drools Expert

```
// (1)
KnowledgeBuilder builder = KnowledgeBuilderFactory.newKnowledgeBuilder();
builder.add( ResourceFactory.newClassPathResource( "regras.drl" ),
ResourceType.DRL );

// (2)
KnowledgeBase base = builder.newKnowledgeBase();
StatefulKnowledgeSession session = base.newStatefulKnowledgeSession();

// (3)
FactType cliente = base.getFactType( "com.brunozambiazi.tcc", "Cliente" );
Object clienteBruno = cliente.newInstance();

// (4)
cliente.set( clienteBruno, "nome", "Bruno" );
cliente.set( clienteBruno, "idade", 25 );
cliente.set( clienteBruno, "telefone", "12345678" );

// (5)
session.insert( clienteBruno );
session.fireAllRules();
```

Fonte: elaborado pelo autor.

O DRL, embora bastante flexível, se caracteriza como algo entendível apenas por desenvolvedores e pessoas com conhecimento técnico em linguagens de programação. Segundo Bali (2009), o Domain Specific Language (DSL) visa amenizar este problema, pois é uma forma de representar as sentenças das regras de negócio de uma forma semelhante à linguagem natural. Conforme o autor, o domínio representa a área de negócio e suas regras devem ser expressas com as terminologias pertinentes a tal.

Em um arquivo DSL, cada sentença é expressa em uma linha. A Listagem 6 mostra o formato de especificação de cada uma, na qual é necessária a informação do escopo (*scope*)[13], tipo de estrutura (*type*)[14], definição (*definition*) e código (*code*). Na Listagem 7 é visto um DSL referente ao DRL de validação de um cliente, apresentado anteriormente na Listagem 3.

Listagem 6 - Formato básico de um DSL

```
[<scope>][<type>]<definition>=<code>
```

Fonte: elaborado pelo autor.

Listagem 7 - Exemplo de sentenças de regras no formato DSL

```
[condition][]Cliente inválido=
   $cliente : Cliente( endereco == null || telefone == null )
[consequence][]Exibir mensagem de erro para Cliente=
   System.out.println( "O cliente "+ $cliente + " é inválido pois não possui
endereço ou telefone." );
```

Fonte: elaborado pelo autor.

Para que seja possível trabalhar com as sentenças definidas em um DSL, é necessário criar as regras em um arquivo DSLR. A diferença básica deste tipo de arquivo para um DRL é a necessidade de utilizar o atributo *expander* acima da definição das regras, indicando qual o arquivo de domínio que será utilizado para a resolução delas. A Listagem 8 apresenta como seria um DSLR para validação de um cliente, com base na DSL apresentada na Listagem 7.

Listagem 8 - Exemplo de regra no formato DSLR

```
expander clientes.dsl
rule "Verificar se o cliente é inválido"
    when
        Cliente inválido
    then
        Exibir mensagem de erro para Cliente
end
```

Fonte: elaborado pelo autor.

3.1.2 Drools Guvnor

O Drools Guvnor é o módulo do Drools que possui uma interface *web* para gerência e

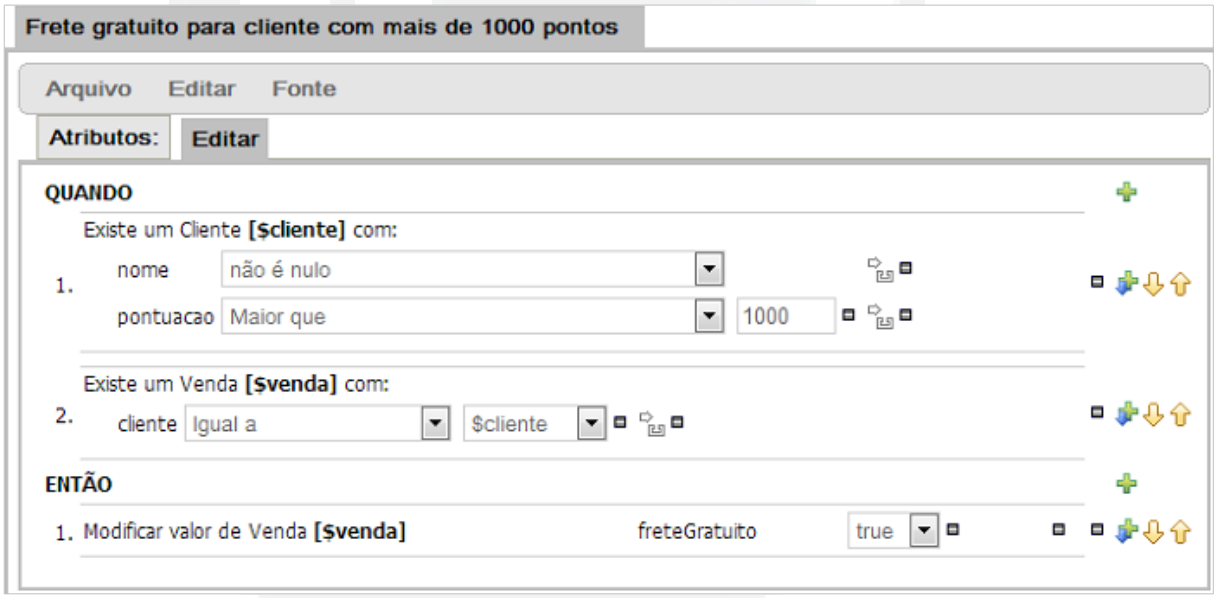
13 O escopo da sentença de um DSL pode ser *condition* (quando representa uma condição, ou LHS) ou *consequence* (quando representa uma ação a ser realizada, ou RHS).

14 O tipo de estrutura de um DSL indica se aquela sentença é aplicável de forma geral ou a apenas alguma estrutura em específico, como o Cliente dos exemplos. É um elemento opcional.

manutenção das regras de negócio. Trabalha em conjunto com o módulo Drools Expert, uma vez que boa parte de suas operações resulta em tarefas executadas no módulo citado – como criação, edição e teste das regras. O Drools Guvnor oferece a possibilidade de gerenciar diversas bases de conhecimento, dentro das quais é possível a criação ilimitada de pacotes, regras, cenários de testes, entre outros.

Um dos diferenciais do Drools Guvnor é sua tela para a criação de regras de negócio, conhecida como editor guiado. Funcionando como uma interface gráfica para a geração dos arquivos DRL, DSL e DSLR, ela não requer nenhum tipo de conhecimento em linguagens de programação e nem mesmo da sintaxe dos arquivos citados, embora seja possível editar as regras de forma manual.

Figura 8 - Editor guiado para criação de regras do Drools Guvnor



Fonte: elaborado pelo autor.

A Figura 8 demonstra uma regra criada com o editor guiado. O segmento denominado "Quando" possui as condições da regra, enquanto a parte "Então" possui as ações a serem realizadas. No exemplo apresentado, a regra indica que não deve ser cobrado o frete para uma venda associada a um cliente cujo nome não é nulo e a pontuação é maior do que mil.

O Drools Guvnor também oferece a possibilidade de versionamento das regras, facilitando, assim, a correção de problemas causados devido a alterações defeituosas, e dispõe de opções administrativas para categorização de regras e estruturas, possibilitando controle de acesso às categorias de objetos. Além disso, permite que toda a estrutura de dados e informações seja salva em SGBDs como Oracle, MySQL, PostgreSQL, entre outros.

3.2 OpenL Tablets

OpenL Tablets é uma ferramenta escrita em Java e mantida pela empresa Exigen Services. O projeto é gratuito e de código-fonte aberto, utiliza a licença LGPL[15] 3 e sua primeira versão foi lançada em 2004. Atualmente na versão 5.10.2, lançada em 06/06/2013, o OpenL Tablets é um BRMS que se baseia na execução de regras extraídas a partir de tabelas criadas com softwares de planilhas eletrônicas – Microsoft Excel e semelhantes.

A ferramenta possui código-fonte livre, ótima documentação e está em constante evolução. Basicamente, o OpenL Tablets pode ser considerado um processador avançado de tabelas que extrai informações de planilhas como o Microsoft Excel, por exemplo, e as torna acessíveis para programas escritos em Java.

O OpenL Tablets possibilita a criação de regras de duas formas: utilizando qualquer software de planilhas eletrônicas que suporte arquivos no formato *.xls* e um ambiente *web* conhecido como Web Studio. Ambas as possibilidades serão exploradas neste trabalho.

3.2.1 Planilhas eletrônicas

O OpenL Tablets suporta uma grande variedade de modelos de tabelas criadas diretamente com editores de planilhas eletrônicas, tais quais Microsoft Excel e LibreOffice Calc. A única restrição é quanto ao formato do arquivo, que deve ser *.xls*, *.xlsx* ou *.xlsm*. A ferramenta possui modelos de *templates* para determinados tipos de tabelas de regras, exigindo que seus padrões sejam seguidos para que a tabela seja corretamente interpretada.

O principal modelo de tabela do OpenL Tablets é conhecido como tabela de decisão, sobre o qual pode ser visto um exemplo simples na Figura 9. Tabelas de decisão podem ter seus métodos – decisões – acessados via programa Java pela API do OpenL Tablets, o que será explicado posteriormente.

15 A LGPL é uma licença de software gratuito de autoria da Free Software Foundation, que permite que desenvolvedores utilizem softwares sobre esta licença dentro de seus próprios softwares e ambientes de desenvolvimento, mesmo que estes sejam proprietários, não exigindo a liberação de seu código-fonte.

Figura 9 - Exemplo de tabela de decisão, criado com o LibreOffice Calc, para o OpenL Tablets

	A	B	C
1	Rules String BoasVindas(int horaAtual)		
2	C1	C2	RET1
3	*horaInicial <= horaAtual*	*horaAtual <= horaFinal*	*mensagem + ", mundo!"*
4	int horaInicial	int horaFinal	String mensagem
5	Inicial	Final	Mensagem de boas-vindas
6	0	11	Bom dia
7	12	17	Boa tarde
8	18	23	Boa noite

Fonte: elaborado pelo autor.

Para a criação de tabelas de decisão, é necessário seguir um padrão de elementos. O primeiro deles, apresentado na linha 1 da Figura 9, é o cabeçalho da tabela e deve ser formado pelo prefixo *Rules* (regras) seguido de uma assinatura de método[16] Java. A assinatura indica como o método poderá ser utilizado dentro de um programa Java, o que será visto posteriormente.

A linha 2 da Figura 9 é um cabeçalho que indica o objetivo da coluna: colunas de condições, identificadas pela letra C seguida de um número, indicam que a coluna contém regras condicionais para execução do procedimento; colunas de ações, identificadas pela letra A seguida de um número – não está presente no exemplo, indicam que a coluna possui regras de ação, tais quais mostrar ou setar algum valor; colunas de retorno, identificadas pelo prefixo RET seguido de um número, indicam que a coluna possui valores a serem retornados para o programa que executou a regra.

A linha 3 possui células com a definição do código que será executado conforme o tipo da coluna. Nas colunas A e B, por exemplo, há códigos indicando as condições a serem atendidas pela regra. Já na coluna C, há o código referente ao retorno – no caso, a concatenação do retorno obtido conforme o atendimento das regras (*String mensagem*) com o texto *", mundo!"*.

A linha 4 possui células de definição de parâmetros, sendo estes utilizados na linha superior para verificação da condição ou especificação do retorno. A linha 5 contém apenas uma descrição sobre o que é a coluna, enquanto as últimas três linhas indicam os valores aplicados aos parâmetros. Se a regra do exemplo fosse utilizada com o parâmetro de entrada 15, a saída seria *"Boa tarde, mundo!"*. Outros tipos de tabelas relevantes, tais quais as de tipos, de dados e de testes, serão explicadas posteriormente, quando da apresentação dos

16 Uma assinatura de método, em Java, contém os seguintes elementos: tipo de retorno, nome do método e parâmetros de entrada.

resultados e comparação das ferramentas.

Um importante conceito do OpenL Tablets são as classes empacotadoras (*wrappers*) que expõem as tabelas de decisão como métodos Java através de sua API. Para que isso seja possível, é necessária a existência de uma interface que possua as mesmas assinaturas de método de determinada tabela de decisão. A Listagem 9 apresenta uma classe empacotadora baseada na tabela de decisão da Figura 9.

Listagem 9 - Exemplo de classe empacotadora para tabela de decisão do OpenL Tablets

```
public interface Wrapper {
    String boasVindas( int horaAtual );
}
```

Fonte: elaborado pelo autor

A utilização de uma classe empacotadora, que dispara as regras presentes numa tabela de decisão, é bastante simples e pode ser vista na Listagem 10. É importante ressaltar que, neste caso, a planilha se chama Rules.xls e está na raiz do dispositivo de armazenamento. O trecho de código realiza as seguintes operações: (1) cria uma instância do motor de regras baseado no arquivo Rules.xls, indicando que o mesmo poderá ser convertido para a classe Wrapper; (2) cria o objeto Wrapper a partir do motor de regras e executa o método *boasVindas* informando o valor 15 como parâmetro, armazenando seu resultado para posterior (3) exibição. Neste exemplo, o resultado seria: *"Boa tarde, mundo!"*.

Listagem 10 - Execução de tabela de decisão com classe empacotadora pela API do OpenL Tablets

```
import org.openl.rules.runtime.RuleEngineFactory;

public class WrapperTest {
    public static void main( String[] args ) {
        // (1)
        RuleEngineFactory< Wrapper > rulesFactory = new RuleEngineFactory<
Wrapper >( "/Rules.xls", Wrapper.class );
        // (2)
        Wrapper wrapper = rulesFactory.makeInstance();
        String result = wrapper.boasVindas( 15 );
        // (3)
        System.out.println( result );
    }
}
```

Fonte: elaborado pelo autor

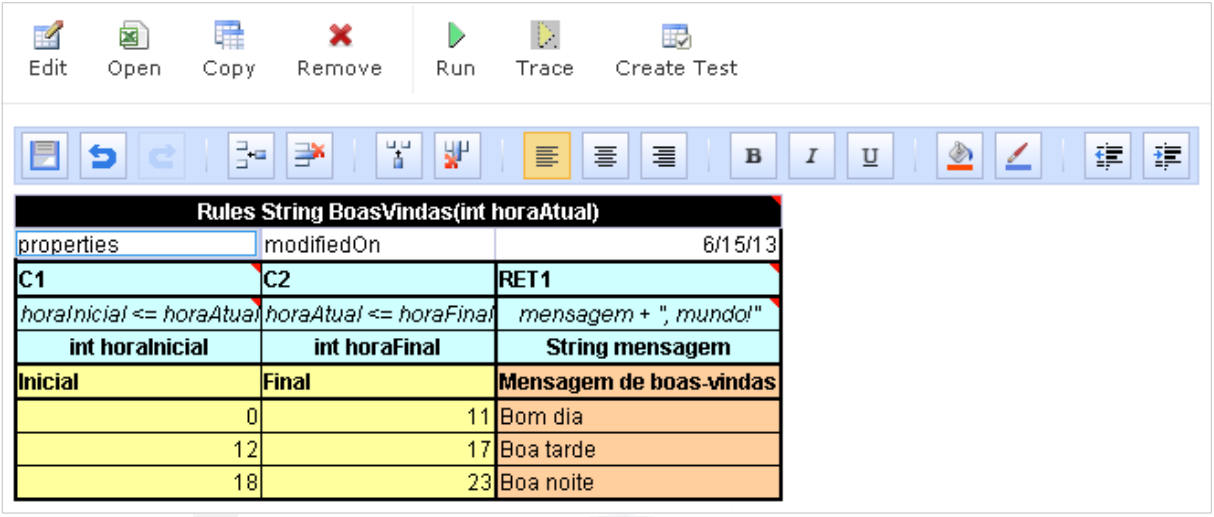
3.2.2 Web Studio

O Web Studio é uma aplicação *web* que permite que os usuários visualizem, editem e gerenciem regras de negócio e projetos de regras criados com a tecnologia OpenL Tablets. A aplicação possui um editor de planilhas que, embora bastante simplificado e com uma gama de ferramentas baixíssima em relação a um editor de planilhas como o Microsoft Excel, por exemplo, atende às necessidades básicas para criação de regras de acordo com seus *templates*. Além disso, a utilização do Web Studio possibilita maior controle das regras e organização das planilhas em diferentes projetos.

O Web Studio também provê um editor passo a passo para a criação de regras, dispõe de controle de versões – gerando uma nova revisão a cada vez que um usuário salva uma regra, gráfico de dependências entre planilhas, execução e análise de testes. Diferentemente das regras criadas em planilhas e acessadas via classes empacotadoras pela API do projeto, as regras criadas com o Web Studio devem ser expostas como WSs para que sejam utilizadas pelas partes interessadas.

A Figura 10 apresenta a visão geral do editor de regras do Web Studio. No exemplo, foi criada a mesma regra de boas vindas apresentada anteriormente na Figura 9. Pode-se notar que há apenas uma diferença significativa, que é a existência de uma linha cuja primeira célula contém o valor “propriedades” (*properties*). Esta é uma linha especial – pode haver mais de uma – que pode conter valores como a descrição e categoria da regra, autor, responsável, entre outras informações em nível de documentação. Na ilustração mencionada, é exibido o atributo “modificado em” (*modifiedOn*), automaticamente adicionado e atualizado pelo editor quando a regra é salva.

Figura 10 - Edição de regra com o OpenL Tablets Web Studio

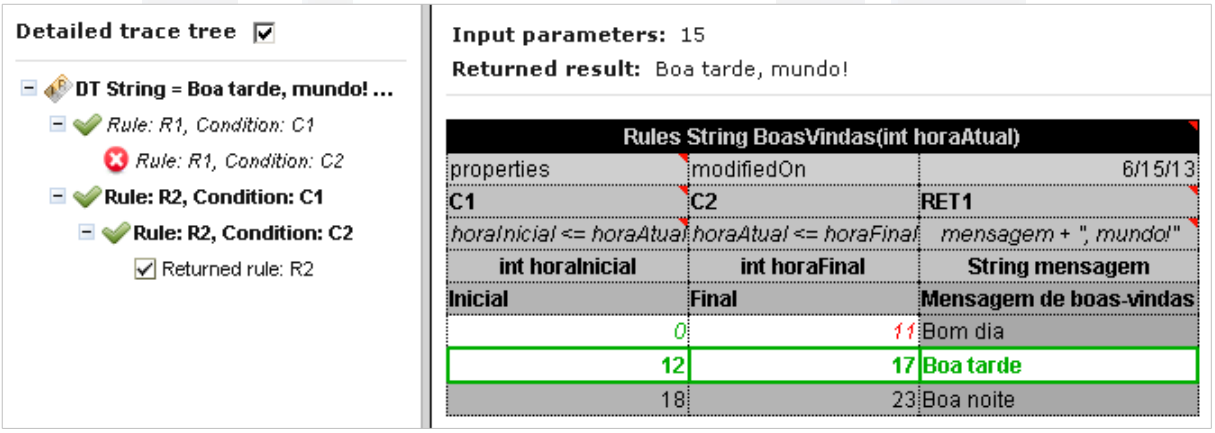


Rules String BoasVindas(int horaAtual)		
properties	modifiedOn	6/15/13
C1	C2	RET1
horaInicial <= horaAtual	*horaAtual <= horaFinal*	*mensagem + ", mundo!"*
int horaInicial	int horaFinal	String mensagem
Inicial	Final	Mensagem de boas-vindas
0	11	Bom dia
12	17	Boa tarde
18	23	Boa noite

Fonte: elaborado pelo autor.

A Figura 11, por sua vez, apresenta a tela de análise de execução de uma tabela de decisão. No caso em questão, é exibido em detalhes como o mecanismo do OpenL Tablets chegou ao resultado (*returned result*) *"Boa tarde, mundo!"*, para a execução da tabela citada anteriormente, recebendo, como parâmetro de entrada (*input parameters*), o valor quinze.

Figura 11 - Análise de um teste de tabela de decisão com o OpenL Tablets Web Studio



Rules String BoasVindas(int horaAtual)		
properties	modifiedOn	6/15/13
C1	C2	RET1
horaInicial <= horaAtual	*horaAtual <= horaFinal*	*mensagem + ", mundo!"*
int horaInicial	int horaFinal	String mensagem
Inicial	Final	Mensagem de boas-vindas
0	*11*	Bom dia
12	17	Boa tarde
18	23	Boa noite

Fonte: elaborado pelo autor.

4 ESTUDO DE CASO

Neste capítulo, são expostos a proposta e o estudo de caso deste trabalho, demonstrando os conceitos relevantes à definição do caso de testes e apresentação dos resultados obtidos a partir da comparação das ferramentas JBoss Drools e OpenL Tablets, assim como os critérios de avaliação e a implementação das regras em cada uma das soluções.

4.1 Proposta de trabalho

O presente trabalho analisou duas ferramentas para gerenciamento das regras de negócio de uma organização, no que se refere a sua utilização dentro de SIs. Com base nas características essenciais e aspectos necessários para tratamento de regras por um BRMS, o trabalho objetivou identificar o nível de preparação das ferramentas em relação às necessidades da área empresarial de uma organização. Além disso, buscou verificar como elas proveem a reutilização das regras de negócio, permitindo sua separação do código-fonte e consequente facilidade de manutenção, algo atualmente essencial para permitir que o software corresponda às expectativas do negócio.

O processo de análise das ferramentas foi feito conforme exemplos, explicados e especificados a seguir, iguais em ambas as plataformas. Para suportar os casos de teste, a análise foi feita sobre um conjunto básico de regras necessário para a implantação e manutenção de um hipotético sistema de comércio eletrônico (*e-commerce*), o qual costuma sofrer constantes atualizações em suas regras, seja devido a trâmites fiscais/contábeis ou, simplesmente, novas promoções ou restrições sobre a venda de produtos.

Para Venetianer (2000), o comércio eletrônico pode ser caracterizado como uma transação comercial realizada entre a empresa e seus clientes através da internet. Turban e King (2004) afirmam que um sistema de *e-commerce* proporciona uma série de benefícios para as empresas, dentre as quais: aceleração da expansão no mercado, maior abrangência do negócio e redução do intervalo de tempo entre a concepção de uma ideia e sua execução – devido ao ambiente dinâmico proporcionado pela internet. Para que tudo isso seja possível, e atendendo a um requisito básico para um *e-commerce* de qualidade – a possibilidade de mudanças constantes, é essencial que a área de negócios da empresa tenha o controle do sistema e consiga, com facilidade, gerenciar as regras que o guiam.

Visando apoiar a definição dos casos de testes que foram aplicados às ferramentas,

este trabalho utilizou alguns dos conceitos básicos referentes à engenharia de software: a elicitação das regras de negócio e um diagrama de classes.

4.1.1 Regras de negócio

Para a análise e estudo das ferramentas BRMS deste trabalho, o Quadro 1 apresenta um conjunto de regras de negócio básico mapeado de forma a atender às demandas de um sistema de *e-commerce*. Cada regra possui um identificador formado da seguinte forma: prefixo RN, letra identificadora da seção da regra e um número sequencial dentro da seção. É importante a percepção de que estão listadas regras de negócio envolvidas com todo o processo por trás de um sistema de comércio eletrônico, desde o cadastro de clientes até a liberação e envio de pedidos.

Quadro 1 - Regras de negócio do sistema de *e-commerce*

Regra	Descrição
***Categoria:* Clientes**	
RN-C1	Cliente especial é o que possui pelo menos 1.000 pontos acumulados.
***Categoria:* Descontos**	
RN-D1	De acordo com a pontuação (inclusive), conceder percentual de desconto sobre o valor da venda se a operação for realizada por um cliente especial: • De 1.000 a 1.999: 5% • De 2.000 a 4.999: 7,5% • A partir de 5.000: 10,0%
RN-D2	Conceder desconto de 25% para produtos em liquidação.
RN-D3	Conceder desconto de 5% se o pagamento for realizado por boleto bancário.
***Categoria:* Parcelamento / Juros**	
RN-PJ1	O número máximo de parcelas é 12.
RN-PJ2	O valor mínimo de uma parcela, quando houver mais de uma, é R$ 10,00.
RN-PJ3	Os juros devem ser calculados de acordo com a quantidade de parcelas, conforme (inclusive): • 2 a 4: 1,0% • 5 a 8: 2,0% • 9 a 12: 3,5%
***Categoria:* Vendas**	
RN-V1	Só pode ser feita a venda de produtos que estejam disponíveis em estoque.
RN-V2	O valor final da venda deve ser calculado conforme somatório da multiplicação do valor dos itens pela sua quantidade, decrescidos os descontos e aplicados os juros.
RN-V3	Conceder um ponto ao cliente a cada R$ 1,00 em compras.

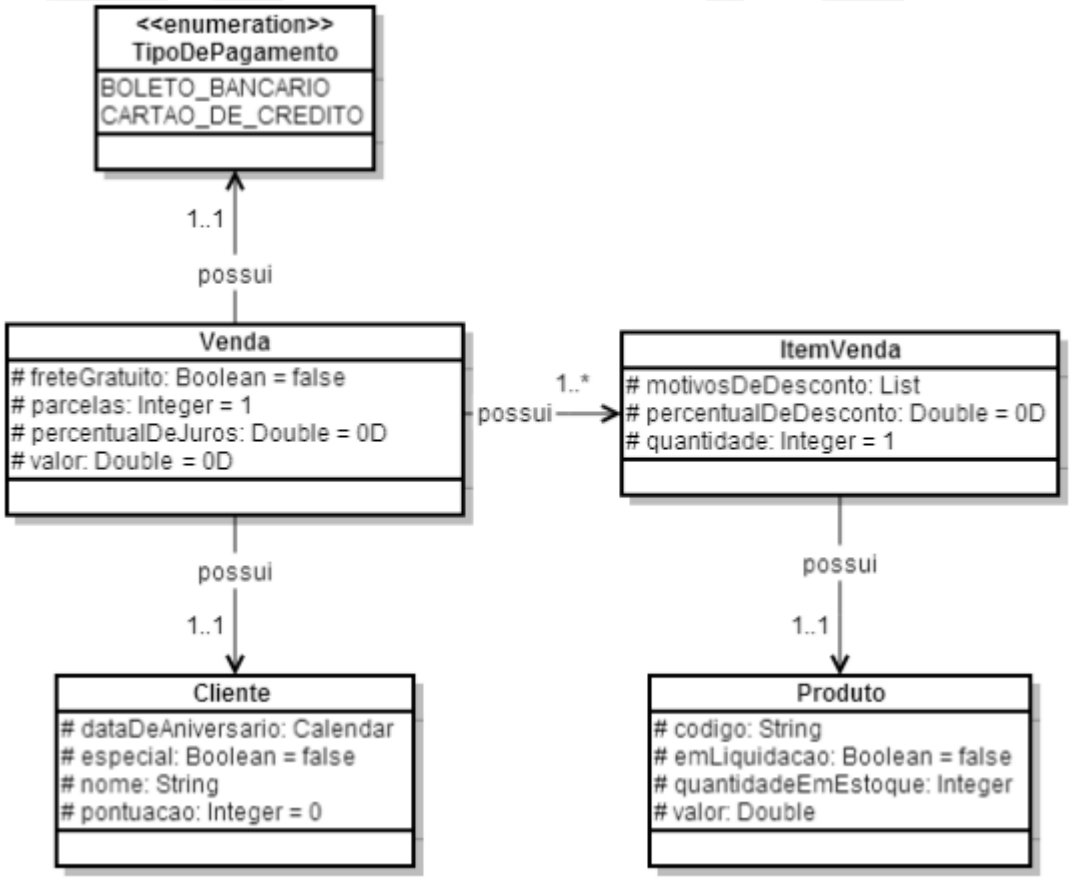
RN-V4	Conceder 100 pontos ao cliente se ele efetuar uma compra no dia do seu aniversário.
RN-V5	Não cobrar frete nas seguintes situações: • Valor final da venda é superior a R$ 149,00; • Valor final da venda é superior a R$ 99,00 e o tipo de pagamento é boleto bancário.

Fonte: elaborado pelo autor.

4.1.2 Diagrama de classes

A Figura 12 apresenta um diagrama de classes[17] contendo as classes necessárias para o estudo de caso. Pode-se perceber a existência da classe Venda, a qual possui relacionamento com um único Cliente e com ao menos um ItemVenda. Este último, por sua vez, possui uma associação com um único Produto. Por último, há um *enum* TipoDePagamento, vinculado a Venda, com valores predefinidos para os tipos de venda aceitos.

Figura 12 - Diagrama de classes para as classes do estudo de caso



Fonte: elaborado pelo autor.

17 Um diagrama de classes contém toda a estrutura de classes utilizada pelo sistema, especificando os atributos e métodos de cada uma, bem como seu relacionamento e associação geral (GUEDES, 2009).

4.2 Critérios de avaliação

Para realizar os processos de teste, análise e comparação das ferramentas, foram definidos critérios apropriados para tal. A Tabela 10 demonstra os tópicos utilizados e contém um resumo sobre cada um deles.

Tabela 10 - Critérios utilizados na avaliação e comparação das ferramentas

Critério	Descrição
Interface web	Verificar o funcionamento geral e as possibilidades disponíveis na interface *web* da ferramenta.
Formatos de regra	Pesquisar as opções de criação de regras fornecidas pela solução.
Testes	Analisar cenários de testes de regras disponibilizados pela solução.
Segurança	Averiguar questões referentes a *login* e controle de permissões sobre as regras.
Auditoria e versionamento	Analisar como as ferramentas lidam com as questões sobre quem, quando e o quê foi alterado no repositório, bem como sobre a criação de versões para cada alteração efetuada em alguma regra.
Integração externa	Explorar os formatos de integração da solução com sistemas externos.
Repositório	Verificar o modelo de repositório utilizado pela ferramenta.

Fonte: elaborado pelo autor.

4.3 Avaliação e comparação das ferramentas

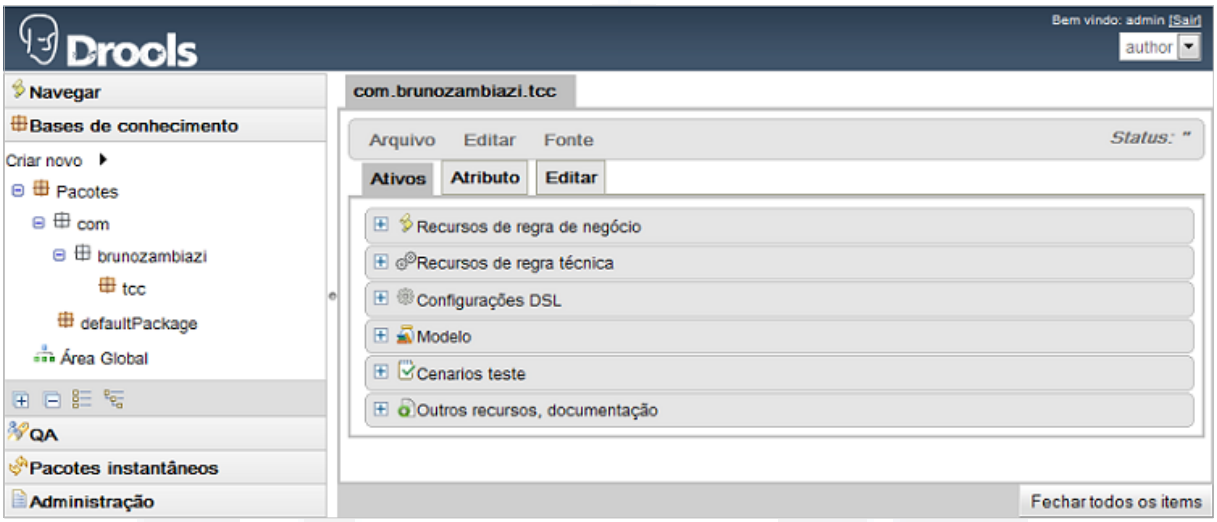
Com base nos tópicos e critérios definidos para avaliação e comparação das ferramentas JBoss Drools e OpenL Tablets (TABELA 10), as regras de negócio apresentadas no Quadro 1, no item 4.1.1 deste capítulo, foram implementadas nos dois softwares, utilizando, em ambos, suas ferramentas *web* – Guvnor e Web Studio, respectivamente. Os resultados e impressões obtidos são descritos nos tópicos a seguir.

4.3.1 Interface *web*

As ferramentas alvo de estudo deste trabalho se propõem a possibilitar que usuários do negócio gerenciem as regras de negócio de um SI, sem que, para isso, seja necessária a intervenção de usuários da TI. Para suprir esta necessidade, tanto o JBoss Drools quanto o OpenL Tablets dispõem uma interface *web* que as caracterizam, de fato, como sistemas de BRMS. Seus nomes são, respectivamente, Guvnor e Web Studio, ambas explicadas brevemente no Capítulo 3, respectivamente nos itens 3.1.2 e 3.2.2.

O Guvnor trabalha com o conceito de “bases de conhecimento” para realizar a separação do conteúdo do repositório. Dentro da base de conhecimento, separados em pacotes, são criados os ativos – regras de negócio, funções, modelos de dados, cenários de testes e arquivos gerais, como documentações. A Figura 13 ilustra a tela de visualização dos pacotes de uma base de conhecimento – a utilizada neste trabalho é *com.brunozambiazi.tcc*, à esquerda, bem como a categoria dos ativos existentes neste pacote, à direita.

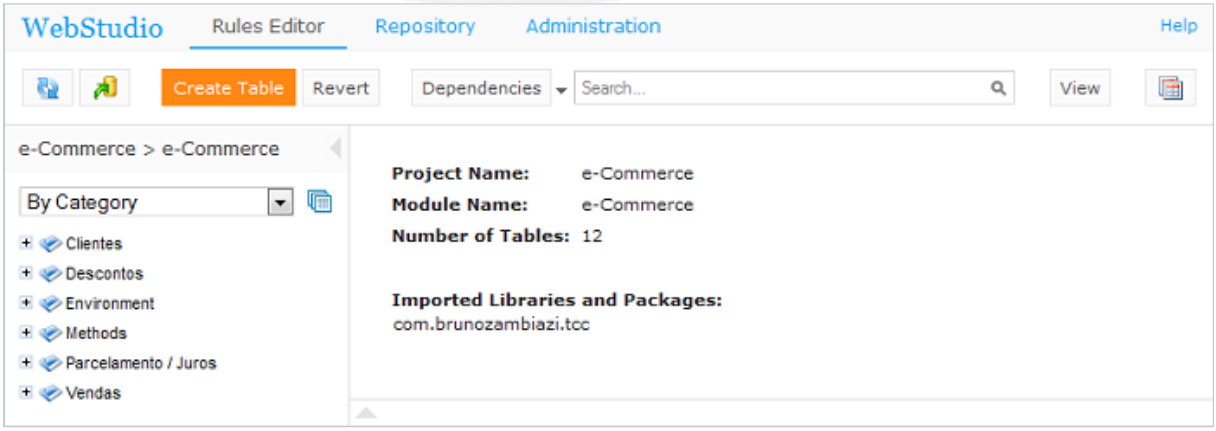
Figura 13 - Visão geral de uma base de conhecimento do Guvnor



Fonte: elaborado pelo autor.

O Web Studio trabalha com o conceito de “projetos de regras”, no qual são criados todos os tipos de recursos permitidos pelo OpenL Tablets – tabelas, métodos e modelos de dados. A Figura 14 ilustra a visualização inicial de um projeto – denominado *e-commerce*, agrupando seus recursos em categorias, à esquerda, e exibindo um resumo dele, à direita.

Figura 14 - Visão geral de um projeto do Web Studio



Fonte: elaborado pelo autor.

Uma das principais desvantagens do Web Studio, em relação à interface *web*, é o fato da ferramenta não possuir versão em português, o que a torna ainda mais difícil de ser utilizada levando em consideração o contexto de empresas brasileiras – o Guvnor possui uma versão em português que, embora contenha alguns erros de tradução, facilita seu entendimento. Além disso, o Web Studio não dispõe de funcionalidades para criação de documentação referente às regras, o que acaba sendo uma desvantagem em relação ao Guvnor – que permite documentação em praticamente todos seus recursos – uma vez que descrições textuais e explícitas sobre determinada regra são essenciais para a manutenção da mesma no futuro.

De forma geral, as duas ferramentas não são de fácil utilização. Além de não possuírem uma interface clara e amigável para indicar as ações do usuário, ambas exigem um tempo considerável de aprendizado para que se consiga realizar tarefas relativamente simples, como, por exemplo, a criação de uma regra e seu teste de aceitação.

4.3.2 Formatos de regra

Cada BRMS trabalha com conceitos próprios para a execução de suas regras de negócio. O OpenL Tablets, por exemplo, é conhecido como um processador avançado de tabelas que captura determinados padrões em planilhas eletrônicas, tornando acessíveis como regras as tabelas encontradas, e exigindo, assim, que elas sejam chamadas explicitamente para serem executadas. Ferramentas como o JBoss Drools, por outro lado, possuem um motor de regras e inferência que descobre, dado um conjunto de fatos, quais regras devem ser executadas e em que ordem, baseando-se em aprimoramentos do algoritmo Rete[18].

Além disso, cada BRMS também possui formatos específicos para a criação de regras de negócio. Os formatos suportados pelas ferramentas de estudo deste trabalho foram brevemente explicados no Capítulo 3, e as regras de negócio relacionadas ao hipotético sistema de comércio eletrônico, expostas no Quadro 1, no item 4.1.1 do Capítulo 4, foram criadas em ambas as soluções através de suas interfaces *web*.

O fato do OpenL Tablets suportar apenas tabelas de decisão como modelo e formato

18 Rete é um algoritmo largamente utilizado em sistemas de regras devido a sua eficiência em determinar o que – e em qual ordem – deve ser executado, a partir de um conjunto de regras e fatos. De forma geral, sistemas de BRMS implementam este algoritmo com melhorias que julgam adequadas e, de fato, são estas melhorias que fazem com que uma ferramenta tenha vantagens de desempenho em relação às outras. Conforme Forgy (1982), "o algoritmo Rete é um método eficiente para comparação de uma grande coleção de padrões com uma grande coleção de objetos. Ele encontra todos os objetos que correspondem com cada padrão."

de regras, faz com que o processo de criação tenha de ser analisado e implementado de forma diferente. Isto porque as tabelas de decisão permitem expressar conjuntos de condições e de ações em um único elemento – a própria tabela. A implementação no Drools, ao contrário, exige que cada condição sobre o mesmo objeto seja feita em uma nova regra, devido ao fato da ferramenta utilizar o modelo definido como ECA[19] – explicado no Capítulo 2, item 2.4.2.2.

Um exemplo dessa diferença é a implementação da regra de negócio RN-PJ3, que define o percentual de juros que deve ser aplicado a uma venda conforme o número de parcelas. Sua criação no formato de tabela de decisão permite que todas as condições e ações sejam expressas numa única tabela, conforme mostra a Figura 15, enquanto no modelo ECA ela precisaria ser dividida em três regras semelhantes à apresentada na Figura 16, mudando apenas os valores correspondentes.

Figura 15 - Implementação da regra RN-PJ3 como tabela de decisão do OpenL Tablets

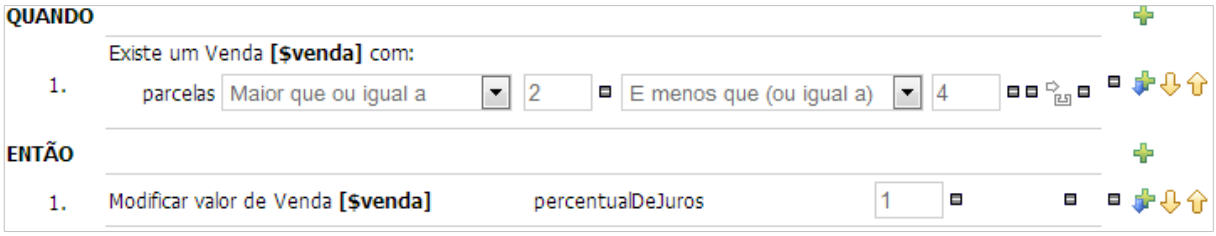
	B	C
37	Rules Boolean calcularJuros(Venda venda)	
38	*name*	Calcular o percentual de juros da venda.
39	*category*	**Parcelamento / Juros**
40	**C1**	**A1**
41	venda.parcelas	venda.percentualDeJuros = juros
42	*IntRange*	*Double juros*
43	**Número de parcelas:**	**% de juros:**
44	2-4	1
45	5-8	2
46	9-12	3,5

Fonte: elaborado pelo autor.

A Figura 16 apresenta a regra implementada com o editor guiado do Guvnor. Nele, definem-se todas as condições – "quando" – que devem ser satisfeitas para que a ação da regra – "então" – seja disparada. No exemplo, verifica-se a existência de um objeto do tipo *Venda* – o qual é denominado *$venda* – com o atributo *parcelas* maior ou igual a dois e menor ou igual a quatro. Sendo satisfeita a regra, o mesmo objeto terá o atributo *percentualDeJuros* modificado, passando a receber o valor um.

19 É importante citar que o Drools também possui suporte completo a tabelas de decisão. No entanto, o objetivo deste trabalho é analisar diferentes perspectivas quanto à criação e manutenção de regras de negócio, e, por isso, optou-se pela sua avaliação apenas no que se refere ao seu grande diferencial: a linguagem própria e extensível para criação de regras e o editor guiado de seu módulo de BRMS.

Figura 16 - Implementação de parte da regra RN-PJ3 no formato ECA, utilizando o editor guiado do Guvnor

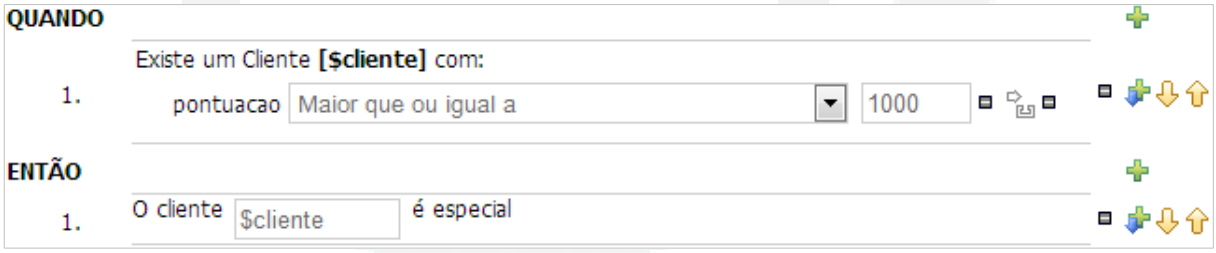


Fonte: elaborado pelo autor.

Um dos grandes diferenciais do Drools, além do editor guiado, é o conceito de DSL. Conforme explicado no Capítulo 3, item 3.1.1, DSL é definido por Browne (2009) como um recurso que visa facilitar o trabalho de usuários do negócio, permitindo que eles escrevam as regras em um formato similar à linguagem humana. De forma geral, DSL pode ser visto como uma espécie de dicionário de associação, visto que serve para vincular uma frase qualquer com uma sentença em linguagem nativa do Drools, o DRL.

Um exemplo desse recurso é a implementação da regra RN-C1, que define a pontuação mínima para que um cliente seja considerado especial. Conforme pode ser visto na Figura 17, a regra possui uma condição padrão[20] e uma ação personalizada que utiliza uma sentença DSL – armazenada de acordo com a Listagem 11.

Figura 17 - Implementação da regra RN-C1 no Guvnor, utilizando sentença DSL



Fonte: elaborado pelo autor.

Listagem 11 - DSL com a sentença da regra RN-C1

```
[then]O cliente {cliente} é especial=
    {cliente}.setEspecial( true );
```

Fonte: elaborado pelo autor.

Embora mais limitado neste contexto, o OpenL Tablets também possui algumas expressões que podem ser utilizadas para facilitar o trabalho de quem escreve as regras. Valores booleanos, por exemplo, podem ser substituídos pelas palavras *verdadeiro* e *falso*,

20 Frases como "Existe um", "Não existe", "Modificar valor de" e "Inserir valor", são padrão do Guvnor e estão disponíveis para qualquer regra criada com o editor guiado, independentemente de haver um DSL.

enquanto sinais de menor, menor ou igual, maior e maior ou igual, por frases – em inglês – como: *less than <number>*, *more than <number>*, *<number> and more*. A Figura 18 demonstra a implementação da regra RN-C1 no OpenL Tablets, utilizando as frases nas condições da pontuação do cliente, e os termos *falso* e *verdadeiro* para serem aplicados a um atributo booleano do objeto *Cliente*.

Figura 18 - Implementação da regra RN-C1 como tabela de decisão do OpenL Tablets

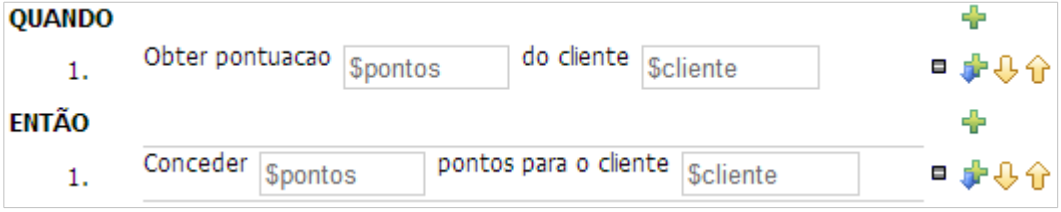
	B	C
2	Rules void clienteEspecial(Cliente cliente)	
3	*name*	Indicar se o cliente é especial.
4	*category*	Clientes
5	C1	A1
6	cliente.pontuacao	cliente.especial = especial
7	*IntRange*	*Boolean especial*
8	Pontuação:	O cliente é especial?
9	less than 1000	FALSO
10	1000 and more	VERDADEIRO

Fonte: elaborado pelo autor.

A maior dificuldade na utilização de ambas as ferramentas ocorre quando se torna necessário efetuar mais operações aritméticas e cálculos em geral. Um exemplo disso é a regra RN-V3, que especifica que o cliente deve receber um ponto para cada R$ 1,00 em compras. O problema, neste caso, é a necessidade de realizar um arredondamento no resultado da divisão do valor total da compra pelo número de pontos, que poderia, por causa dos centavos, retornar um valor decimal – sendo que o atributo que armazena os pontos do cliente é um inteiro.

No Guvnor, a forma apropriada para a criação desta regra é a utilização de sentenças DSL que deixam-na mais fácil de ser entendida, abstraindo de sua implementação toda a parte do cálculo e do arredondamento. A Figura 19 demonstra como fica a regra utilizando o DSL da Listagem 12.

Figura 19 - Implementação da regra RN-V3 no Guvnor, utilizando sentenças DSL



Fonte: elaborado pelo autor.

Listagem 12 - DSL da regra RN-V3

```
[when]Obter pontuacao {pontuacao} do cliente {cliente}=
   Venda( {cliente} : cliente != null, {pontuacao} : Math.floor( valor/1 ) )

[then]Conceder {pontos} pontos para o cliente {cliente}=
   {cliente}.setPontuacao( {cliente}.getPontuacao()+{pontos} );
```

Fonte: elaborado pelo autor.

A mesma regra implementada no OpenL Tablets poderia ser feito conforme a Figura 20. Nela, a complexidade do cálculo e arredondamento continuam presentes na própria regra, sendo necessário, inclusive, realizar a conversão explícita do resultado para o tipo adequado.

Figura 20 - Implementação da regra RN-V3 como tabela de decisão do OpenL Tablets

	B	C
82	Rules Boolean pontuarClienteAposUmaVenda(Venda venda)	
83	*name*	Conceder pontos ao cliente conforme o valor da sua compra.
84	*category*	**Vendas**
85	**C1**	**A1**
86	venda != null && venda.cliente != null	venda.cliente.pontuacao += (int)Math.floor(venda.valor / pontos)
87	*Boolean*	*Integer pontos*
88	**Venda e cliente:**	**Pontos concedidos por cada R$ em compras:**
89	VERDADEIRO	1

Fonte: elaborado pelo autor.

Outra amostra de dificuldade se refere a regras nas quais é necessário um maior número de operações, o que as torna, consequentemente, mais complexas. Das regras de negócio citadas para o sistema de comércio eletrônico deste trabalho, a de maior complexidade é a RN-V2, que indica o cálculo do valor final da venda, consistindo no somatório do valor dos itens e aplicação de descontos e juros.

No Guvnor, não foi possível criar a regra através do editor guiado. A única forma encontrada para expressá-la foi através do conceito de regra técnica, recurso que o BRMS disponibiliza para a criação de regras de forma livre. A implementação dessa regra pode ser vista na Listagem 13.

Listagem 13 - Implementação da regra RN-V2 como uma regra técnica, no Guvnor

```
package com.brunozambiazi.tcc

rule "Calcular valor final da venda"
   when
      $venda : Venda()
      $valor : Double() from accumulate(
         $i : ItemVenda( venda == $venda, $p : produto )
       , sum( $p.getValor()
            * $i.getQuantidade()
            * ( 1 - $i.getPercentualDeDesconto()/100 )
```

```
            )
        )
    then
        $venda.setValor( $valor
                        * ( 1 + $venda.getPercentualDeJuros()/100 )
                        );
end
```

Fonte: elaborado pelo autor.

No OpenL Tablets, a complexidade da regra existe não apenas pelas operações que ela exige, mas também pela necessidade de iterar sobre os itens da venda. Assim, a criação da regra demandou a utilização de dois conceitos diferentes que a ferramenta disponibiliza para os casos mais complexos: tabela de método (*method table*) e tabela de planilha (s*preadsheet table*).

Enquanto as tabelas de método consistem na escrita de códigos Java interpretados pela API do OpenL Tablets em tempo de execução, as tabelas de planilha possuem fórmulas e cálculos separados em linhas e colunas. Para a regra RN-V2, foi necessário criar uma tabela de método que realiza a iteração sobre os itens da venda, chamando, para cada um, a tabela de planilha que calcula o valor referente àquele item.

A implementação e complexidade das duas tabelas pode se vista nas Figuras 21 e 22. Na primeira, é necessário utilizar código Java para conseguir percorrer todos os itens da venda, fazendo seu somatório através da chamada à tabela apresentada na segunda figura, que, por sua vez, funciona em um modelo de declaração de variáveis – coluna esquerda – com valores calculados – coluna direita.

Figura 21 - Tabela de método para a regra RN-V2, no OpenL Tablets

	A
1	Method void calcularValorFinalDaVenda(Venda venda)
2	for (int x = 0; x < venda.itens.length; x++) {
3	venda.valor += calcularValorDoItem(venda.itens[x]).doubleValue();
4	}

Fonte: elaborado pelo autor.

Figura 22 - Tabela de planilha para a regra RN-V2, no OpenL Tablets

	A	B	C
1	Spreadsheet DoubleValue calcularValorDoItem(ItemVenda item)		
2	*properties*	*name*	Calcular o valor do item numa venda, aplicando descontos.
3		*category*	**Vendas**
4	**Variaveis:**	**Valor:**	
5	Desconto	{item.percentualDeDesconto / 100}	
6	Percentual Desconto	{1 - $Desconto}	
7	Valor Item	{item.produto.valor * item.quantidade * $Percentual Desconto}	
8	Juros	{item.venda.percentualDeJuros / 100}	
9	Percentual Juros	{1 + $Juros}	
10	**RETURN**	**{$Valor Item * $Percentual Juros}**	

Fonte: elaborado pelo autor.

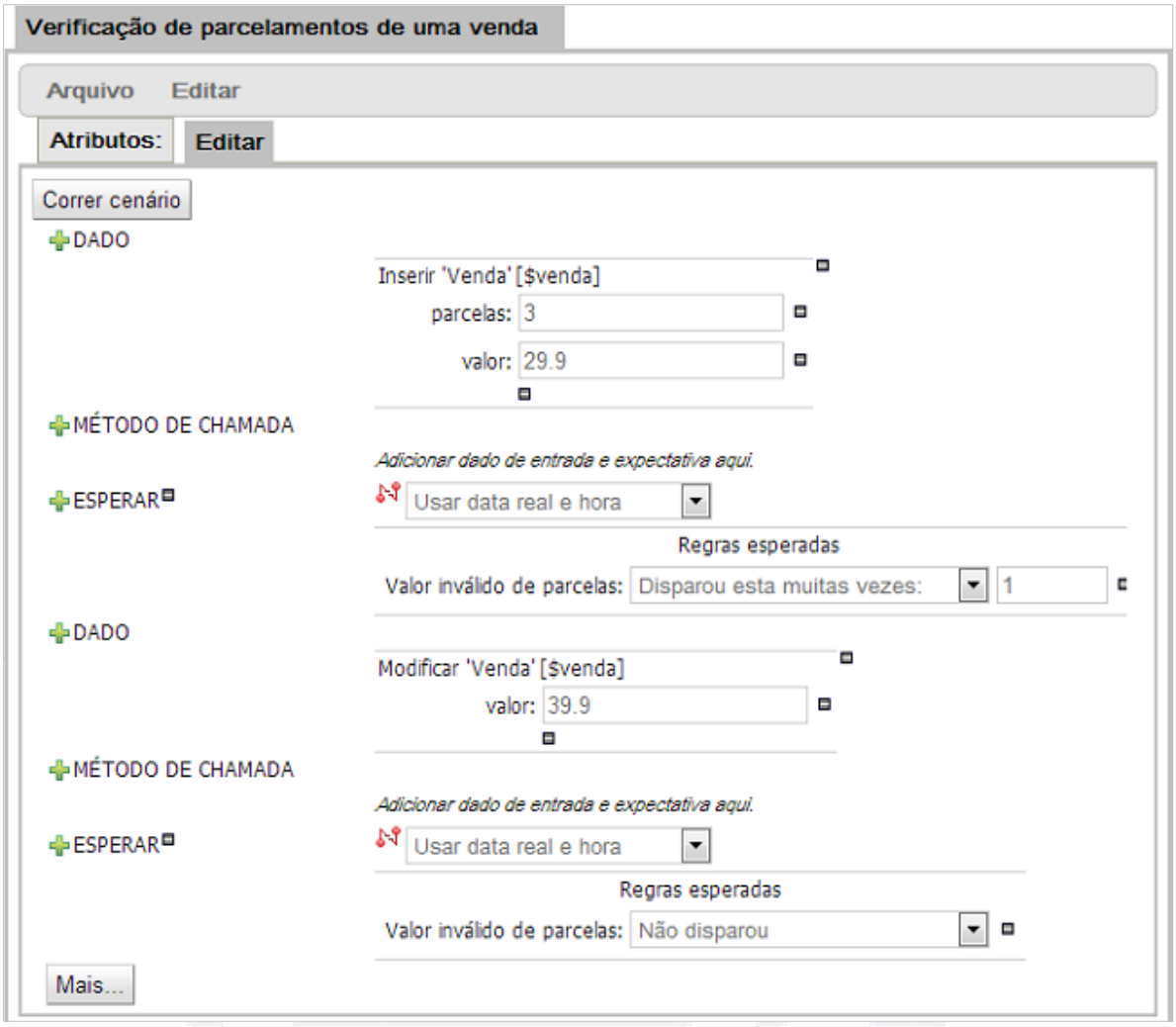
4.3.3 Testes

A possibilidade de testar as regras de negócio antes que elas sejam aplicadas numa base de produção, é um dos requisitos básicos para qualquer solução de BRMS. Tanto o Guvnor quanto o Web Studio possuem ferramentas específicas para isso.

Por parte do Drools, o Guvnor trabalha com o conceito de cenários de testes. Na prática, um cenário nada mais é do que um conjunto de valores de entrada/saída e da informação do que se espera de resultado em termos de regras disparadas – é possível informar, inclusive, quantas vezes se espera que uma regra seja executada. Assim como as regras, os cenários de testes também são separados por pacotes, o que é importante devido ao motor de inferências do Drools que fará a avaliação apenas das regras existentes nos pacotes de mesmo nome.

As regras RN-PJ1 e RN-PJ2 representam validações sobre o número máximo e o valor mínimo das parcelas de uma venda, respectivamente. Sua implementação no Drools, feita em forma de duas regras, pode ser testada com um único cenário de testes que contemple todas as situações relacionadas. A Figura 23 demonstra o cenário de testes para tal, ao mesmo tempo em que deixa evidente a apresentação gráfica confusa da ferramenta, que causa dificuldades na identificação de cada fato referente ao teste.

Figura 23 - Cenário de testes do Guvnor para as regras RN-PJ1 e RN-PJ2



Fonte: elaborado pelo autor.

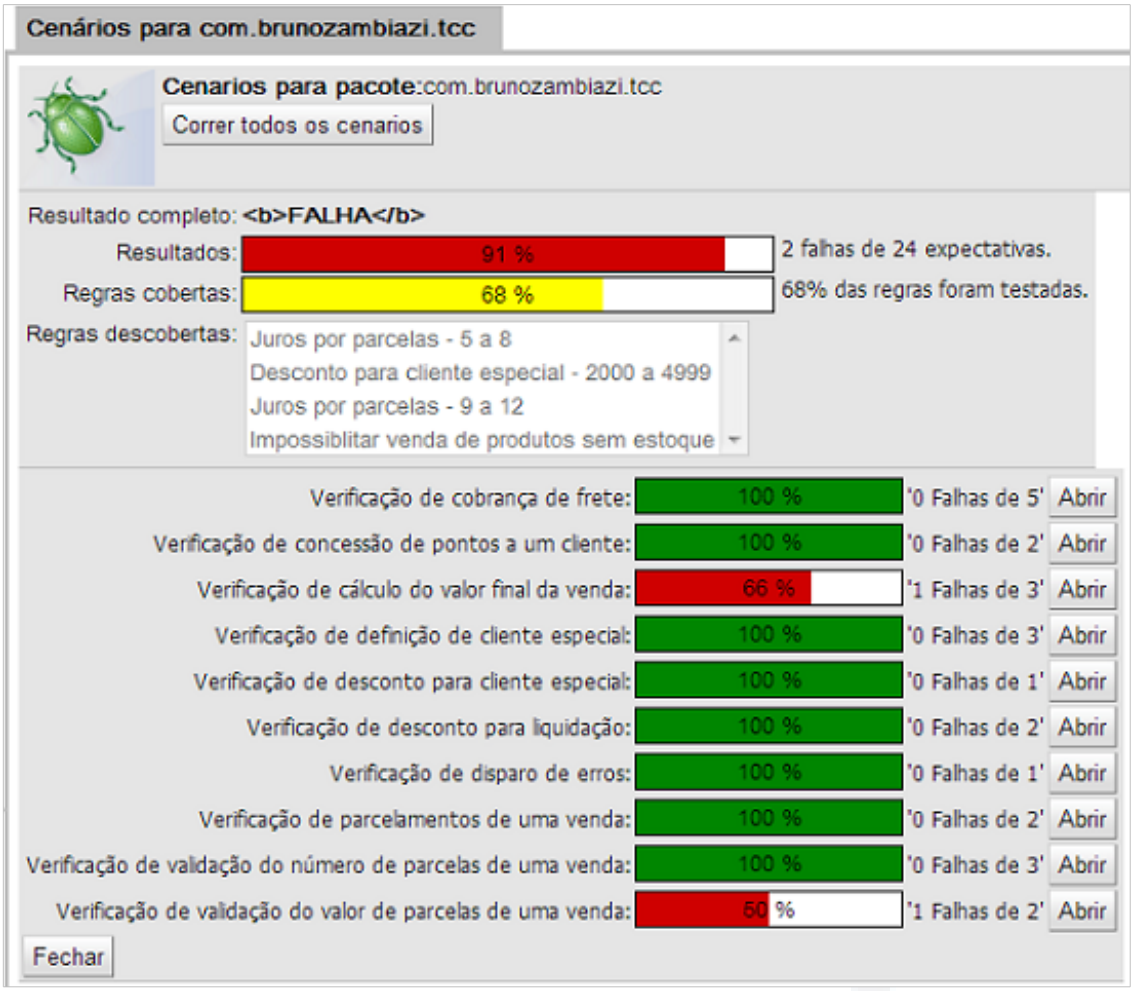
No exemplo demonstrado, há dois testes referentes a uma venda:

1. Venda com três parcelas e valor equivalente a R$ 29,90: espera-se que a regra de validação do valor das parcelas seja disparado uma vez, visto que ela especifica que o valor mínimo de cada parcela é R$ 10,00;
2. Venda com três parcelas e valor equivalente a R$ 39,90: espera-se que não seja disparada a regra de validação do valor das parcelas, visto que cada parcela teria valor superior ao mínimo de R$ 10,00.

Devido à grande quantidade de regras geralmente existentes, o Guvnor possui uma área específica para facilitar a execução de todos os testes de forma simultânea. Para isso, há o recurso conhecido como Análise de Qualidade (FIGURA 24), o qual demonstra o percentual de sucesso dos testes em relação às expectativas sobre os mesmos. Nele, também é possível

verificar o percentual de regras disparadas, bem como as regras que não são executadas por nenhum dos cenários de teste existentes.

Figura 24 - Execução do teste de Análise de Qualidade do Guvnor



Fonte: elaborado pelo autor.

O diferente formato de execução de regras, entre o Drools e o OpenL Tablets, também é refletido na maneira como seus testes são criados e efetuados. Enquanto o primeiro, devido a seu motor de inferências, analisa todas as regras de um mesmo pacote e dispara as que têm suas condições atendidas pelos fatos fornecidos pelos cenários de teste, o OpenL Tablets trabalha com o conceito de testes unitários[21].

Assim como todas as suas regras, os testes do OpenL Tablets também são criados no formato de tabela. Por denotarem testes unitários, as tabelas de teste são atreladas diretamente

21 "Teste unitário é o processo onde blocos individuais, funções, métodos ou 'unidades' de código são testados individualmente. Um conjunto de entrada predeterminado é usado para gerar uma saída que é comparada contra uma saída esperada. […] Testes unitários permitem que você saiba quando seu código quebra." (CASEY, 2010)

à tabela da regra a que representam, possuindo as colunas de entrada de dados e uma específica para o retorno esperado. A Figura 25 representa uma tabela de testes do OpenL Tablets para as regras de negócio RN-PJ1 e RN-PJ2, contendo os mesmos valores de entrada aplicados ao cenário de testes apresentado anteriormente no Guvnor. Nesta figura, pode-se perceber que a linha de cabeçalho contém a palavra-chave "Test", o nome da tabela de decisão das regras, que no caso seria "validarParcelamento" (FIGURA 26), e um nome qualquer representando o próprio teste.

Figura 25 - Tabela de testes do OpenL Tablets para as regras RN-PJ1 e RN-PJ2

	A	B	C
1	Test validarParcelamento validarParcelamentoTest		
2	venda.parcelas	venda.valor	_res_
3	**Parcelas**	**Valor**	**Resultado**
4	3	29.9	FALSE
5	3	39.9	TRUE

Fonte: elaborado pelo autor.

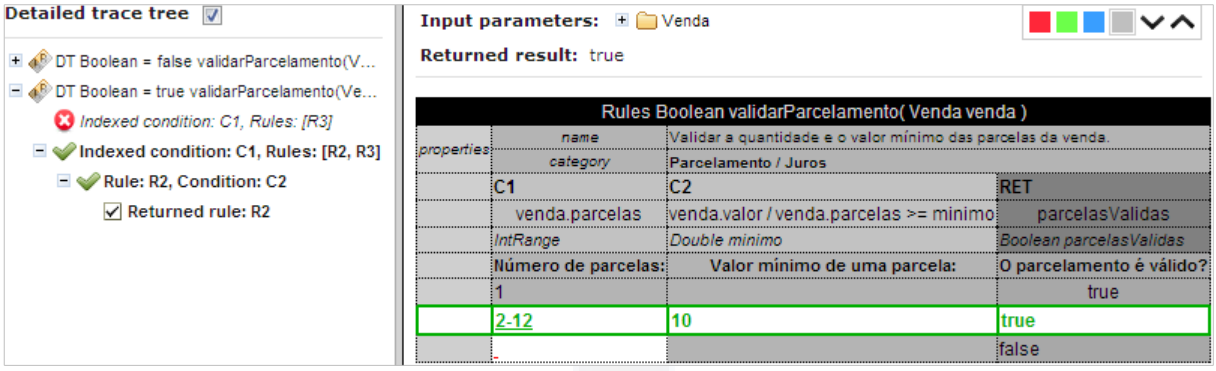
Figura 26 - Implementação das regras RN-PJ1 e RN-PJ2 como tabela de decisão do OpenL Tablets

	B	C	D
48	Rules Boolean validarParcelamento(Venda venda)		
49	*name*	Validar a quantidade e o valor mínimo das parcelas da venda.	
50	*category*	**Parcelamento / Juros**	
51	**C1**	**C2**	**RET**
52	venda.parcelas	venda.valor / venda.parcelas >= minimo	parcelasValidas
53	*IntRange*	*Double minimo*	*Boolean parcelasValidas*
54	**Número de parcelas:**	**Valor mínimo de uma parcela:**	**O parcelamento é válido?**
55	1		VERDADEIRO
56	2-12	10	VERDADEIRO
57			FALSO

Fonte: elaborado pelo autor.

Um recurso interessante do Web Studio é o de possibilitar que a execução de um teste seja rastreada, permitindo que o usuário da ferramenta entenda exatamente o que aconteceu para acarretar em determinado resultado. A Figura 27 demonstra a tela de rastreio da tabela de testes supracitado. Nela, é possível selecionar – do lado esquerdo – qual dos testes se deseja detalhar – visto que a tabela de testes possuía dois casos, visualizando, à direita, o resultado obtido.

Figura 27 - Tela de rastreio do teste das regras RN-PJ1 e RN-PJ2, no Web Studio



Fonte: elaborado pelo autor.

De forma geral, ambas as ferramentas possuem bons recursos para a realização de testes adequados à sua realidade e peculiaridades de execução de regras – embora, principalmente no caso do Drools, questões como a usabilidade da interface poderiam ser melhor exploradas. Algo evidente na comparação das ferramentas, também, é a falta de um recurso, no Drools, que possibilite ao usuário entender por que determinada regra não foi disparada – qual condição não foi atendida, uma vez que seus logs de execução demonstram apenas o rastreio das regras cujas condições foram satisfeitas.

4.3.4 Segurança

O controle de permissões referente ao domínio das regras é um fato importante em um BRMS, uma vez que, dependendo da área de atuação da organização, muitas das regras existentes possuem informações vitais e seu acesso deve ser restrito a poucas pessoas, muitas vezes apenas as do alto escalão – diretores, gerentes, etc.

No momento da criação de uma regra – e de qualquer outro ativo – no Guvnor, o usuário pode associá-la a uma ou mais categorias, e precisa associá-la a um único pacote. Categorias e pacotes são elementos criados previamente por usuários com permissão de administração na ferramenta, e servem para separar os ativos de forma lógica, geralmente por afinidade e/ou funcionalidades a que representam.

Basicamente, o modelo de segurança do Guvnor funciona sobre três pilares: administração, categorias e pacotes. A Tabela 11 indica os tipos de permissão que cada usuário da ferramenta pode receber.

Tabela 11 - Tipos de permissão para usuários do Guvnor

Tipo	Significado
Administrador	É o maior nível de permissão, podendo o usuário realizar qualquer tipo de operação dentro da ferramenta.
Analista	Tipo de usuário com permissão para operar a ferramenta em nível de categorias.
Analista - Leitura	Permissão apenas de leitura nas categorias determinadas.
Pacote - Desenvolvedor	Tipo de usuário com permissão para operar a ferramenta em nível de pacotes.
Pacote - Leitura	Permissão apenas de leitura nos pacotes determinados.

Fonte: elaborado pelo autor.

O Web Studio, embora também permita a separação das regras em projetos e categorias, não fornece maneiras de restringir o acesso especificamente a estes elementos. Seu modelo de segurança consiste na criação de grupos de privilégios referentes a ações como criação, edição, exclusão e execução de regras e projetos, e posterior associação dos usuários a um ou mais destes grupos. Consequentemente, este modelo se torna mais limitado e mais simples do que o aplicado pelo Guvnor, não possibilitando que ocorra o controle de acesso a determinada parte do domínio de regras.

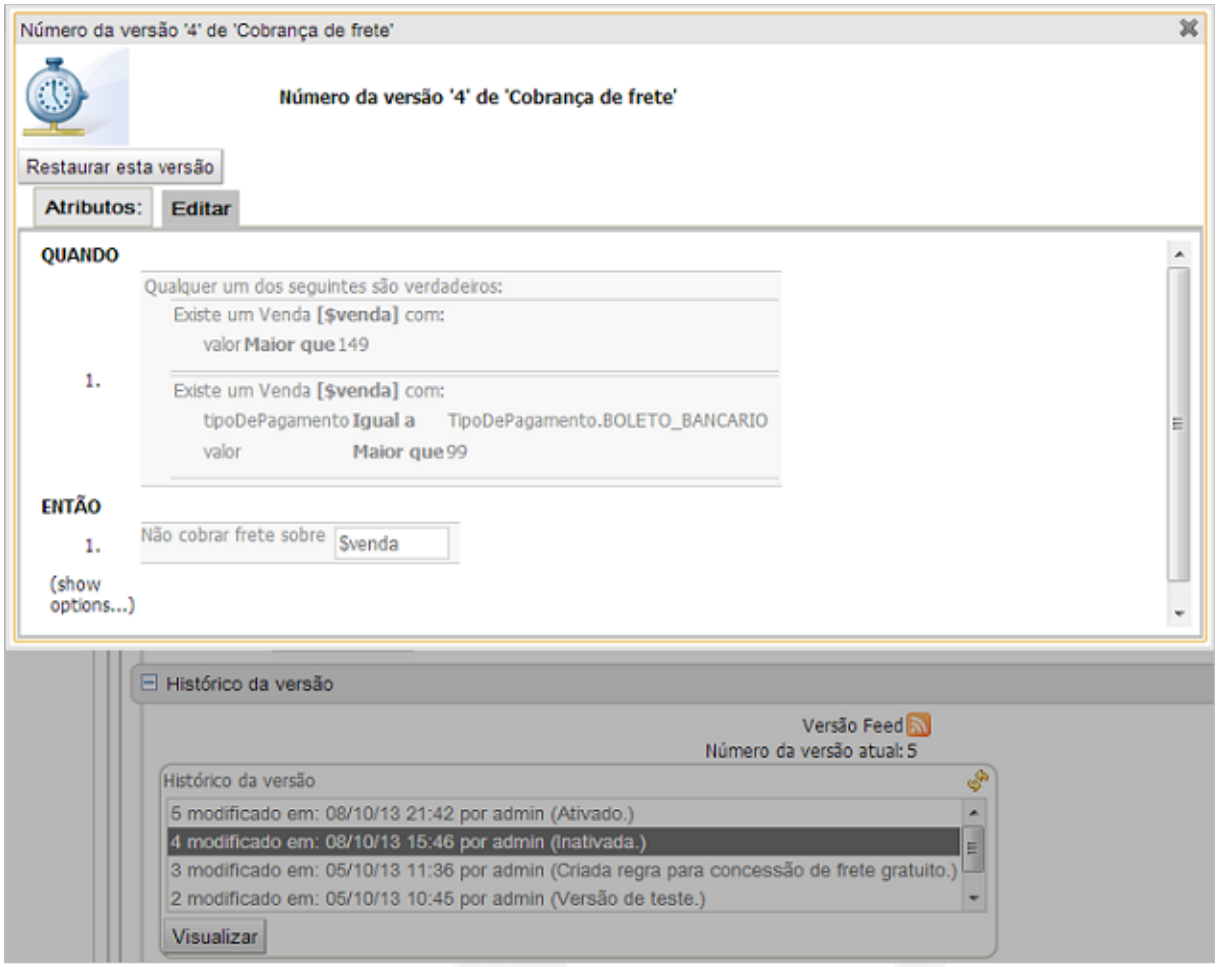
4.3.5 Auditoria e versionamento

Com as regras de negócio mudando com relativa frequência e representando elementos fundamentais de um SI integrado a um BRMS, é vital que as ferramentas de regras possuam maneiras de controlar quatro aspectos cruciais relacionados às suas modificações: quem, quando, como e por quê. É imprescindível, também, que os usuários possam verificar o histórico de cada regra, tendo a possibilidade de voltá-las a uma revisão qualquer quando cometerem algum erro – ou quando assim entenderem.

Neste sentido, o Guvnor cria revisões de todos os seus ativos para cada operação de escrita, sempre possibilitando a informação de um comentário para facilitar a identificação do que foi feito. Com isso, o usuário da ferramenta consegue visualizar detalhadamente todo o histórico de uma regra, tendo a possibilidade de restaurá-la a um estado previamente existente. No entanto, embora seja possível visualizar a regra em determinada revisão, o ponto negativo é a inexistência de um recurso de comparação da entre duas revisões. Assim, o usuário acaba sendo forçado a abrir as revisões desejadas e procurar visualmente as modificações.

A Figura 28 apresenta o histórico de versões de determinada regra, na parte de baixo, e a própria regra em uma versão anterior, na parte de cima.

Figura 28 - Histórico de versões de uma regra no Guvnor

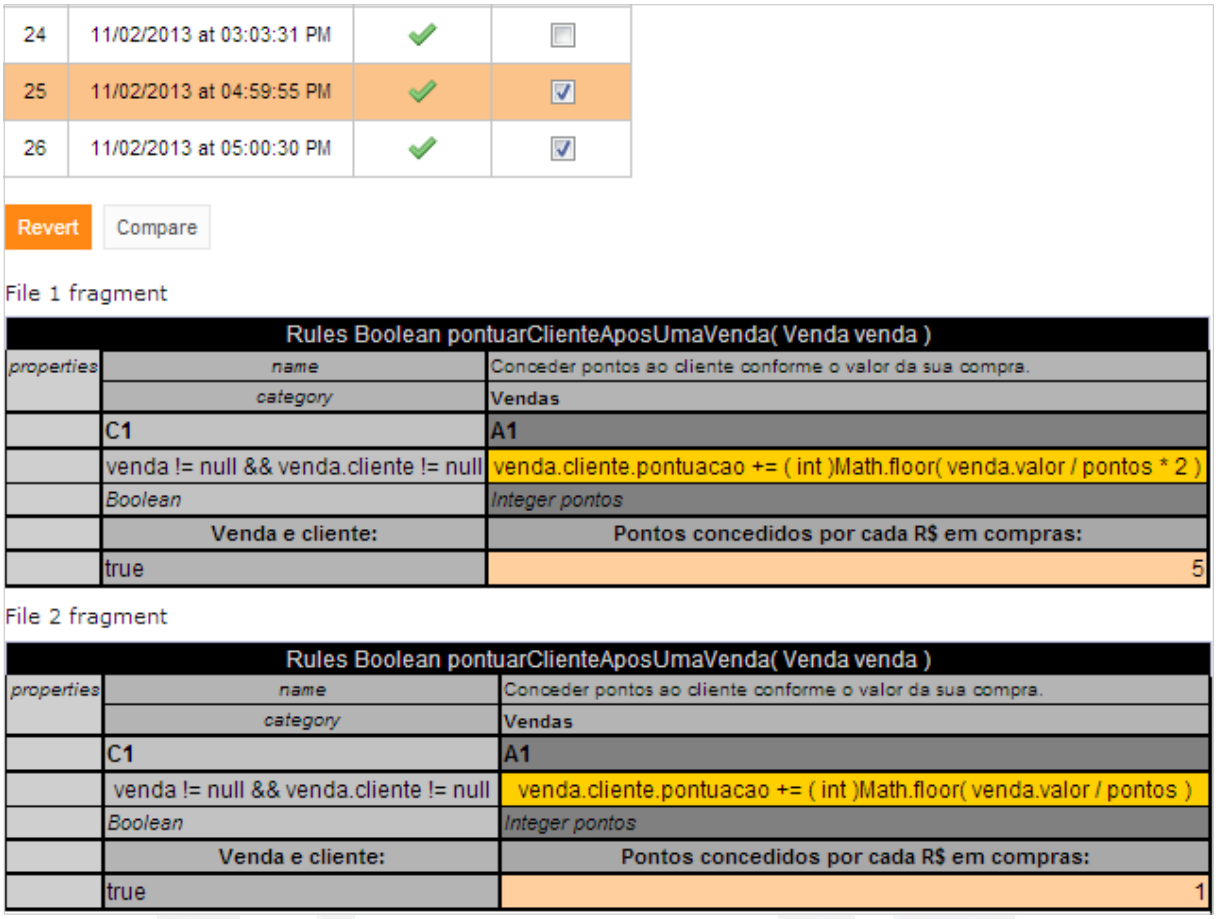


Fonte: elaborado pelo autor.

Embora o Web Studio também crie revisões para cada operação de escrita de regras, a ferramenta não disponibiliza um recurso para visualizar o histórico completo de uma regra em específico, pois suas revisões seguem um modelo de sequenciamento geral. Dessa forma, o usuário precisa abrir cada uma das revisões – sempre são exibidas todas as existentes para o projeto – e verificar se foi feito algo com a regra desejada. Não obstante, tal tarefa se torna ainda mais complicada pelo fato de não ser possível informar comentários no momento de salvar uma regra.

A vantagem do Web Studio, porém, é que possui justamente o recurso inexistente no Guvnor: a comparação de uma regra em diferentes revisões. A Figura 29 ilustra esse recurso, que deixa destacadas as células modificadas.

Figura 29 - Comparação de regra em diferentes revisões no Web Studio



24	11/02/2013 at 03:03:31 PM	✓	☐
25	11/02/2013 at 04:59:55 PM	✓	☑
26	11/02/2013 at 05:00:30 PM	✓	☑

Revert Compare

File 1 fragment

Rules Boolean pontuarClienteAposUmaVenda(Venda venda)		
properties	*name*	Conceder pontos ao cliente conforme o valor da sua compra.
	category	Vendas
	C1	A1
	venda != null && venda.cliente != null	venda.cliente.pontuacao += (int)Math.floor(venda.valor / pontos * 2)
	Boolean	*Integer pontos*
	Venda e cliente:	**Pontos concedidos por cada R$ em compras:**
	true	5

File 2 fragment

Rules Boolean pontuarClienteAposUmaVenda(Venda venda)		
properties	*name*	Conceder pontos ao cliente conforme o valor da sua compra.
	category	Vendas
	C1	A1
	venda != null && venda.cliente != null	venda.cliente.pontuacao += (int)Math.floor(venda.valor / pontos)
	Boolean	*Integer pontos*
	Venda e cliente:	**Pontos concedidos por cada R$ em compras:**
	true	1

Fonte: elaborado pelo autor.

Em resumo, o modelo de versionamento do Guvnor se mostra muito mais eficiente e aplicável. Isso porque a possibilidade de visualização do histórico completo de uma regra torna os trabalhos de auditoria e pesquisa muito mais simples, uma vez que não é necessário procurar por modificações em revisões das quais não se possui descrição e nem mesmo autor.

4.3.6 Integração externa

Uma característica fundamental para sistemas de BRMS, é permitir que as regras criadas em seu repositório sejam acessíveis por sistemas externos. Diferentemente dos exemplos apresentados nos itens 3.1.1 e 3.2.1, do Capítulo 3, que exemplificam a utilização das APIs do Drools e o OpenL Tablets, tanto o Guvnor quanto o Web Studio possibilitam a disponibilização de suas regras como serviços que podem ser consumidos via *web services*, criando uma arquitetura orientada a serviços – conhecida como Service Oriented Architecture

(SOA)[22]. A grande vantagem desse tipo de abordagem é o fato das regras passarem a ser acessíveis por sistemas e programas desenvolvidos independentemente da linguagem de programação, ao contrário da utilização das APIs, que são disponibilizadas apenas para linguagens específicas – no caso do Drools e do OpenL Tablets, apenas para Java.

Sistemas de BRMS devem poder ser incorporados em arquiteturas SOA e suas regras precisam permitir sua disponibilização como *web services*. Os softwares de estudo deste trabalho permitem que isso seja feito, embora tais recursos façam parte da etapa de configuração da ferramenta, o que, consequentemente, não é feito por usuários do negócio, mas, sim, por pessoas da TI com conhecimento técnico para tal.

4.3.7 Repositório

Por padrão, o repositório de regras do Guvnor é salvo de acordo com o *framework* Apache Jackrabbit, uma implementação Java Content Repository (JCR). Muitas vezes, porém, é necessário – ou faz parte da cultura da organização – a utilização de um SGBD para isso, fato que a documentação da própria ferramenta indica ser mais confiável do que a implementação padrão. Para isso, atualmente, o Guvnor suporta os seguintes SGBDs: Microsoft SQL Server, MySQL, Oracle – versões 9i, 10g e 11, PostgreSQL, Derby e H2.

O Web Studio, ao contrário, por padrão salva seu repositório em um diretório local da máquina onde está sendo executado – e não há, na documentação oficial do projeto, detalhamento sobre este comportamento. No entanto, também é possível que o repositório seja salvo em um servidor remoto e seja acessado via Remote Method Invocation (RMI)[23] ou pelo protocolo WebDav[24].

Esse tipo de configuração não é demonstrado neste trabalho visto ser tarefa especificamente de cunho técnico, assim como a configuração dos *web services*, ficando a cargo e sob responsabilidade totais de usuários da TI. Consequentemente, não têm interferência – e, até mesmo, importância – no trabalho de usuários do negócio.

22 Conforme Graham (2007), SOA é uma forma de desenvolvimento de softwares que enfatiza a utilização de serviços para suportar os requisitos do negócio. Neste modelo, os serviços podem ser utilizados individual ou coletivamente, por outros serviços, e são independentes de tecnologias específicas, sendo geralmente acessados através de padrões como Simple Object Access Protocol (SOAP) e Web Services Description Language (WSDL).

23 RMI é a implementação Java do protocolo Remote Procedure Call (RPC), que permite a execução de procedimentos existentes em outras máquinas, em um ambiente distribuído (LAROQUE, 2003).

24 WebDav é uma extensão do protocolo Hipertext Transfer Protocol (HTTP) que permite que os clientes publiquem, bloqueiem e gerenciem recursos *web* (FITZPATRICK; PILATO; SUSSMAN, 2011).

5 CONCLUSÃO

A utilização em massa de sistemas de informação por organizações de todos os portes, para a resolução de todo tipo de problema, faz com que eles implementem as regras de negócio que guiam a empresa no alcance de suas metas e objetivos. Sua atualização constante, portanto, se tornou uma nova necessidade, visto que, devido a políticas, regulamentações, leis ou novas oportunidades de mercado, as regras de negócio costumam mudar com muita frequência. É fato, porém, que estas regras não são responsabilidade da área de tecnologia da informação, uma vez que sua definição é feita, geralmente, pela alta gerência de uma empresa ou por pessoas designadas especificamente para este fim, como os analistas de negócio.

Dessa forma, acabou surgindo um novo desafio para os envolvidos com tecnologia da informação: como possibilitar a atualização constante das regras de negócio de um sistema de informação, permitindo o acesso às regras pelo próprio negócio e tornando este apto à realização de alterações quando necessário? O principal objetivo deste trabalho, assim, foi verificar se algumas das ferramentas de BRMS utilizadas atualmente atendem ao que determina a bibliografia: solucionar o desafio supracitado e, como consequência, remover a lacuna existente entre as áreas de negócio e tecnologia da informação.

De forma resumida, pode-se concluir que as ferramentas analisadas neste trabalho não estão capacitadas a fornecer, para as pessoas do negócio, toda a autonomia esperada com a implantação de um BRMS. Foi possível obter esta resposta, inclusive, sem que os testes fossem realizados por pessoas do negócio, uma vez que o desenvolvimento das regras em qualquer uma das ferramentas, mesmo sendo feito por uma pessoa com conhecimento técnico em linguagens de programação, mostrou-se um processo complexo e demorado, que exigiu diversas tentativas para ser terminado.

A implantação do JBoss Drools ou do OpenL Tablets como BRMS de uma organização, portanto, poderia resultar no aumento da complexidade do trabalho dos envolvidos, contrariando a agilidade exigida pela demanda do mercado. Além disso, a tão sonhada independência da área de TI não seria alcançada, uma vez que qualquer regra de complexidade mínima exigiria a intervenção de pessoas com conhecimentos de linguagens de programação e das especificações da ferramenta utilizada.

Por outro lado, a possibilidade de ter as regras de negócio separadas do código-fonte das aplicações, permitindo seu constante reaproveitamento, acaba sendo um dos atrativos para

que empresas adotem sistemas de BRMS. É inegável, portanto, que a existência de um repositório central para as regras de negócio de uma organização, criado especificamente para este fim e acessível por usuários que não são exclusivamente técnicos, poderia criar um ambiente de maior colaboração entre as áreas envolvidas. Consequentemente, poderia resultar na visão de que o software da empresa não é apenas um elemento da TI.

Além disso, um modelo de desenvolvimento híbrido – no qual os analistas e desenvolvedores de sistemas criassem um grande conjunto de sentenças DSL e/ou as tabelas de decisão mais complexas, com o negócio sendo responsável apenas por manutenções mais simples referentes a dados, por exemplo – poderia criar a impressão de que o usuário do negócio realmente participa do desenvolvimento do software. Afinal de contas, o sistema deve funcionar conforme as necessidades e especificações deste usuário.

De qualquer forma, é preciso cuidado antes de optar pela implantação de um BRMS. Deve-se levar em conta fatores como a complexidade do projeto, a necessidade de atualizações e o próprio conhecimento da ferramenta. Conforme Browne (2009), não deve ser escolhida uma ferramenta de regras se a aplicação for relativamente simples ou se o projeto possui prazo que impeça a equipe toda de aprender satisfatoriamente o funcionamento da ferramenta escolhida. Por outro lado, o mesmo autor indica casos de alteração constante da lógica de negócio como ideais para a adoção de um BRMS.

O desenvolvimento deste trabalho deixou claro o fato de que as regras de negócio, por pertencerem ao próprio negócio, não devem ficar sob responsabilidade – nem dependerem da implementação – de equipes de TI. A necessidade de atualização constante das regras, bem como a habilidade dos sistemas de informação para reagir rapidamente, tornam-se essenciais não apenas para o sucesso da empresa, mas também para sua sobrevivência básica em um mundo de competição globalizada. É fundamental, portanto, que haja softwares capazes de permitir que as regras de negócio sejam controladas por pessoas do negócio, embora as ferramentas disponíveis atualmente – ao menos as testadas neste trabalho – não estejam aptas a atenderem satisfatoriamente a esta demanda.

REFERÊNCIAS

3 SCALE. **What is an API? Your guide to the Internet Business (R)evolution**. 2012. Disponível em: <http://www.3scale.net/wp-content/uploads/2012/06/What-is-an-API-1.0.pdf>. Acesso em: 19 maio 2013.

AMADOR, Lucas. **Drools Developer's Cookbook**. Birmingham: Packt Publishing, 2012.

ANDREESCU, Anca; MIRCEA, Marinela. **Informatica Economica.** Managing Knowledge as Business Rules, Bucharest, v. 13, n. 4, p. 63-74, 2009.

APPLETON, Daniel S. Datamation. **Business Rules: The Missing Link**. p. 145-150, outubro 1984.

BAJEC, Marko; KRISPER, Marjan. Information Systems. **A methodology and tool support for managing business rules in organisations**, Oxford, v. 30, n. 6, p. 423-443, setembro 2005.

_____. Issues and challanges in business rule-based information systems development. In: ECIS 2005, 13., 2005, Regensburg, **Information Systems in a Rapidly Changing Economy**. Regensburg, 2005. p. 887-898.

BALI, Michael. **Drools JBoss Rules 5.0**: Developer's Guide. Birmingham: Packt Publishing, 2009.

BRINCK, Tom; GERGLE, Darren; WOOD, Scott. **Usability for the web**: Designing Web Sites that Work. [S. I.]: Sagebrush Education Resources, 2001.

BROWNE, Paul. **JBoss Drools Business Rules**: Capture, automate, and reuse your business processes in a clear English language that your computer can understand. Birmingham: Packt Publishing, 2009.

CARKENORD, Barbara A. Seven Steps to Mastering Business Analysis. **The Bridge**, [S. I.], v. 5, n. 2, 2008. Disponível em: <http://www.b2ttraining.com/wp-content/uploads/thebridgefall2008.pdf>. Acesso em: 11 maio 2013.

CASEY, Keith. **Unit Testing Strategies**. 2010. Disponível em: <http://www.phparch.com/wp-content/uploads/2010/11/CWX_UnitTesting.pdf >. Acesso em: 31 outubro 2013.

CRUZ, Carla; RIBEIRO, Uirá. **Metodologia Científica**: Teoria e Prática. Rio de Janeiro: Axcel Books, 2003.

DATE, C. J. **Twelve Rules for Business Rules**. 2000. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.197.266&rep=rep1&type=pdf>.

Acesso em: 22 maio 2013.

DRUZDEL, Marek; FLYNN, Roger. Encyclopedia of Library and Information Science. **Decision Support Systems**, New York, n. 2, 2002.

FITZPATRICK, Brian W.; PILATO, C. Michael; SUSSMAN, Ben Collins. **Version Control with Subversion**. 2011. Disponível em: <http://svnbook.red-bean.com/en/1.7/svn-book.pdf>. Acesso em: 3 novembro 2013.

FORGY, Charles. **Artificial Intelligence.** Rete: A fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, n. 19, p. 17-37, 1982.

GERRITS, Rick. Putting Business Rules in the Hands of the Business. **Business Rules Journal**, [S. I.], v. 13, n. 4, abril 2012. Disponível em: <http://www.brcommunity.com/a2012/b645.html>. Acesso em: 11 maio 2013.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4 ed. São Paulo: Atlas S.A., 2002.

GOTTESDIENER, Ellen. Business Rules – Show Power, Promise. **Application Development Trends**, v. 4, n. 3, março 1997. Disponível em: <http://www.ebgconsulting.com/Pubs/Articles/BusinessRulesShowPowerPromise-Gottesdiener.pdf>. Acesso em: 08 maio 2013.

GOUGEON, Alain. **Everything you ever wanted to know about Business Rules**. 2003. Disponível em: <http://www.bptrends.com/publicationfiles/07-03 ART Everything About Bus Rules - Gougeon.pdf>. Acesso em: 22 maio 2013.

GRAHAM, Ian. **Business Rules Management and Service Oriented Architecture**: A Pattern Language. New York: John Wiley & Sons, Inc, 2007.

GUEDES, Gilleanes T. A. **UML 2: uma abordagem prática**. São Paulo: Novatec Editora, 2009.

HALLE, Barbara von. **Business Rules Applied**: Building Better Systems Using the Business Rules Approach. New York: John Wiley & Sons, Inc, 2002.

HAY, David; HEALY, Kery A. **The Guide Business Rules Project**: Final Report. Chicago: Guide International Corporation, 2000. v. 1.3.

HERBST, H.; KNOLMAYER, G.; MYRACH, T.; SCHLESINGER, M. The Specification of Business Rules: A Comparison of Selected Methodologies. In: IFIP WG 8.1 Conference, 1994, Amsterdan, **Proceedings of the IFIP WG 8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle**. Amsterdan, 1994, p. 29-46.

HERBST, H.; MYRACH, T. Database Application Semantics. **A Repository System for Business Rules**, London, p. 119-138, 1997.

JACOB, R. J. K.; FROSCHER, J. N. IEEE Transactions on Knowledge and Data Engineering. **A software engineering methodology for rule-based systems**, [S. I.], v. 2, n. 2,

p. 173-189, 1990.

KOORNNEEF, Maarten. **Technology and Implementation of Business Rules Management Systems**. Rotterdam: Erasmus University Rotterdam, 2006. Disponível em: <http://www.kneef.nl/dl/Afstudeerscriptie Maarten Koornneef.pdf>. Acesso em: 15 maio 2013.

KOVACIC, Andrej. Business Process Management Journal. **Business renovation: business rules (still) the missing link**, v. 10, n. 2, p. 158-170, 2004.

KRUG, Steve. **Não me faça pensar!** Uma abordagem de bom senso à usabilidade na Web. [S. I.]: Alta Books, 2006.

LAROQUE, Philippe. **Remote Method Invocation in Java**. 2003. Disponível em: <http://depinfo.u-cergy.fr/~pl/wdocs/coo/rmi.pdf>. Acesso em: 3 novembro 2013.

MORGAN, Tony. **Business Rules and Information Systems**: Aligning IT with Business Goals. Indianapolis: Addison-Wesley, 2002.

NARAYANAN, Rajagopalan. Business Rules Change All the Time, But Your Applications Don't Have To. **SAP Insider**, [S. I.], v. 10, n. 2, abril 2009. Disponível em: <http://www.sap.com/campaign/emea/Assets/09_BRMWebinar_Sept/Business_Rules_Change_All_The_Time.pdf>. Acesso em: 12 maio 2013.

OCHEM, Nicolas. **Business Logic management in a Web Application**. Stockholm: Royal Institute of Technology, 2008. Disponível em: <http://web.it.kth.se/~johanmon/theses/ochem.pdf>. Acesso em: 17 maio 2013.

PONNIAH, Paulraj. **Data Warehousing Fundamentals**. New York: John Wiley & Sons, Inc, 2001.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995. Tradução de José Carlos Barbosa dos Santos.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Database Management Systems**. [S. I.]: McGraw-Hill, 2001.

REZENDE, Denes Alcides. **Engenharia de software e sistemas de informação**. 3 ed. Rio de Janeiro: Brasport, 2005.

ROSENBERG, Florian; DUSTDAR, Schahram. Busines Rules Integration in BPEL: A Service-Oriented Approach. In: Seventh IEEE International Conference on E-Commerce Technology, 2005, Washington, **Proceedings...**. Washington, 2005, p. 476-479.

ROSS, Ronald G. What is a “business rule”? **BR Community**, março 2000. Disponível em: <http://www.brcommunity.com/b005.php>. Acesso em: 05 maio 2013.

_____. Business Rules Manifesto: The Principles of Rule Independence. **Business Rules Group**, v. 2, novembro 2003. Disponível em: <http://www.businessrulesgroup.org/brmanifesto.htm>. Acesso em: 06 maio 2013.

_____. Rule Speak® Sentence Forms: Specifying Natural-Language Business Rules in English. **Business Rules Journal**, [S. I.], v. 10, n. 4, abril 2009. Disponível em: <http://www.brcommunity.com/a2009/b472.html>. Acesso em: 22 maio 2013.

_____. Requirements are Rules: True or False? **Business Rules Journal**, [S. I.], v. 14, n. 3, março 2013. Disponível em: <http://www.brcommunity.com/a2013/b692.html>. Acesso em: 12 maio 2013.

ROSS, Ronald G; LAM, Gladys S. W. Business Analysis with Business Rules. **Business Rules Journal**, [S. I.], v. 12, n. 10, outubro 2011. Disponível em: <http://www.brcommunity.com/a2011/b616.html>. Acesso em: 11 maio 2013.

SANTOS, Rildo. **Gestão por Processos**. [2010?]. Disponível em: <http://www.slideshare.net/Ridlo/gesto-por-processo>. Acesso em: 07 maio 2013.

SELVEITH, H. Tempora Project. **Modelling Business Rules**. Janeiro 1991.

SHAO, J; POUND, C. J. BT Technology Journal. **Extracting Business Rules from Information Systems**, Massachusetts, v. 17, n. 4, p. 179-186, outubro 1999.

SHPIGEL, Michael. **BPM, BRMS and SOA: Delivering on the Promise of Organizational Agility**. [2011?]. Disponível em: <http://www.mitx.org/files/BPM_BRMS_and_SOA.pdf>. Acesso em: 20 maio 2013.

SOMMERVILLE, Ian. **Engenharia de Software**. 8 ed. São Paulo: Pearson Addison-Wesley, 2007. Tradução de Selma Shin Shimizu Melnikoff, Reginaldo Arakaki, Edilson de Andrade Barbosa.

SUDA, Brian. **SOAP Web Services**. 2003. 75 f. Disponível em: <http://suda.co.uk/publications/MSc/brian.suda.thesis.pdf>. Acesso em: 11 março 2013.

TURBAN, Efraim; KING, David. **Comércio Eletrônico**: Estratégia e Gestão. São Paulo: Pearson Prentice-Hall, 2004. Tradução de Arlete Simille Marques.

VENETIANER, Tom. **Como vender seu peixe na internet**: um guia prático de marketing e comércio eletrônicos. Rio de Janeiro: Campus, 2000.

WILLMOR, D; EMBURY, S. M. Testing: Academia and Industrial Conference – Practice and Research Techniques. **Testing the Implementation of Business Rules Using Intensional Database Tests**, p. 115-126, agosto 2006.

ZANDIPOUR, Ramin. **Evaluation of Interoperability Issues between Business Rules Management Systems of IBM and SAP**. Hamburg: Technische Universität Hamburg-Hamburg, 2011. Disponível em: <http://www.sts.tu-harburg.de/pw-and-m-theses/2011/zandi11.pdf>. Acesso em: 17 maio 2013.