UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

VERIFICAÇÃO DE CONFLITOS EM REGRAS DE INTERAÇÃO DE AMBIENTES COLABORATIVOS

RENATO LUIZ BRESSAN

ORIENTADOR: PROF. DR. LUIZ CAMOLESI JÚNIOR

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA

2008

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

VERIFICAÇÃO DE CONFLITOS EM REGRAS DE INTERAÇÃO DE AMBIENTES COLABORATIVOS

RENATO LUIZ BRESSAN

ORIENTADOR: PROF. DR. LUIZ CAMOLESI JÚNIOR

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA
2008

VERIFICAÇÃO DE CONFLITOS EM REGRAS DE INTERAÇÃO DE AMBIENTES COLABORATIVOS

RENATO LUIZ BRESSAN

ORIENTADOR: PROF. DR. LUIZ CAMOLESI JUNIOR

Dissertação apresentada à Banca Examinadora constituída pelos professores:

_____

Prof. Dr. Luiz Camolesi Junior

UNIMEP

_____

Prof. Dr. Marcos Augusto Francisco Borges

UNICAMP

_____

Prof. Dr. Plínio Roberto Souza Vilela

UNIMEP

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CSCW	Computer-Supported Collaborative Work.
CSCWD	Computer-Supported Collaborative Design.
CSCL	Computer-Supported Collaborative Learning.
ECA	Evento – Condição – Ação.
ECA-P	Política de Evento – Condição – Ação.
SMA	Sistema Multiagente.

1. INTRODUÇÃO

1.1. CONSIDERAÇÕES INICIAIS

O processo de desenvolvimento de sistemas possui entre seus elementos a descrição de rotinas (existentes ou desejáveis) para compor um ambiente regrado e controlado. Para tanto, diversos métodos de análise foram criados e outros ainda serão, para aperfeiçoar esse processo.

Com o melhoramento dos recursos físicos (processadores mais velozes, internet rápida, cobertura de rede de telefonia) e o emprego dessa tecnologia possibilitou uma maior interação entre os atores. Com a facilidade de comunicação, o ambiente onde a ação é executada, em muitos dos casos deixou de ser importante para o modelo.

A interação do mundo físico com o virtual tornou-se um diferencial para as corporações. Empresas com essa característica ganharam agilidade e mobilidade nos processos.

Através de simuladores é possível verificar o resultado de um produto durante a fase de planejamento, essa facilidade aperfeiçoa os processos e reduz os custos de produção.

Em muitos projetos vários especialistas participam do levantamento: permitir que eles atuem colaborativamente é o desafio de muitas equipes. Para possibilitar que uma tarefa seja executada por mais de uma pessoa, o ambiente deve possuir uma estrutura que facilite o modelo.

O papel do analista é fundamental para estruturar um conjunto de regras, ele deve verificar os possíveis conflitos existentes nas regras antes de ativá-las, criando assim uma política de regras.

1.2. OBJETIVO DA PESQUISA

O objetivo deste trabalho é estabelecer uma técnica de verificação da consistência de um conjunto de regras (política) definidas utilizando o modelo M-Forum (CAMOLESI & MARTINS, 2006). A técnica se baseia em princípios da inspeção de especificações para identificar situações de conflito existentes em uma política de interações.

A técnica identifica situações de conflito de regras para análise de um testador, agindo como um pré-teste de especificação que aprimora o desenvolvimento espiral.

As regras com possíveis inconformidades passam pela avaliação refinada de um analista que possua um maior entendimento no assunto, enquanto que os analistas menos experientes alimentam o banco de regras do projeto nas fases de levantamento primário.

Validar uma especificação funcional, ou atributos de um sistema é um desafio na área de desenvolvimento. A presente proposta parte da premissa de que não existe uma estratégia de resolução de conflitos que seja adequada a qualquer tipo de sistema, porém os conflitos detectados podem ser classificados e a utilização de uma estratégia específica em função do tipo de conflito pode conduzir a resultados mais satisfatórios.

Os conflitos de regras geralmente ocorrem quando temos um mesmo objeto ou ator para executar ações opostas, um exemplo simples é a especificação de passar por uma porta, nela temos: a regra de um ator abrir a porta, a regra de um ator passar pela porta aberta e de um ator fechar a porta.

Imaginemos o trabalho de um verificador de regras onde abrir e fechar, ações antagônicas aponta como regras conflitantes, porém para a especificação do modelo são regras corretas.

Para tanto o objetivo é identificar os tipos de conflitos para um testador identificá-los e se possível tratá-los, tendo na fase de especificação um pré-

teste de conceito, agilizando o processo e reduzindo os custos do projeto.

1.3. MOTIVAÇÃO

A dinamização do trabalho colaborativo traz a redução de custo, a velocidade de desenvolvimento necessária para aproveitar as oportunidades de mercado, a otimização dos recursos humanos e não-humanos e a reciclagem, visando melhoramento e disseminação do conhecimento.

Criar uma técnica que possibilite a verificação de uma tarefa colaborativa por um testador agindo como um pré-teste antes do final da mesma, melhorando o processo e reduzindo o custo de sua realização.

Normalmente um ambiente compartilhado coexiste com ambientes privados de cada elemento do grupo, isso implica em possuir diferentes mecanismos de gestão da informação, principalmente com o ajuste da concorrência do acesso.

Atualmente a demanda de CSCW (COMPUTER SUPPORT COOPERATIVE WORK) é voltada para pesquisa do trabalho cooperativo de usuários em muitas frentes, entre elas: os projetos de engenharia; na especificação e desenvolvimento de sistemas; na coordenação de processos de trabalho (workflow); na telecooperação; no ensino; na arquitetura; na telemedicina.

1.4. METODOLOGIA DE TRABALHO

Na busca de como ‘validar os conflitos em regras’ surgiu a ‘técnica de verificação de conflitos em regras de ambientes colaborativos’. A pesquisa percorreu o ramo da inteligência artificial e do conhecimento incremental focando o tratamento dado ao problema de certificar um conjunto de regras colaborativas.

A pesquisa mostrou a atenção de alguns grupos que compactuam dos mesmos interesses em possibilitar o trabalho colaborativo, porém existem poucas publicações que tratam a validação de regras.

Nos modelos em que a validação é presente existe a figura de um mediador, o projetista principal, que define os conflitos de regras da especificação que foram selecionadas pelo modelo.

O que é conflito, quais deles podem ser verificados no modelo M-Forum (CAMOLESI & MARTINS, 2006), com quais instrumentos e de que forma, foi o centro da pesquisa e da proposta.

A proposta deste trabalho foi estabelecer uma técnica de verificação da consistência de um conjunto de regras (política) definidas utilizando o modelo M-Forum. Para isso, a técnica baseia-se em princípios da inspeção de especificações para identificar situações de conflito existentes em uma política de interações.

O trabalho está dividido em duas áreas de verificação com soluções diferentes, a verificação de política intra-regras, que foca as ações presentes na própria regra e a verificação de política inter-regras, que foca as interações de elementos comuns nas várias regras.

Para compor a verificação foram utilizadas tabelas lógicas e diagramas que facilitaram o processo permitindo os cruzamentos das dimensões (atores, atividades, objetos, tempo e espaço) auxiliando na busca de possíveis conflitos.

1.5. ESTRUTURA DA DISSERTAÇÃO

A dissertação está dividida em cinco capítulos. No primeiro capítulo estão descritos os objetivos da pesquisa e os motivos que incentivaram sua realização.

No segundo capítulo são tratados os conceitos de ambientes colaborativos e cooperativos e algumas soluções de autores que pesquisaram o tema e propuseram um modelo para tratá-los.

No terceiro capítulo com base em um dos modelos propostos para especificar ambientes colaborativos, o M-Forum (CAMOLESI & MARTINS, 2006), é proposto um método de verificação.

O quarto capítulo descreve um estudo empírico do método de verificação proposto no capítulo anterior com uma análise de eficiência.

No quinto capítulo e último, conclui a proposta.

2. AMBIENTES COLABORATIVOS E COOPERATIVOS

2.1 CONSIDERAÇÕES INICIAIS

Compreender os termos ‘colaboração’ e ‘cooperação’ na área de CSCW é fundamental para um bom entendimento do assunto, várias interpretações diferentes são emitidas em artigos e muitos autores simplesmente tratam os termos como sinônimos, quando outros (DILLENBOURG et al. 1995) traçam uma distinção entre eles:

A cooperação e a colaboração não diferem nos termos se a tarefa está distribuída, mas pela maneira em que é dividida. Na cooperação a tarefa é partilhada hierarquicamente em sub-tarefas independentes. Nos processos de colaboração as tarefas podem ser divididas em camadas entrelaçadas existindo uma dependência entre elas.

Quando uma tarefa é realizada por mais de um ator sobre o mesmo objeto, estamos executando de forma colaborativa, já quando uma tarefa é partilhada e executada por mais de um ator, mas sobre objetos distintos com relacionamento, estamos executando de forma cooperativa, porque a ação no objeto é executada por apenas um ator.

Temos portanto, que mais de uma pessoa enchendo um balde executam uma tarefa colaborativa e mais de uma pessoa trabalhando em uma linha de montagem de um veículo executam várias sub-tarefas cooperativas.

O primeiro passo de uma tarefa cooperativa é a necessidade de coordenação das sub-tarefas independentes. Neste contexto, a coordenação pode ser definida como “o gerênciamento de dependências entre atividades e o suporte de (inter) dependências entre atores” (BORDEAU e WASSON 1997).

Os ambientes virtuais de trabalho e de aprendizagem colaborativos, através da interconexão entre máquinas nas redes de computadores locais e pela Internet,

são foco de pesquisa e desenvolvimento de tecnologias na área de CSCW. Esse ambiente recebe a denominação de *Groupware*.

O termo *Groupware* como é apresentado na figura 1, designa a tecnologia (hardware e / ou software) gerada pelas pesquisas sobre CSCW. As ferramentas utilizadas para o trabalho em grupo podem ser síncronas e assíncronas (MOECKEL, 2001). Um dos requisitos fundamentais da tecnologia Groupware é que os sistemas sejam altamente configuráveis, para se adaptarem às necessidades dos usuários (BROOKE,1993).

Groupware pode ser considerado um guarda-chuva, que engloba diversas tecnologias baseadas no mesmo princípio: pessoas trabalhando juntas para que as atividades sejam realizadas com sucesso em todas as partes do processo, independente de quem as desenvolva.

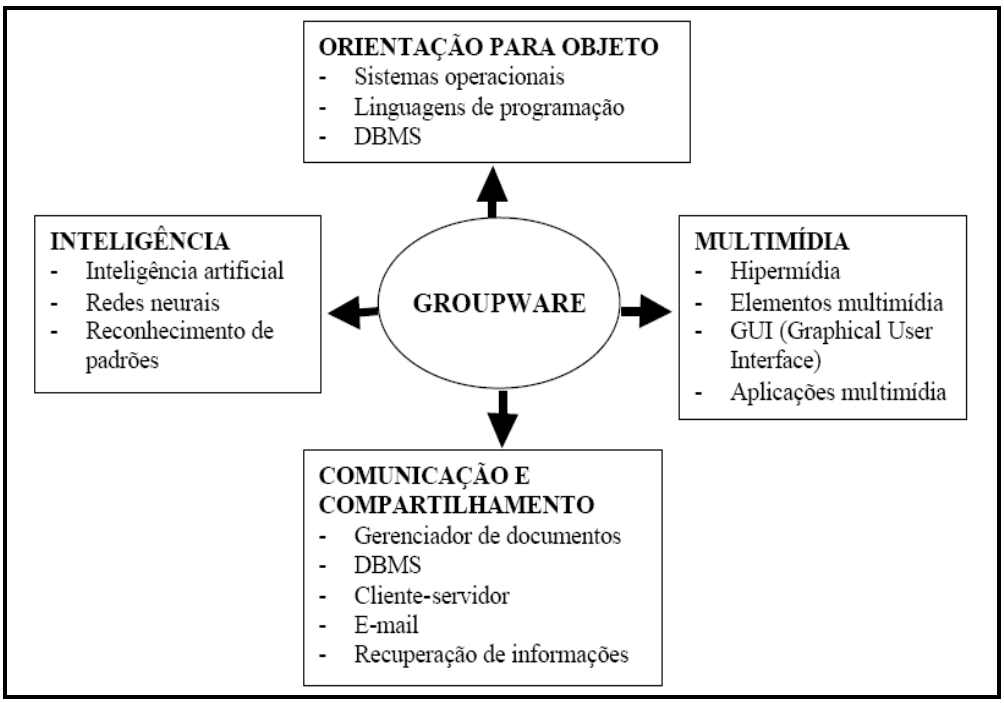

FIGURA 1 – MODELO GROUPWARE (BROOKE,1993)

Estes ambientes devem prover elementos de percepção de forma a permitir a coordenação em tarefas cooperativas, principalmente onde a comunicação direta não ocorre.

Uma comunicação pode ser definida como um processo, pois a informação é trocada entre indivíduos através de um sistema comum dos símbolos, sinais ou comportamentos, “a comunicação é o cimento da organização e quanto maior a necessidade de coordenação e cooperação, maior a necessidade de comunicação”. (BREHMER 1991)

Para possibilitar uma cooperação eficiente, a percepção torna-se um fator fundamental na comunicação, coordenação e cooperação de um grupo de trabalho. Além disso, os ambientes devem prover elementos de percepção para que indivíduos possam interpretar eventos, prever possíveis necessidades e transmitir informações de maneira organizada.

Perceber as atividades dos outros indivíduos é essencial para o fluxo e naturalidade do trabalho e para diminuir as sensações de impessoalidade e distância, comuns nos ambientes virtuais (FUKS & ASSIS 2001).

Enquanto a interação entre pessoas e ambiente dentro de uma situação face-a-face parece natural, visto que sentidos como visão e audição estão disponíveis em sua plenitude, a situação fica menos clara quando há a tentativa de fornecer suporte à percepção em ambientes virtuais (ASSIS, 2000).

Os membros do grupo podem ajudar à identificar inconsistências no raciocínio dos indivíduos e buscar em conjunto idéias, informações e referências para auxiliar na resolução dos problemas.

Geralmente, o grupo tem mais capacidade de gerar alternativas, levantar as vantagens e desvantagens de cada uma, selecionar as viáveis e tomar decisões do que os indivíduos separadamente (TUROFF & HILTZ, 1982).

Para possibilitar a coordenação do grupo são necessárias informações sobre o que está acontecendo para que seja possível tomar decisões adequadas sobre os procedimentos a serem tomados para favorecer a cooperação.

Estas informações são fornecidas através de elementos de percepção que capturem e condensem as informações coletadas sobre a interação dos participantes.

Para trabalhar colaborativamente, um indivíduo tem que compartilhar idéias, estar em sintonia com os outros membros do grupo e realizar suas tarefas de maneira satisfatória (FUKS et al, 1999).

As pessoas devem se comunicar para coordenar seus esforços de trabalho e cooperar em torno de um objetivo. Para cooperação, há a necessidade de comunicação, seja ela direta ou através de informações obtidas dentro do ambiente de trabalho.

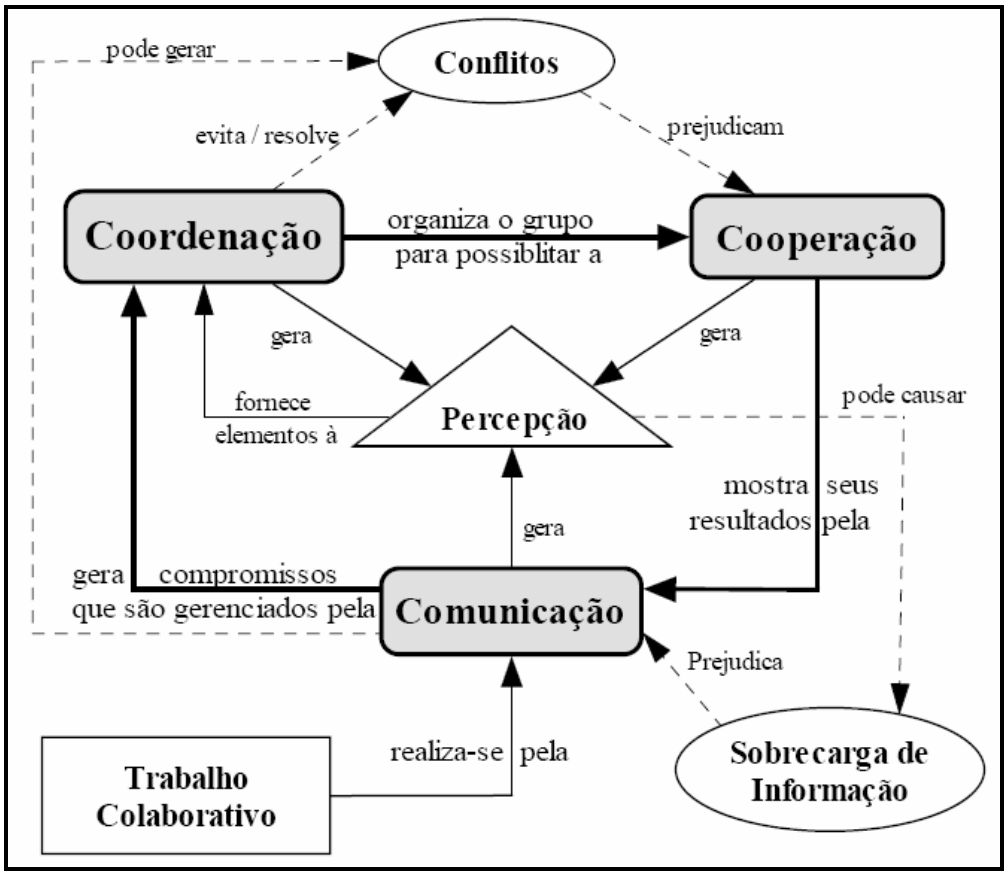

FIGURA 2 – GERÊNCIA DO MODELO COLABORATIVO (FUKS & ASSIS, 2001)

Em cada relacionamento há o estímulo fornecido pelas informações de percepção e que possibilitam a ocorrência do entendimento compartilhado em torno de um objetivo de cooperação para resolução de uma tarefa ou de todo o trabalho.

Tendo percepção das atividades dos companheiros e dos impactos que

ocorrem no conhecimento gerado pela cooperação, as pessoas terão informações que auxiliam na sincronização do trabalho, coordenando-se em torno de seus contextos individuais (FUKS & ASSIS, 2001).

Há diversos tipos de elementos de percepção, classificados por finalidade, tempo, escopo, abstração, agregação, perspectiva, forma de fornecimento, personalização, entre outros. (BRINCK & MCDANIEL, 1997).

Estes elementos visam responder basicamente às questões “quem, o quê, onde, quando e como”. Em todo ambiente deve-se fazer estas perguntas buscando identificar quais elementos os usuários deveriam conhecer para perceber a situação e proporcionar o entendimento.

Uma vez identificados os elementos de percepção, ou seja, os elementos que devem ser percebidos pelos participantes, o próximo passo é analisar como as informações serão reunidas e distribuídas.

Deve-se levar em conta se a informação vai ser explicitamente gerada, direcionada e separada do objeto de trabalho compartilhado; ou passivamente colhida, distribuída e apresentada no mesmo ambiente compartilhado como um objeto da cooperação.

Cabe ressaltar ainda que um elemento de percepção não existe sozinho em um sistema e sim estará implícito nos mecanismos de comunicação, coordenação e/ou cooperação.

2.2 AMBIENTES BASEADOS EM REGRAS

Existem vários ambientes baseados em regras, seu precursor foi na área de bases de dados ativos onde surgiu o termo "regra ativa".

As regras ativas são usadas para impor restrições de integridade estática (nos estados válidos dos dados) ou dinâmica (nas transições de estado válidas), para calcular automaticamente dados derivados (calculados em função de outros) e para sustentar regras de negócio.

Uma aplicação interativa de bases de dados é um programa de computador que permite a um utilizador visualizar e manipular dados armazenados em uma base de dados através de uma interface, disponibilizando formulários, relatórios e gráficos.

Na área de desenvolvimento rápido de aplicações (de bases de dados) também são usados gatilhos (mais limitados) para a validação e possível correção dos dados introduzidos pelo utilizador, para o cálculo automático de dados.

Uma regra ativa dirigida pelos dados, é uma regra ativa com eventos implícitos de um tipo restrito - eventos de modificação de dados - que podem ser inferidos a partir da parte de condição e/ou da parte de ação da regra.

Fórmulas de cálculo de dados derivados em folhas de cálculo ou em ferramentas de desenvolvimento rápido de aplicações em SQL, são exemplos de entidades que podem ser tratadas como regras ativas dirigidas pelos dados.

Na área de ensino, a interação entre os alunos e seu mestre, com a utilização de editores de texto coletivos, permite pensar sobre conceitos de colaboração e cooperação entre outros necessários na realização deste trabalho.

É importante ressaltar a mudança na postura docente que estas novas formas de construção do conhecimento promovem, podendo haver um maior nível de aprendizagem quando se tem trocas entre os participantes.

2.3 MODELOS E LINGUAGENS PARA REGRAS

As regras de interação em um ambiente, de modo geral, são expressas em uma grande variedade de formas e estilos, considerando que a linguagem natural permite esta variabilidade de sua representação (CHANG et al, 1999).

Garantir a integridade do conjunto de regras que regem o ambiente é o fator sustentador da sua governabilidade e crucial para a produtividade e qualidade dos trabalhos realizados.

Hoje, a gerência das regras é foco em vários esforços na indústria e nas universidades. Os trabalhos recentes sobre *e-business*, *web services*, abordam a necessidade global de gestão, intercâmbio e compartilhamento das políticas, regulamentos e regras comerciais entre diferentes sistemas na *web* e em ambientes distribuídos.

2.3.1 MODELO E LINGUAGEM SPIN

Artigos técnicos sobre SPIN são encontrados sobre duas frentes: a primeira, uma ferramenta de software popular de código aberto, utilizada por milhares de pessoas em todo o mundo, que pode ser utilizada para a verificação formal de sistemas de software distribuído.

A ferramenta foi desenvolvida por Bell Labs do grupo Computing Sciences Research Center, com início em 1980. O *software* está disponível livremente desde 1991 e continua a evoluir para acompanhar os novos desenvolvimentos na área. Extensões são permitidas no software através das LTL interface de Xspin.

Em uma delas, a Promela/Spin não é clara a distinção entre o modelo e os critérios de validação. Os critérios são compostos durante a validação, não podem ser totalmente expressos na linguagem validação, ou a linguagem é muito abstrata que afasta do validador e testador os erros que são cometidos durante a tradução dos requisitos informais.

A Rastreabilidade é um problema sério principalmente no modo de manter a ligação entre o sistema que se quer validar e o modelo que eventualmente foi validado.

```
ActivityTemplate Course (AssignedRoles Assistant, Instructor, Student) {
        ObjectType BulletinBoard .....

        ActivityTemplate Examination (Owner Instructor,
                AssignedRoles Examiner) {
                ObjectType ExamPaper .....
                ObjectType AnswerBook .....

                Role Examiner {....}
                Role Examinee (Reect parentActivity.Student) {....}
                Role Grader (Reect parentActivity.Assistant, parentActivity.Instructor) {....}

                ActivityTemplate ExamSession(Owner Grader,
                        Objects (ExamPaper exam, AnswerBook ans),
                        AssignedRoles Candidate) {
                   TerminationCondition #(Checker.Grade._nish)>0
                        Role Candidate {
                                AdmissionConstraints
                                        member(thisUser, parentActivity.Examinee)
                                        & member(thisActivity.Creator, thisUser)
                                        & #members(thisRole)<1
                                ActivationConstraints
                                        date > DATE(May, 10, 2003, 9:00)
                                        & date < DATE(May, 10, 2003, 11:00)
                                Operation OpenExam{
                                        Precondition #(OpenExam.start)=0
                                        Action exam.readPaper() }
                                Operation Write {
                                        Precondition #(OpenExam._nish)>0
                                        Action ans.writeAnswer(data) }
                                Operation Submit {
                                        Precondition Write._nish>0 }
                        }
                        Role Checker {
                                AdmissionConstraints
                                        #members(thisRole)<1
                                        & member(thisUser, parentActivity.Grader)
                                Operation Grade {
                                        Precondition #(Candidate.Submit._nish)=1
                                        Action ans.setGrade(data) }
                        }
                }
        }
}
```

FIGURA 3 – MODELO DE REGRA SPIN (HOLZMANN,1997)

A segunda frente é a que trata regras, o modelo SPIN (HOLZMANN 1997) de verificação de regra inclui um nome de regra, as regras assinaladas para

integração, a regra de admissão e ativação, os gatilhos e sua condição de execução.

Na especificação, uma função *member(role, user)* verifica se um participante está presente na regra e *members(role)* retorna à lista de participantes da regra. O número de participantes em uma regra é *#(members(role))* .

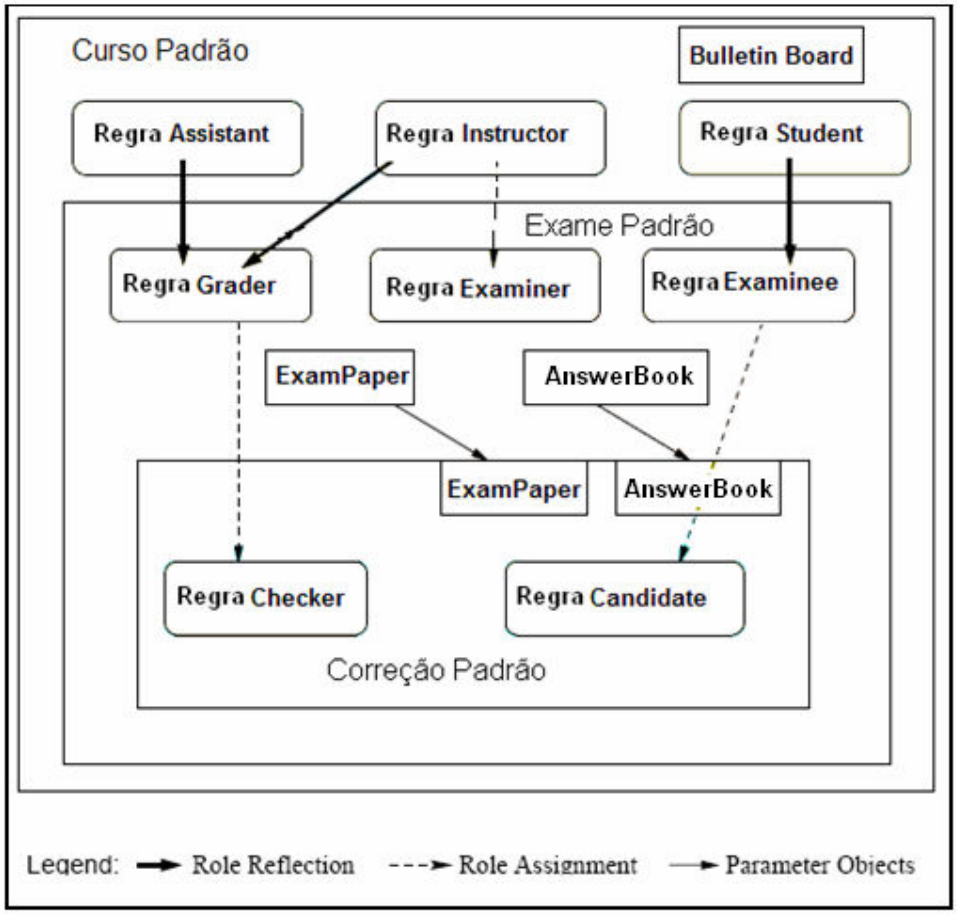

FIGURA 4 – DIAGRAMA DE REGRA SPIN (HOLZMANN, 1997)

A regra de admissão pré-condicionada deve ser verdadeira quando um usuário é admitido para a execução da regra. A Figura 4 exemplifica a verificação da sentença como é mostrado abaixo:

```
AdmissionConstraints
        #members(thisRole) ≤ 1
        & member(thisUser, parentActivity.Staff)
        & !member(thisUser, Student)
```

Essa consistência da regra de admissão facilita a separação do usuário que não pode ser um estudante no curso, onde a qualificação desejada é de ser membro da regra de *Staff* e possui a cardinalidade máxima de um.

Um segundo modelo de admissão é a de regra de atribuição dinâmica onde cada regra de admissão é especificada no momento da instanciação. O tag *AssignedRoles* é usado para especificar quais regras devem ser dinamicamente assinaladas para instanciação.

```
Role Examinee (Reflect parentActivity.Student) {
    Operation StartExam {
    .......
        Action fans=new Object (AnswerBook);
        Act = new Activity ExamSession (
                (exam, ans), Candidate-thisUser)  } } }
```

Quando um examinado invoca esta operação, uma instância do *ExamSession* é criado, o participante da instância é dinamicamente assinalado para a regra *Candidate.*

Para especificar coordenação e políticas de controle de acesso dinâmico são habilitados pré-condições. Um exemplo é a operação de especificação da regra *Examiner,* onde a operação *SetPaper* pode ser executada apenas como especificada em sua pré-condição. Esta operação resulta na criação de um objeto *exam* do tipo *ExamPaper* e uma invocação do método *setQuestions* deste objeto.

```
Role Examiner {
    Operation SetPaper {
        Precondition #(SetPaper.start) = 0
        Action { exam-new Object(ExamPaper)
                exam.setQuestions(data)  }  }  }
```

A regra de validação especifica outras regras em que cada participante pode ou não estar presente como membro para ser validado. Anteriormente, a consistência de ativação da regra especifica as condições que são comuns, pré-condicionando todas as operações definidas para a regra.

2.3.2 LINGUAGEM RULEML

A especificação de RuleML constitui uma família de Web sub-linguaguens modulares, cuja raiz acessa a linguagem como um todo e seus membros são configurados combinando subconjuntos da linguagem.

Cada uma das famílias de sub-linguagens possuem uma definição de XML *Schema* endereçado por uma URI que permite a herança entre as sub-linguagens. O sistema modular de XML *Schema* está atualmente na versão 0,91 (http://www.ruleml.org/modularization).

No primeiro nível da família RuleML como é mostrado na figura 5, estão as distinções de derivação de regras, consultas, consistência de integridade, bem como a produção e reação de regras.

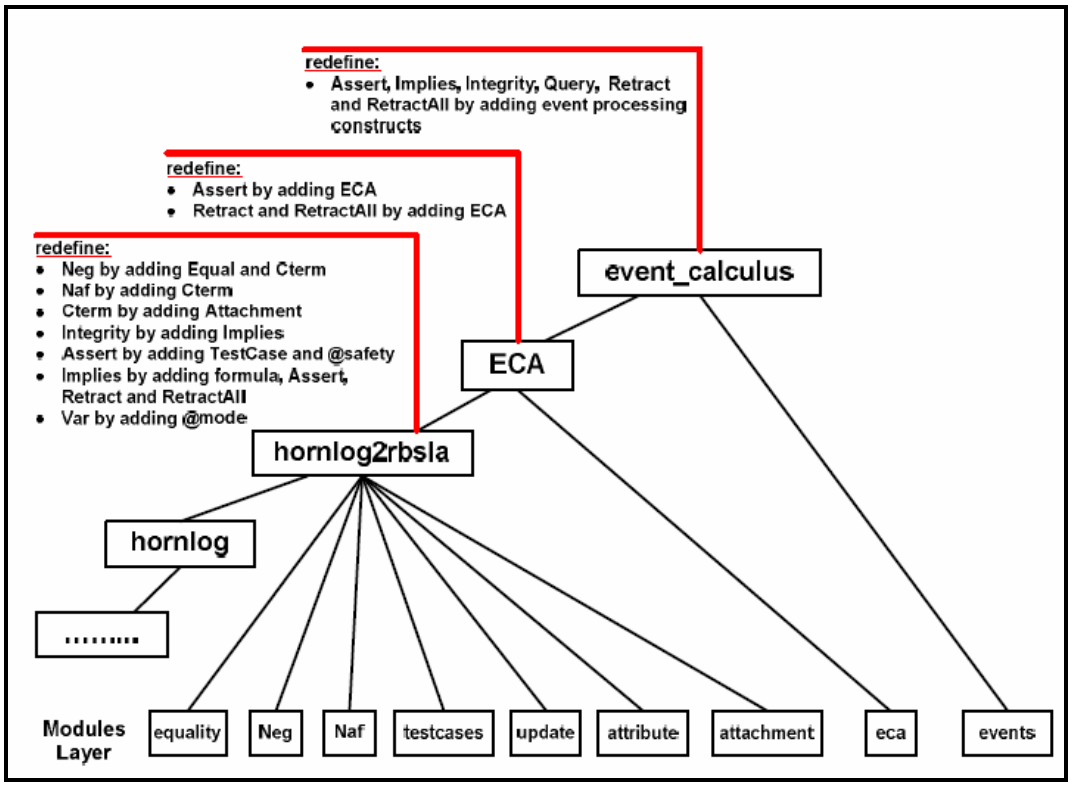

FIGURA 5 – ESTRUTURA DO RULEML (PASCHKE, 2005)

O modelo possui a ação de regra definida como (Naf | Neg | Cterm | Assert | Retract | RetractAll), que é usada nos modelos de <ECA>, <Happens>, <Planned>, <Initiates> and <Terminates>.

O conjunto Attachment habilita a integração do processo de ligação na ECA-RuleML e seus elementos são definidos como (oid?, (Ind | Var | Cterm), Ind). Os elementos <oid>, <Ind>,<Cterm> e <Var> são definidos por RuleML.

Na camada ECA da ECA-RuleML o <Cterm> deve ser redefinido para que o <Attachment> seja incluído. O conteúdo de <Cterm> deve ser alterado por: (oid?, (op | Ctor | Attachment), (slot)*, (resl)?, (arg | Ind | Data | Skolem | Var | Reify | Cterm | Plex)*, (repo)?, (slot)*, (resl)?).

As condições de regras para as ações e eventos (Naf | Neg | Cterm | Assert | Retract | RetractAll) são (oid?, time?, event?, action, postcondition?). O elemento ECA habilita a em série às regras reativas.

Exemplo de ECA-RuleML:

```
<ECA>
        <time>
            <Cterm>
                <Ctor> everySec </Ctor>
                <Ind> 10 </Ind>
            </Cterm>
        </time>
        <action>
            <Cterm>
                 <Ctor> updateKnowledge </Ctor>
            </Cterm>
        </action>
        <postcondition>
            <Cterm>
                <Ctor> test </Ctor>
            </Cterm>
        </postcondition>
</ECA>
```

As regras de evento/condições para as ações (Naf | Neg | Cterm | Assert | Retract | RetractAll) são (oid?, time?, event?, action, postcondition?). Outros elementos fazem parte das regras: <Happens>, <Planned>, <Initiates> e <Terminates>.

A regra reserva em @mode o modo dos atributos, se a variável é intencionada a ser um termo de entrada ou de saída.

O atributo possui três categorias: “?” indefinido, “+” para entrada e “-“ para saída (seu uso é opcional), os atributos são adicionados na lista de elementos <Var> na camada hornlog2eca como é demonstrado na figura 5.

O atributo @safety é restrito para os valores transaction e normal, isto é uma regra para indicar quando a função deve ser iniciada como uma transação e quando não, o atributo é incluído para uma redefinição de <Assert> na camada hornlog2eca e também faz parte da lista de atributos de <Retract> e <RetractAll>.

O atributo @semantics é restrito para valores *string*. Esta regra provê informações sobre diferentes semânticas. Isto ocorre justamente em <TestCase>: nele os elementos definidos no modulo testcases são: (oid?, Test+, Atom*, Implies*, Integrity*) e o uso da @semantics é um atributo opcional.

O RuleML pretende incorporar outros recursos como lógicas para capturar os direitos e as obrigações como aspectos da política regras.

Com este novo conceito, surgirá uma nova categoria ou classe de regras RuleML na hierarquia que permitirá:

- Permissão e proibição regras

  Uma particular regra de atribuição de permissão de direitos exclusivos, com a seguinte forma:

  – Apenas um agente de tipo A possui o direito de efetuar ações de tipo a.

  Exemplos:

  – Gerentes estão autorizados a conduzir veículos para estação de serviço;

  – Outros funcionários estão proibidos de conduzir veículos para a estação de serviço;

- Dever e cessão regras

  Dever cessão regras com a seguinte forma:

  – Agentes do tipo A tem o dever de cumprir compromissos (ou reagir aos acontecimentos, ou para acompanhar créditos), de certo tipo.

  Exemplos:

  – Serviço de Apoio ao Cliente têm o dever de reagir as reservas pedidos;

  – Os taxistas têm o dever de cumprir os compromissos de fornecer veículos adequados aos clientes em locais especificados;

  – Os funcionários do departamento financeiro têm o dever de acompanhar reclamações contra os clientes para pagar as faturas dentro dos prazos;

RuleML inicialmente se propõe a construir uma Política RuleML Comitê Técnico para o uso de RuleML como uma forma de interoperar diferentes sistemas políticos, para isso define uma lista de prioridades de foco (http://policy.ruleml.org).

A meta inicial do Comitê Técnico foi estudar cenários de uso para desenvolver um intercâmbio RuleML como um veículo para a política das linguagens de regras, e normas para o intercâmbio adequado para o cenário.

A política do RuleML está prevista para trabalhar e colaborar com outras políticas e regulamentações.

Para isso o Comitê Técnico irá emitir um veículo de interoperação para a política de especificações.

2.3.3 MODELO M-FORUM

Baseado na análise de diversos ambientes colaborativos, os autores (CAMOLESI & MARTINS) desenvolveram um modelo e uma linguagem voltada para a especificação de regras de interações.

Os elementos envolvidos nas regras são denominados de *dimensões*, representando atores, atividades, objetos, tempo, espaço, além das associações entres eles.

Uma regra deve envolver pelo menos um ator e um objeto com a utilização de operadores específicos.

Atores humanos são representações de pessoas envolvidas, os atores não-humanos são “seres” virtuais interagindo em atividades que podem conter atores humanos.

2.3.3.1 HISTÓRICO

O modelo *M-Forum* apresentado se destina à análise e especificação dos requisitos para a modelagem de interações em ambientes colaborativos. Criado pelo Grupo de pesquisas em Ambientes Colaborativos (GAC - FACEN/UNIMEP) este modelo teve sua primeira versão finalizada em 2004 ao contemplar conceitos elementares da especificação de regras baseadas em eventos (CAMOLESI & MARTINS) .

Atualmente a versão *M-Forum* 1.5 (Figura 6) permite uma modelagem de semântica mais refinada, a qual originou a formalização da linguagem *L-Forum.*

A Linguagem *L-Forum* é uma linguagem formal para definição de regras que auxilia a governabilidade de um ambiente colaborativo.

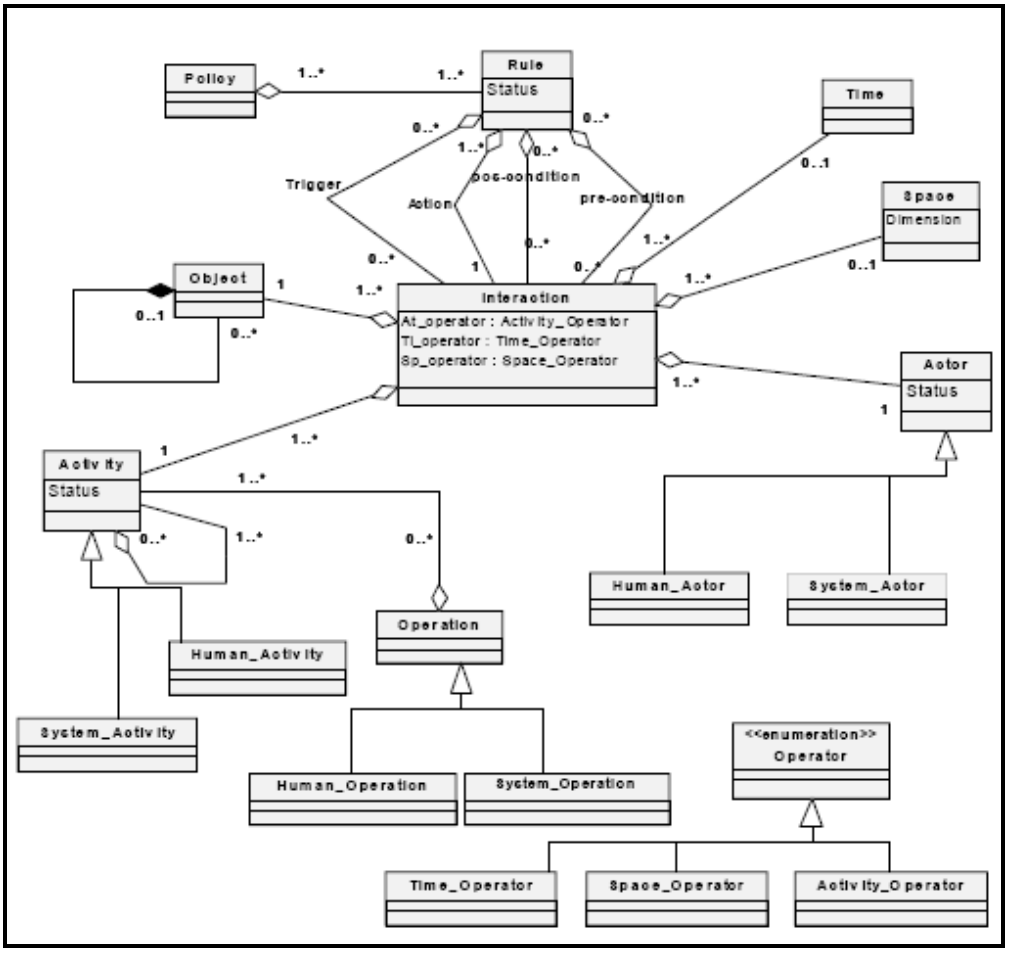

FIGURA 6 – MODELO M-FORUM (CAMOLESI & MARTINS, 2006)

2.3.3.2 CONSTRUÇÃO DE REGRAS

Regras são normas de orientação para o relacionamento entre os elementos do ambiente ou sistema. No estado de ativas, as regras estabelecem condições para a interação e as conseqüências destas interações para o usuário ou ambiente.

Os elementos envolvidos nas regras, no modelo *M-Forum,* são denominados de *dimensões*, representando atores, atividades, objetos, tempo, espaço, além das associações entres estes. Uma regra deve envolver pelo menos um ator e um objeto com a utilização de operadores específicos.

Um ator é um agente em um ambiente colaborativo, pode ser classificado em humano e não-humano. Tem um papel bem definido, conforme os direitos, proibições e obrigações de suas atividades. Atores são responsáveis pela execução de atividades, podendo assim, interagir com objetos, com outro ator ou grupo de atores.

Atores humanos são representações de pessoas envolvidas, os atores não-humanos são "seres" virtuais interagindo em atividades que podem conter atores humanos.

Todo ator tem identificador, um estado corrente e um conjunto de atributos (KAGAL et al, 1998). Considerando *qh* a quantidade de atores humanos e *qs* a quantidade de atores não humanos de um ambiente, temos:

$AchS = \{Ach_1, Ach_2, ..., Ach_{qh}\}$, $AcsS = \{Acs_1, Acs_2, ..., Acs_{qs}\}$
$AchS \neq \varnothing \vee AcsS \neq \varnothing, AchS \cap AcsS = \varnothing$
$AcSS = AchS \cup AcsS$
$Ach_i = (Ach\_id_i , AchState_i , Ach\_AttS_i)$
$Acs_i = (Acs\_id_i , AcsState_i , Acs\_AttS_i)$

Atividade é um elemento de execução que pode ser realizado por um ator ou grupo de atores, envolvem normalmente a manipulação ou transformação de um objeto. É composta por um identificador, um subconjunto de atividades, um subconjunto de operações e um conjunto de atributos.

As Atividades devem ser expressas em interações usando *Operadores de Atividade* que permitem a definição do direito, dever, dispensa ou proibição. Operadores de Atividade são requeridos para especificar a interação de uma atividade entre atores e objetos.

Considerando *qa* a quantidade de atividades de um ambiente, temos:

$AtSS = \{At_1 , At_2 , ..., At_{qa}\}$
$At_i = (At\_id_i , AtState_i\ AtS_i , OpS_i , At\_AttributeS_i)$
$AtS_i \subseteq AtSS , OpS_i \subseteq OpS$

Objetos representam elementos que constituem conceitos ou entidades do mundo real. Os objetos carregam consigo a representação das características

estruturais e do comportamento do que está representando.

Objetos podem ser compostos por outros objetos e caracterizam-se por um estado e um conjunto de atributos.

Uma modelagem de objetos estabelece uma uniformidade de visão e de tratamento, útil tanto para projetos quanto para implementação de ambientes colaborativos (KAGAL et al, 1998).

Atividades e Operações podem ser realizadas sobre Objetos podendo alterar suas características. Considerando *qo* a quantidade de objetos de um ambiente, temos:

$ObSS = \{Ob_1 , Ob_2 ,..., Ob_{qo} \} \vee ObSS \neq \varnothing$
$Ob_i = (Ob\_id_i , CompObS_i , ObState_i , Ob\_AttS_i)$
$CompObS_i \subset ObSS$

Espaço virtual permite o “armazenamento” ou localização atores e objetos, além das áreas específicas envolvidas em atividades e operações, sendo imprescindíveis para a modelagem de um ambiente colaborativo.

Considerando um espaço colaborativo em que *qe* é a quantidade de espaços, temos:

$Coordinate = \{(x, y) \mid x, y \in N\}$
$ELoc = (Fo1, Fo2, P)$
$Fo1, Fo2, P \in Coordinate$
$PLoc_i = (Sp\_id_i , SpState_i, \{Po1 , Po2 , .., Pon \})$
$Po1 , .. . Pon \in Coordinate$
$Ellipse = \{el \mid el \in Eloc \} , Polygon = \{po \mid po \in Ploc \}$
$SpSS = Ellipse \cup Polygon$
$SpSS = \{Sp_1 , Sp_2 , ..., Sp_{qe}\}$

Sobre aplicações de espaços geométricos, elementos da dimensão espaço devem ser expressos em interações usando o *Operador de Espaço* para a especificação de posição e tamanho de atores e objetos em ambientes colaborativos.

spop ∈ So = { <,<=,>,>=,=,<>,==(attribution), not equal, inside, outside, intersect, meet, overlap, north, south, east, west}

A formalização básica para o aspecto temporal pode ser baseada no conjunto de números naturais (*N*) para representar anos (*Ty*), meses (*Tm*), dias (*Td*), horas (*Th*), minutos (*Th*) e segundos (*Ts*) no *Tempo* e *Intervalo*.

Para *Datas,* conjuntos enumerados são usados para representar valores relativos (*Tmr, Tdr, Thr, Tmir, Tsr*) de um determinado calendário. Considerando *qt* a quantidade de intervalos ou momentos de tempo de um ambiente, temos:

Ty, Tm, Td, Th, Th, Ts $\in \Delta$
Tmr $\in$ {1, 2, 3, 4,..., 11, 12} , Tdr $\in$ {1, 2, 3, 4, 5,.., 31}
Thr $\in$ {0, 1, 2, 3,..., 24} , Tmir,Tsr, $\in$ {0, 1, 2, 3,.., 59}
Time = {Ti_id ,(Ty , Tm , Td , Th , Tmi , Ts)}
Datetime = {Ti_id ,(Ty , Tmr , Tdr , Thr , Tmir , Tsr)}
Tb, Te $\in$ Datetime , Interval = {Ti_id , (Tb, Te)}
TiSS = Time U Datetime U Interval
TiSS = {Ti_1 , Ti_2 , ..., Ti_{qt} }

A representação semântica do tempo permite a utilização precisa de *Operadores de tempo* para uso em expressões de interação em um ambiente. Na modelagem de tempo, *Durações*, *Datas* e *Ocorrências* têm semânticas fundamentais para estabelecimento de referências temporais.

Estas semânticas são usadas para definir a lógica temporal de operadores para tempo de duração, data de ocorrência ou intervalo de ocorrência de atividades e operações definidas em interações entre atores e objetos (MOK et al, 2002).

tiop $\in$ To = { <,<=,>,>=,=,<>,== (attribution), precedes, succeeds, directly precedes, directly succeeds, overlaps, is overlapped by, occurs during, contains, starts, is stated with, finishes, is finished with, coincides with }

Em situações especificas, pode-se ou deve-se estabelecer ou verificar associações entre atores, atores e objetos, atores e tempo e também entre atores e espaços.

Estas associações são *estáticas* se definidas no início das interações e permanecerem inalteradas até o final de um processo colaborativo, ou são *dinâmicas se* precisarem ser alteradas em ações permitidas.

Em processos a prática de abstração, busca um detalhamento maior dos

elementos de um ambiente e seus relacionamentos.

As abstrações de maior ocorrência em ambientes colaborativos são as classificações, generalizações e composições.

Políticas são normas para as interações em um ambiente, podendo ser estabelecidas por agrupamento lógico de regras com metas ou objetivos bem definidos.

A definição de política em sistemas pode ser definida de duas formas:

· Intra-regras: em que a política é definida internamente às regras de um ambiente, ou seja, na definição de uma regra podem existir referências a outras regras que possuam algum grau de interdependência e assim, as associações entre regras que caracterizam uma política estão estabelecidas internamente ao corpo das regras;

· Extra-regras: em que a política é definida externamente às regras e desta forma, todas as inter-relações entre regras que estabelecem uma composição para normalização de um ambiente são definidas em um elemento independente, no caso, podendo ser a própria Política.

2.3.3.3 LINGUAGEM L-FORUM

A Linguagem *L-Forum* é uma linguagem formal (APÊNDICE A) para definição de regras que auxilia a governabilidade de um ambiente colaborativo, podendo trazer diversos benefícios, entre os quais podemos citar:

· Orientar na definição de regras do ambiente;

· Aumentar o grau de precisão das regras, diminuindo a ambigüidade na interpretação pelos atores do ambiente;

· Facilitar a depuração de inconsistências;

· Permitir a flexibilização do ambiente com a retirada, modificação ou inclusão

de regras, permitindo que a Política de um ambiente possa ser definida ou adaptada;

```
Atores:         Bombeiro, Auxiliar, Morador
Objetos:        Balde, Objeto em chamas, Local do Incêndio, extintor, telefone
Grupo:          Corpo de Bombeiros, Residentes, Grupo de Objetos em Chama, Grupo
                de Baldes
Atividades:     Encher (inserir/em), Esvaziar (atirar/sobre), Solicitar, Identificar, colocar
Tempo:
Espaço:         Residência
Abstrações:     Bombeiro.compõe.Corpo de Bombeiros
                Auxiliar.compõe.Corpo de Bombeiros
                Balde.compõe.Grupo de Baldes
                Objeto em chamas.compõe.Grupo de objetos em chama
                Morador.compõe.Residentes
                Balde.possui.extintor
                Água is a extintor, Areia is a extintor

Rule            Solicitar Ajuda         [ACTIVE]
{
Parameters::    (m:Morador, cb:Corpo de Bombeiros, fire:Grupo de Objetos em Chama,
                l:Local do Incêndio, b:Bombeiro, ba:Balde, t:telefone, a: auxiliar)
Applicability:: (fire is on)
Body::          Execute Action [ 1 ] (m obligation solicita cb);
                Execute Action [ 2 ] (m obligation identifica l);
                Rule [ 3 ]              (Apagar Fogo    (b, m, a, ba, fire, e) right);
}

Rule            Apagar Fogo             [ACTIVE]
{
Parameters::    (b:Bombeiro, m: morador, a: auxiliar, ba: Balde, fire: Grupo de Objetos
                em Chama, e:extintor)
Applicability:: (fire is on) and (b is ready)
Body::          If ( fire.tipo = 'elétrico' )
                                Then { Rule  [ 1 ] Abastecer (m, a, ba, e:areia) }
                                Else  { Rule [ 1 ] Abastecer (m, a, ba, e:agua)  }
                Execute Action [    1 ] (b obligation encher.atirar ba.possui.extintor
                encher.sobre fire); //
                Execute Action [ // 1 ] (m prohibition encher.atirar ba.possui.extintor
                encher.sobre fire);
}

Rule            Abastecer               [ACTIVE]
{
Parameters::    (m:Morador, a:Auxiliar, ba:Balde, e:extintor)
Applicability:: (ba not is full)
Body::          Execute Action [ 1 ] (a obligation encher.extintor encher ba);
                Execute Action [ 1 ] (m right encher.extintor encher.ba);

}
```

FIGURA 7 – LINGUAGEM L-FORUM

A linguagem possui três conjuntos de cláusulas com propósitos bem específicos:

· *Contexto*: composto por *parâmetros* para a execução ou ativação de uma regra e pelas condições de aplicabilidade que estabelecem os cenários (valores de atributos, aspectos temporais ou espaciais) em que uma regra pode ser aplicada;

· *Definição (ou corpo)*: conjunto de expressões que estabelece as ações ou condições para as ações que interagem entre os elementos, podendo envolver opcionalmente aspectos de transitoriedade no tempo e no espaço;

· *Regime*: é um item opcional composto pelo conjunto de regras inter-relacionadas que tenham orientação para serem executadas ou aplicadas. Também podem ser definidos os cenários (valores de atributos, aspectos temporais ou espaciais) em que uma regra deve ser ativada ou desativada para uso;

2.4 POLÍTICA EM REGRAS DE COLABORAÇÃO

Os ambientes colaborativos de software têm sido desenvolvidos e o conjunto de regras possui extrema importância no contexto do ambiente, onde muitos usuários interagem de maneira assíncrona ou síncrona (ENGENHOFER & MARK, 1995). As regras de interação em um ambiente asseguram a confiabilidade e a flexibilidade do ambiente, sendo, portanto, fatores críticos de qualidade que devem ser aplicados para melhoria do desempenho dos usuários (TAKADA et al, 1999).

As interações entre atores estão entre as principais características de um ambiente colaborativo, devem, portanto ser eficazmente modeladas para orientar os direitos, proibições e obrigações nas atividades dos atores.

Na forma de regras, as interações devem estabelecer a ordem e restrições das atividades entre atores e entre atores e objetos, em determinado espaço e intervalo ou limite de tempo (KAGAL et al, 1998).

As políticas de regras (ECA-P) são escritas por um administrador em uma linguagem ECA que possui operadores e construtores similares, suportam eventos parametrizados, são compiladas e carregadas dentro do sistema de gerenciamento para administrar o espaço ativo. Nas compilações os conflitos detectados são sistematicamente listados para o usuário resolver.

O sistema de gerenciamento subscreve para eventos específicos dentro da política e inicia ações correspondentes quando aquele evento é disparado, determinando o conjunto de regras necessárias para serem executadas e as verificações de conflitos utilizando um algoritmo de análise dinâmico.

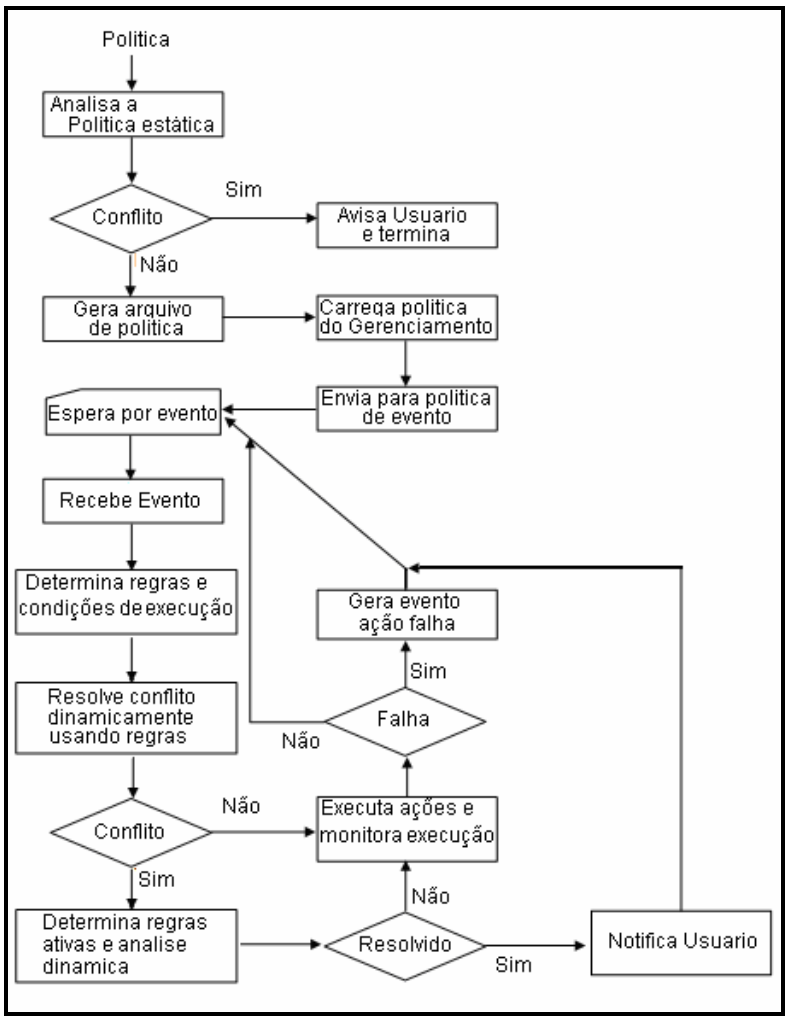

FIGURA 8 – POLÍTICA DE RESOLUÇÃO DE CONFLITOS DE REGRAS ECA (SHANKAR, 2005)

Com o serviço centralizado de resolução de conflito, para cada *rule-action* executada, é monitorado o término das ações relacionadas na regra e verificado suas pós-condições. Quando as pós-condições verificadas retornam falsas, é assumido que falhou a execução e enviado uma *action-failed* para que o próprio serviço resolva. Quando a regra estiver presente no gerenciamento de falha o serviço tomará a ação apropriada.

Na definição de política de colaboração (figura 8), temos que um conjunto de regras orienta uma atividade colaborativa para garantir essa política, Policy = (Rc_1, Rc_2, Rc_n), temos consistências e processos envolvendo os elementos do ambiente (atores, atividades, objetos, tempo e espaço), que são explorados pelas regras.

2.5 CONSISTÊNCIA DE REGRAS

Sistemas de computação podem conter um grande número de diversidades como dispositivos e aplicações móveis. Um caminho comumente utilizado para gerenciar esses ambientes complexos é o uso de políticas.

Políticas podem ser usadas para gerenciar diferentes aspectos de um sistema como a Ausência de Configuração, Monitoramento, Performance e Gerenciamento de Segurança.

Políticas de obrigação são especificadas com regras de Evento-Condição-Ação (ECA). Essas regras especificam as ações a serem executadas, quando ocorrer um evento e quais condições devem ser satisfeitas para a sua execução.

A política baseada em fluxograma apresenta duas limitações, primeiramente a rigidez dos caminhos de detecção e resolução de conflitos das regras e em segundo, não possui mecanismos para garantir que as políticas são executadas corretamente (HOLZMANN, 1997).

2.5.1 CONCEITOS

Duas regras são consideradas conflitantes, se sobre o mesmo evento-condição possuem pós-condições ambíguas. Mecanismos de detecção de conflitos definem regras conflitantes como as que têm ações conflitantes para o mesmo evento-condição.

A política de conflito de regras EAC surgiu para tratar um subconjunto de evento-condição ativo que possua mais de uma ação eleita para execução que conflitam entre sí.

O conflito surge quando há necessidade de escolha entre situações que podem ser consideradas incompatíveis. Todas as situações de conflito são antagônicas e perturbam a ação ou a tomada de decisão por parte da pessoa ou de grupos.

As situações de conflito podem ser resultado da concorrência de respostas incompatíveis, ou seja, um choque de motivos, ou informações desencontradas.

Kurt Lewin (Lewin, 1936) define o conflito "*como a convergência de forças de igual intensidade e sentidos opostos, que surgem quando existe atração por duas valências, positivas mas opostas (desejo de assistir a uma peça e a um filme exibidos no mesmo horário e em locais diferentes) ou duas valências negativas (enfrentar uma operação ou ter o estado de saúde agravado) ou uma positiva e outra negativa, ambas na mesma direção (desejo de pedir aumento e medo de ser despedido por isso)*."

2.5.2 CLASSIFICAÇÃO DE TIPOS DE CONFLITO

Para classificar conflito de regras ECA, Shankar (SHANKAR e outros, 2005) estabelece que o conjunto de conflitos detectado usando restrição de ação é um subconjunto do conjunto de conflitos detectados usando uma equivalente restrição de pós-condições. Como exemplo:

- Inconsistência de ação: onde uma determinada regra permite ao ator realizar uma ação sobre um determinado objeto e outra regra o proíbe, ambas as regras se aplicam no mesmo tempo e espaço.

- Acesso pessimista ao objeto: o controle da concorrência advoga a verificação e a resolução de conflito pelo sistema de forma a anular a possibilidade de mais de uma regra interagir com o objeto no mesmo tempo e espaço, o acesso é gerido de forma a que apenas uma regra atue (locking) no momento para o grupo de atores.

- Inconsistência de atributo valorado: onde uma determinada regra limita a realização da ação valorizando o gatilho em Valor1 e uma segunda regra limita a realização da mesma ação valorizando em Valor2.

- Duplicidade de regras: quando mais de uma regra atua de forma idêntica nos mesmos atores, objetos e condições de execução; apesar de não representar um problema ela despenderá de um maior controle na manutenção, já que o código deverá ser alterado mais de uma vez possibilitando ferir alguma consistência relatada anteriormente, caso uma manutenção não seja replicada para a regra comum.

2.5.3 SOLUÇÃO DOS AUTORES SHANKAR, RANGANATHAN E CAMPBELL

A política de conflito de regras (SHANKAR et al, 2005) escreve um subconjunto de regras para serem acionadas em caso de uma pós-condição conflitante.

O fator positivo dessa técnica é o desenvolvimento incremental utilizando regras simples e de fácil alteração que quando se chocam recebem um tratamento corretivo. O fator negativo é a pulverização do conhecimento das regras do negócio, sendo necessário uma varredura das regras pré e pós-execução para o entendimento do negócio.

A ação é chamada para um método em uma biblioteca de ações onde cada ação é anotada com a pós-condição para o analista principal analisá-la e propor uma política de regras ECA com evento – condição – ação – pós-

condição (ECA-P), embora as pós-condições não são especificadas como parte das regras.

Para os autores uma forma de tratar os conflitos das regras é através de uma política de conflito tratando as condições e pós-condições de cada regra com suas regras relacionadas. Para isso, propõem a adição da pós-condição como uma extensão da regra.

Existem três simbologias: símbolos de eventos primitivos, símbolos de ação e símbolos constantes. Símbolos de eventos primitivos representam eventos básicos que podem ser descritos no sistema, por exemplo, *ObjectEnter* e *ObjectExit.*

Cada símbolo de ação denota o nome de uma *procedure* que pode ser invocada no sistema, uma ação é descrita por proc(t_1, ...t_n) onde proc é o nome da procedure e t_is são os parâmetros.

```
R1: on(ObjectEnter(Person p))
        if (not_equal(role(p), "owner"))
        do (start(authorization_app))
        { status(authorization_app, running)}

R2: on(ObjectEnter("Tom"))
        if (true)
        do (stop_apps())
        { status(log_app, stopped),
        status(authorization_app, stopped)}

R3: on(ObjectExit(Person p))
        if (equal(role(p), "owner"))
        do (start(log_app))
        { status(log_app, running)}

R4: on(ObjectExit(Person p))
        if (equal(person_count(),0))      //space is empty
        do (suspend())                    //suspend space
        { status(log_app, stopped),
        status(authorization_app, stopped)}
```

FIGURA 9 – LINGUAGEM DE POLÍTICA DE CONFLITOS DE REGRAS ECA (SHANKAR ET AL, 2005)

Um gerênciamento típico de regras usado para tratar o *active space* é mostrado na figura 10.

A linguagem de política usa termos e métodos definidos para serviços no *active space*.

O *ObjectEnter(Person p)* é um termo que representa um evento que é disparado para o sistema de locação quando a pessoa entra no *active space, person* é um tipo de dado definido pelo usuário que recebe valores de “nome”, essa política especifica ações para serem executadas quando uma pessoa entra ou existe em um *active space*.

A regra R1 inicia uma aplicação de autorização para os usuários, a regra R2 paralisa todas as aplicações em execução se o usuário de nome “Tom” entra no *active space.*

Quando o criador do *active space* está presente à regra R3 recebe os gatilhos de ação e verificam os ponteiros de todas as aplicações. A regra R4 suspende o *active space* caso não exista pessoas ativas.

As regras R1 e R2 são conflitantes quando a pessoa nomeada ”Tom”, que não é o criador do *active space,* entra no espaço acionando a regra R1 para iniciar a aplicação de autorização, enquanto a R2 a paralisa.

A política direciona o conflito para a aplicação do ponteiro iniciar o uso da regra R3, enquanto esta, é implicitamente paralisada pela regra R4.

A técnica de detecção de conflitos estáticos sobre regras cujos eventos e condições podem ser estaticamente confrontados.

Para determinar se dois eventos são conflitantes os autores comparam seus símbolos de eventos e tipos, e os valores dos parâmetros.

Portanto, na política da figura 9, o termo da regra R1 *(ObjectEnter(Person p))* confronta com o termo da regra R2, *(ObjectEnter(“Tom”)),* este é um exemplo de um processo de chamada unificada usado em verificação de tipos em compiladores.

```
// initialize
PC – post-condition constraint set
P - the Policy
event(r) – event of rule r
condition(r) – condition of rule r
id(r) – rule identification number of rule r
post(r) – post-condition of rule r

// statically detect conflicts for each rule
for each rule r in P
        e := event(r)
        c := condition(r)
        K = { }
        R = { }          // rule id set
        for each rule s in P and s ≠ r
                if (match(event(s),e)&match(condition(s), c))
                        K = K U {post(s)}
                        R = R U {id(s)}
                end if
        end for
        K = K U {post(r)}
        if (K U PC) is not consistent
                print(R)
        end if
end for
```

FIGURA 10 – ALGORITMO1 DE DETECÇÃO DE CONFLITO ESTÁTICO (SHANKAR ET AL, 2005)

O algoritmo1 lista um conjunto de regras conflitantes, para isso é iniciado com o conjunto de critérios de pós-condição PC, e a política de regras especificada como ECA-P.

Cada regra r é verificada contra outras regras na política. Se um evento da regra r conflita com outro da regra s, a pós-condição da regra s é adicionada ao conjunto K e a regra de identificação única registra a regra conflitante na política adicionando ao conjunto.

Desde que todas as regras tenham sido validadas, a pós-condição da regra r é adicionada no conjunto K. A união de K e PC é verificada para consistência.

Se mais de um predicado é verificado como verdadeiro para uma consistência, isso implica que a consistência existente foi violada e a união de K e PC é considerada inconsistente.

Aplicando este algoritmo para a política da figura 10 com consistência de pós-condição, é detectado que as regras R1 e R2 estão em conflito, desde então a aplicação de autorização de execução não pode ser simultânea no processamento de *running* e *stopped*.

P1: $\neg$ (status (App, running) $\wedge$ status (App, stopped))

Uma resolução de regra m é especificada como um simples comando if→then:
if {condition} **then** {post-condition}

Isto é lido como "Se o sistema está com estado representado por condition, então o sistema é escolhido para atingir o estado representado pela pós-condição".
A resolução do algoritmo representado abaixo escolhe a ação correspondente para escolher a pós-condição.

M1: *if (event (ObjectExit)* $\wedge$
can_reach (status (log_app, running)) $\wedge$
can_reach (status (log_app, stopped)) then
can_reach (status (log_app, running))

Para a resolução da regra, M1 implica que se *ObjectEx*it ocorre e a aplicação *log_app* pode retornar estados *running* e *stopped*, para a execução do conflito de regras a escolha da regra correspondente é a de pós-condição *can_reach(status(log_app, running)).*

2.5.4 SOLUÇÃO DOS AUTORES PLACCA, GARCIA & DUTRA

Mapear o fluxo de trabalho dos diversos especialistas em um projeto e as fases na qual passa um produto durante seu desenvolvimento foi a abordagem do modelo proposto pelos autores. A ocorrência de conflitos se dá nas diversas tarefas e fases citadas e uma única estratégia de resolução não retorna resultados satisfatórios por deixar de explorar características inerentes ao tipo de conflito detectado.

A proposta do modelo Tri-Coord parte da premissa de que não existe uma estratégia de resolução de conflitos que seja adequada a qualquer tipo de Sistema Multiagente, porém os conflitos detectados podem ser classificados e a utilização de uma estratégia específica em função do tipo de conflito pode conduzir à resultados mais satisfatórios.

O modelo Tri-Coord+ complementa o anterior no tratamento diferenciado em função do ciclo de desenvolvimento de um projeto e a re-utilização de casos e estratégias utilizadas no mesmo, para otimizar as interações entre agentes, foram focadas técnicas de SMA Cognitiva, voltadas para ambientes de elevado grau de interação, cujos agentes tenham elevado grau de conhecimento individual.

Um sistema multiagente pode ser visto como um grupo de entidades independentes, interagindo entre sí com o intuito de alcançar um objetivo comum. Devido ao conhecimento limitado dos agentes, escassez de recursos e fatores inerentes ao domínio da aplicação, a ocorrência de conflitos entre os agentes é inevitável. Dessa maneira a existência de um mecanismo eficiente de detecção e resolução de conflitos é vital para o bom desempenho do SMA.

Uma tarefa de *groupware*, raramente tem uma evolução seqüencial até sua conclusão final. Em geral, durante as etapas do projeto, alternativas são discutidas e testadas e muitas vezes, especificações são refeitas em função de novos requisitos do projeto.

Dessa maneira, uma solução para um eventual conflito detectado e resolvido numa fase inicial do projeto poderá ter que ser revista numa fase posterior. Para diminuir os conflitos foi sugerida a atualização dinâmica da base de conhecimentos, possibilitando que soluções adotadas na fase inicial de um projeto possam ser aproveitadas como regra em situações de conflito similares em fases posteriores.

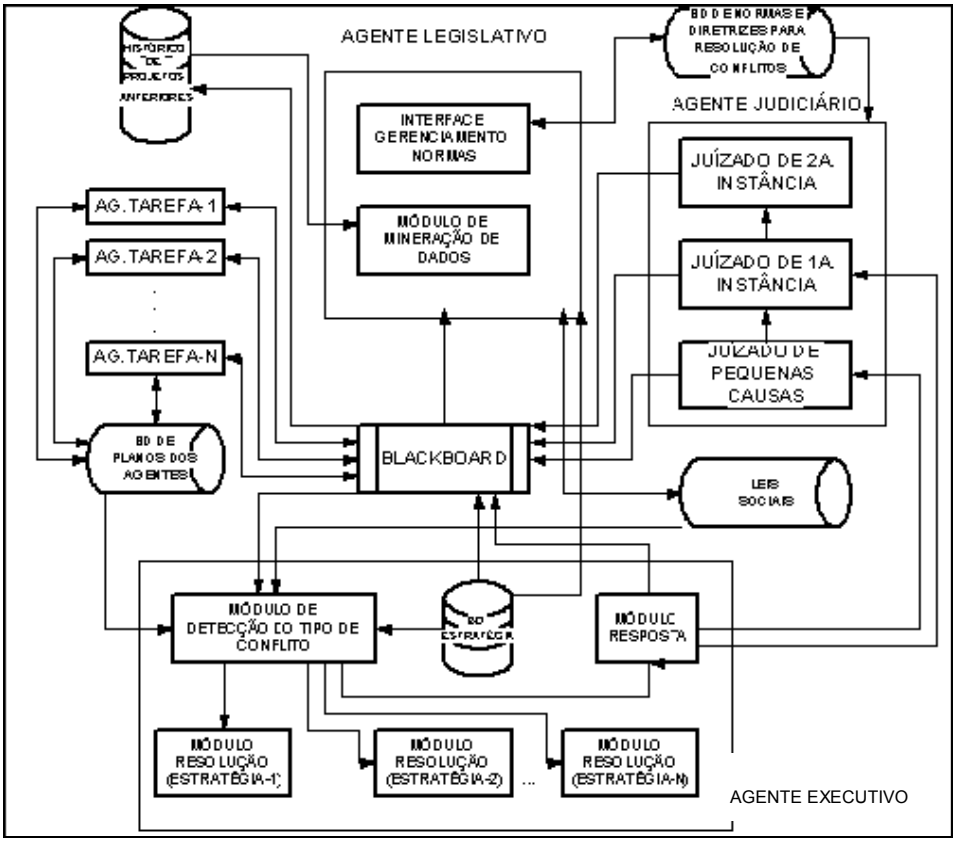

FIGURA 11 – MODELO TRI-COORD+ (PLACCA ET AL, 2003)

O Modelo Tri-Coord+ mantém uma estrutura de *blackboard* pela qual a informação é acessada ou modificada, porém os agentes de tarefas também podem se comunicar entre sí e resolver conflitos sem a necessidade da intervenção de um agente especial. Esse modelo é constituído por quatro tipos de agentes: agentes de tarefas, Agente Executivo, Agente Legislativo e Agente Judiciário.

O Agente de Tarefas é o agente computacional que atua no ambiente, em nome de um usuário e com um objetivo específico.

O Agente Executivo é o responsável pelo gerenciamento imediato dos conflitos e a comunicação de sanções ou decisões aos agentes, sua constituição compreende:

a) Módulo de Detecção: responsável por identificar o tipo de conflito ocorrido e escolher a estratégia de resolução mais apropriada;

b) Módulo de Resolução: implementa as diversas estratégias contempladas no Banco de Dados de Estratégias de Resolução de Conflitos. A cada nova estratégia adicionada ao Banco de Dados, o respectivo Módulo de Resolução deverá ser disponibilizado. Dentre as estratégias que poderão ser contempladas estão: Mecanismos de Votação, Eleição, Leilão, etc.

c) Módulo de Resposta: responsável pela emissão da resposta gerada.

d) Banco de Dados de Estratégias de Resolução e Tipos de Conflitos: armazena as estratégias de resolução pertinentes em função do tipo de conflito, contém para cada tipo de conflito detectado uma lista com as possíveis estratégias de resolução mais adequadas (em ordem de prioridade) e que possibilita ao Agente Executivo executar o procedimento mais adequado para aquela situação.

O Agente Judiciário é o responsável em dirimir conflitos pendentes que não puderem ser resolvidos pelo Agente Executivo. Utiliza-se, em geral, um processo de audiência onde todos os agentes envolvidos são convocados para uma comunicação síncrona, a fim de dirimir o conflito.

O Agente Legislativo é o responsável pela manutenção e renovação do conjunto de regras utilizadas no ambiente. Está constantemente observando o ambiente para incorporar regras que atendam uma necessidade de demanda do momento, sua constituição compreende:

a) Módulo de Interface para permitir a interação manual através de um Agente

Humano;

b) Módulo de Mineração de Dados para permitir a inferência de novas regras baseadas no Histórico de Projetos anteriores, contém as informações relativas à parâmetros e especificações de projetos já realizados. O *Blackboard* é um banco de dados onde cada tipo de agente tem uma visão que compreende o estado dos agentes e suas respectivas ações. Sua função principal é sinalizar a ocorrência de conflitos.

A utilização de estruturas sociais como estratégia para o gerênciamento de sistemas multiagentes tem sido uma técnica largamente utilizada em aplicações baseadas em SMAs (JENNINGS e CAMPOS, 1998). Uma outra corrente bastante difundida para o problema de detecção e resolução de conflitos em SMAs é a escolha da técnica apropriada em função do tipo de conflito detectado (LIU e outros, 1998).

Aplicações de SMAs em Projetos de Engenharia Concorrente, casos típicos de tarefas de *groupware* ou CSCW aplicados à Engenharia são exemplos de classes de aplicações onde a detecção e resolução de conflitos é essencial para o sucesso da tarefa-meta.

O problema enfocado pelos autores diz respeito ao ambiente de interação entre agentes e propõe um modelo para tratamento das interações entre agentes cooperativos visando proporcionar uma maior autonomia dos agentes bem como uma otimização no tratamento de eventuais conflitos durante a execução de um trabalho cooperativo entre múltiplos agentes.

Sua proposta consiste na extensão do Modelo Tri-Coord, que é um modelo de auxílio à resolução de conflitos em ambientes fechados, com múltiplos agentes, isto é, ambientes onde as leis ou regras de interação, comportamento e atuação estão bem definidas no ambiente. O referido modelo é baseado na aplicação de Leis Sociais (SHOHAM e TENNENHOLTZ, 1996) e foi inspirado na teoria do Contrato Social de Jean Jaques Rosseau.

2.6 CONSIDERAÇÕES FINAIS

Validar uma especificação funcional ou atributos de um sistema é um desafio na área de desenvolvimento, principalmente quando é envolvido um grupo de pessoas que podem executar uma única atividade.

Através do levantamento bibliográfico foi possível verificar que não existe ainda uma estratégia de resolução de conflitos que seja adequada a qualquer tipo de Sistema Multiagente, porém os conflitos detectados podem ser classificados e a utilização de uma estratégia específica em função do tipo de conflito pode conduzir à resultados mais satisfatórios.

3. O PROCESSO DE VERIFICAÇÃO DE CONFLITOS EM REGRAS

3.1 CONSIDERAÇÕES INICIAIS

Durante muito tempo, os executivos administraram as empresas por meio de erros e acertos. Com a evolução da coordenação administrativa as técnicas de "como fazer", passaram a se basear em normas, técnicas e regras, ocasionando uma mudança de paradigma com "o que fazer".

Neste contexto, a coordenação pode ser definida como "o gerenciamento de dependências entre atividades e o suporte de (inter) dependências entre atores" (BORDEAU e WASSON, 1997).

O Modelo Tri-Coord+ (PLACCA et al, 2003) propõe uma estrutura de *blackboard* pela qual a informação é acessada ou modificada, porém os agentes de tarefas também possam comunicar entre sí e resolver conflitos sem a necessidade da intervenção de um agente especial.

Sankar (SANKAR et al, 2005) propõe um algoritmo para listar as regras conflitantes através da comparação de seus símbolos de eventos e tipos e os valores dos parâmetros. Com o subconjunto de regras conflitantes Sankar propõe a criação de regras de pós-condições para cada regra relacionada como uma extensão da regra.

O processo de verificação de conflito de regras dessa dissertação tem como objetivo confrontar regras nas suas dimensões (atores, atividades, objetos, tempo, espaço) do modelo M-Fórum (CAMOLESI & MARTINS, 2006), por entender como sendo o mais completo entre os pesquisados.

Através do processo de verificação é possível identificar possíveis pontos de conflitos e direcionar a análise do testador ao ponto em questão, facilitando o desenvolvimento e melhoramento contínuo dessas regras.

3.2 O Processo e os Tipos de Verificação

Utilizando a linguagem L-Forum um ou mais especificadores escrevem as regras nas suas dimensões (atores, atividades, objetos, tempo, espaço).

No momento da verificação é montado um diagrama das intra-regras, depois, são criadas as tabelas para facilitar o processo visual do testador.

Dentre as possíveis verificações temos:

- Garantir que uma regra envolva pelo menos um ator e um objeto, ou outro ator;
- Analisar se duas ou mais atividades envolvem um mesmo objeto;
- Analisar quando duas ou mais atividades são executadas por um mesmo ator;
- Analisar se as ações dos atores nos objetos ou no tempo são seqüenciais ou paralelas;
- Analisar a existência de aplicabilidade repetida dentro de uma regra;
- Analisar o número de parâmetros utilizados internamente na regra versus os relacionados como esperados por ela;
- Visualizar um conjunto de regras que possuam a mesma aplicabilidade e também a associação entre as aplicabilidades durante a chamada de uma regra, que em sua estrutura não conste a aplicabilidade da qual foi chamada (regra interna);
- Analisar a possibilidade de unificar as aplicabilidades comuns em uma regra, reduzindo as execuções parciais.

3.3 TÉCNICAS E MODELOS DE APOIO AO PROCESSO

Com a utilização de modelos de apoio (tabelas tridimensionais e diagramas) a técnica permite mapear se duas ou mais atividades envolvem um mesmo objeto, se duas ou mais atividades são executadas por um mesmo ator e se as ações dos atores nos objetos são seqüenciais ou paralelas.

O conjunto de modelos gerados pretende facilitar a verificação das regras por um testador, além de auxiliar na definição da política de regras e sua evolução.

Para as associações, a técnica possibilita analisar sua criação e a possibilidade de remoção quando essa está em uso por uma regra.

3.3.1 DIAGRAMAS

Diagramas são utilizados na técnica de verificação de conflitos como uma forma de representar as regras.

Esse trabalho define e adota uma simbologia simples que possibilita retratar os elementos do modelo M-Fórum (CAMOLESI & MARTINS, 2006).

Para representar um grupo (objetos, atores...) presente na regra, é utilizada a sobreposição parcial do desenho do elemento, como é apresentado na figura 14 na regra R1, o desenho do elemento ator ‘cb’ (Bombeiro) sobreposto representa o grupo de ator ‘Grupo de Bombeiros’.

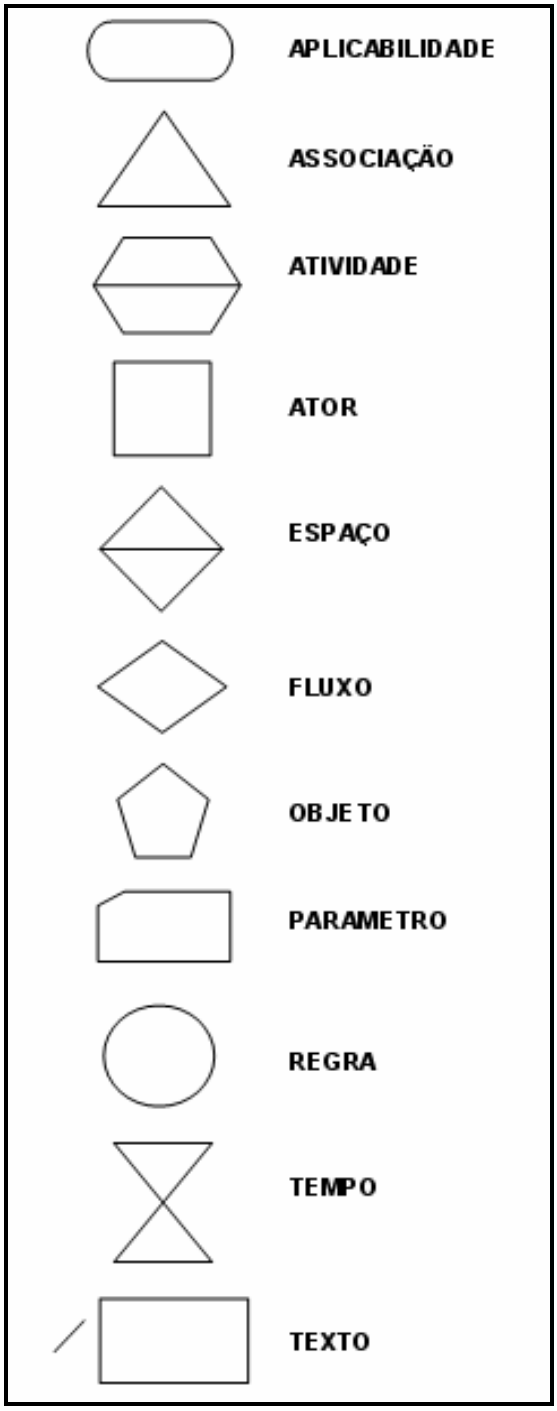

FIGURA 12 – ELEMENTOS DO DIAGRAMA DE VERIFICAÇÃO

Quando existe uma seqüência lógica de ações dentro de uma regra, uma inter-regra, a seqüência do processo é demonstrada graficamente através de um índice numérico (1,2,3...), na figura 13 é apresentada a Regra R1 com a

aplicabilidade ‘Fire is ON’, onde é demonstrada a seqüência de execução de ações no corpo dessa regra.

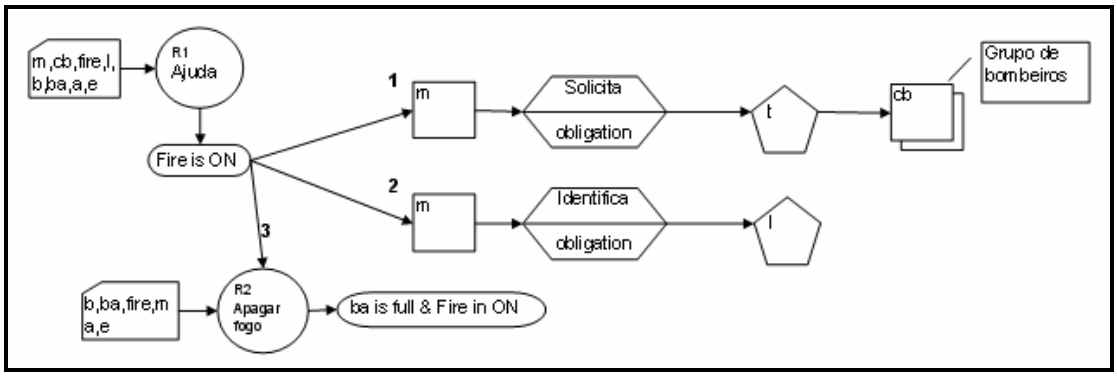

FIGURA 13 – DIAGRAMA DA REGRA 1

Durante a confecção do diagrama é possível verificar se todos os atores e objetos retratados na regra estão presentes no parâmetro.

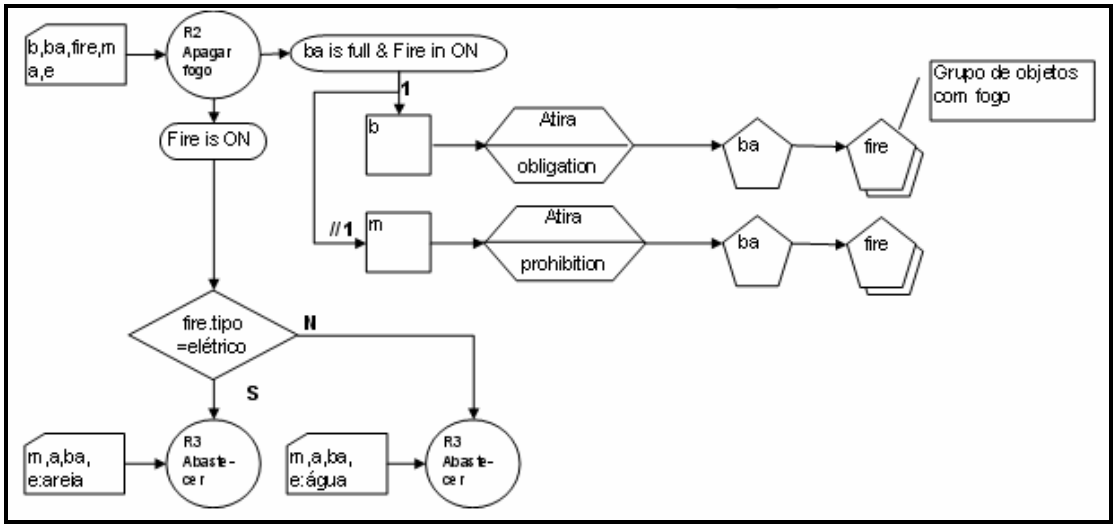

FIGURA 14 – DIAGRAMA DA REGRA 2

Através do diagrama da regra R2 apresentado na figura 14 é possível verificar mais de uma aplicabilidade para uma regra. Cada aplicabilidade retrata a estrutura de um bloco de atividades com execução condicionada, onde internamente podem existir desvios do fluxo de execução.

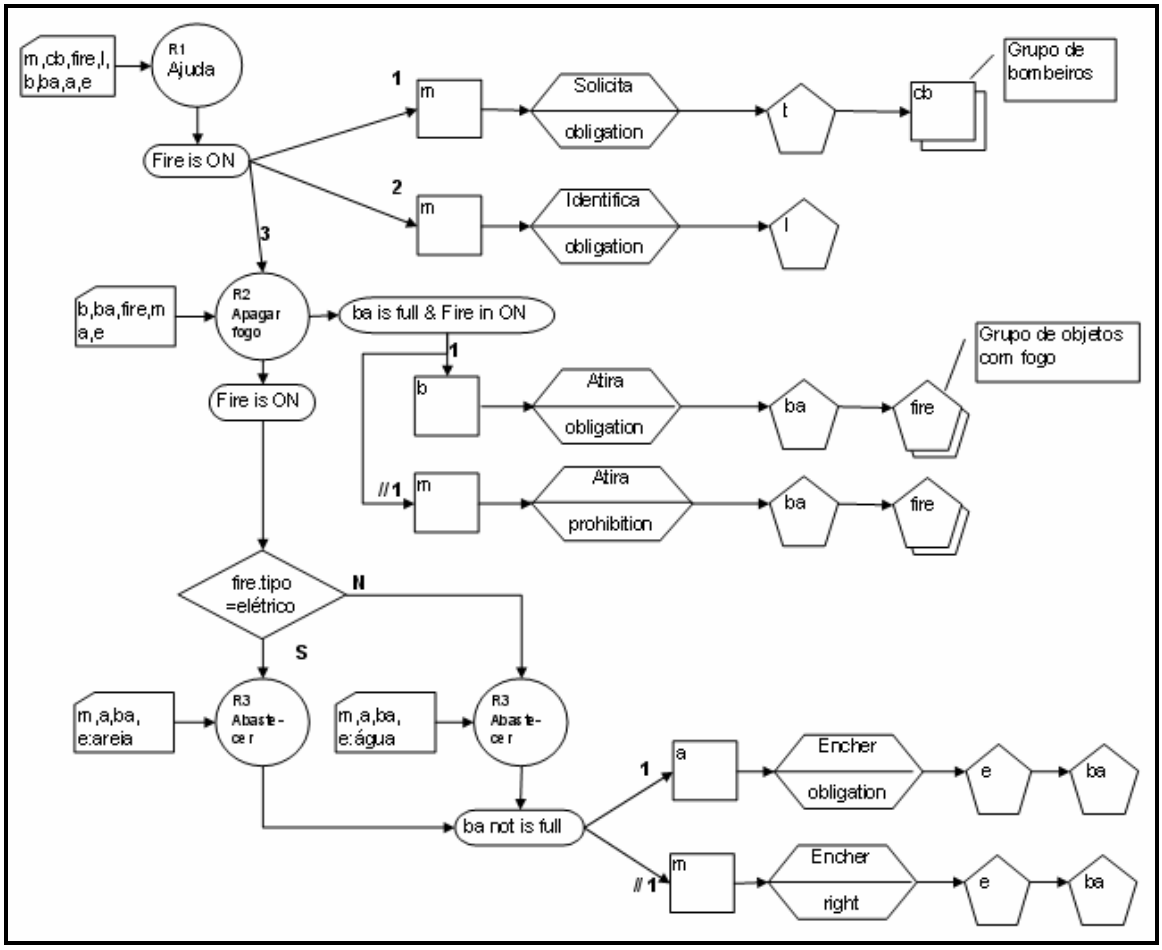

FIGURA 15 – DIAGRAMA DAS REGRAS

Nem sempre é possível a visualização de todas as regras de um projeto (figura 7) em um único diagrama como apresentado na figura 15.

Para facilitar a análise o método propõe a exploração individual das regras, com ressalva para um agrupamento de regras onde o método sugere a verificação decrescente das regras para facilitar a verificação dos parâmetros.

Os atributos das inter-regras devem estar presentes no parâmetro desde a primeira regra e é comum o esquecimento porque que geralmente, o analista escreve uma única regra por vez, porém essa pode estar relacionada a outras.

3.3.2 TABELAS TRIDIMENSIONAIS

As tabelas tridimensionais são utilizadas para identificar as dimensões de Atores e Objetos com as regras e sua aplicabilidade. Com base no diagrama de regras da figura 15, estão representadas abaixo as tabelas MAAR (tabela Ator x Ator x Regra), MAOR (tabela Ator x Objeto x Regra) e MAR (tabela Aplicabilidade x Regra):

- MAAR (Tabela Ator x Ator x Regra);

	Ator		
Ator	Auxiliar	Bombeiro	Morador
Auxiliar	**R3;**	**0**	**0**
Bombeiro	**0**	**R2;**	**R1;**
Morador	**0**	**R1;**	**R1; R2; R3;**

TABELA 1: MAAR – TABELA ATOR X ATOR X REGRA

A tabela MAAR expõe as atividades obrigatórias da dimensão Ator presentes nas regras e as registra para uma visualização rápida da colaboração existente no conjunto de regras.

Um exemplo da tabela 1 é a Regra R3 onde o Ator 'Morador' tem o registro da regra apenas quando nos eixos da tabela existe o Ator 'Auxiliar'.

- MAOR (Tabela Ator x Objeto x Regra);

	Objeto				
Ator	Água	Areia	Balde	Local	Objeto em Chamas
Auxiliar	**R3;**	**R3;**	**R3;**	**R3;**	**R3;**
Bombeiro	**0**	**0**	**R2;**	**R1;R2;**	**R2;**
Morador	**R3;**	**R3;**	**R3;**	**R1;R3;**	**R1;R3;**

TABELA 2: MAOR – TABELA ATOR X OBJETO X REGRA

A tabela MAOR registra a interação das dimensões Ator e Objeto com as regras e possibilita facilmente visualizar um conjunto de regras durante a verificação baseada na colaboração entre Atores e Objetos.

Quanto maior for o registro de regras na intersecção dos eixos da tabela, maior será a necessidade de atenção do testador durante a verificação do conjunto.

- MAR (Tabela Aplicabilidade x Regra);

Aplicabilidade	**Regra Interna**		**Regra Chamada**	
Fire is on	**R1**	**R2**	R1.call.**R2**	R2.call.**R3**
Fire is on & ba is full	**R2**			
ba not is full	**R3**			

TABELA 3: MAR – TABELA APLICABILIDADE X REGRA

A tabela MAR apresenta a interação das regras e possibilita facilmente visualizar um conjunto de regras que possuem uma mesma aplicabilidade, também a associação entre as aplicabilidades durante a chamada de uma regra que em sua estrutura não consta a aplicabilidade da qual foi chamada (regra interna). Através da tabela MAR, é possível verificar a possibilidade de unificar as aplicabilidades comuns em uma regra, como exemplo a regra R2, que contém duas aplicabilidades.

3.3.3 TABELA QUADRIMENSIONAL

A tabela quadrimensional é utilizada para identificar as dimensões de Atividades, Atores e Objetos com as regras. Com base no diagrama de regras da figura 15, está representada abaixo a tabela MARAO (Tabela Atividade x Regra x Ator x Objeto):

		Objeto					
Regra	**Ator**	Água	Areia	Balde	Local	Objeto em Chamas	Telefone
R1	Morador				**Identifica::o**		**Solicita::o**
R2	Bombeiro			**Atira::o**		**Atira::o**	
R2	Morador			**Atira::p**		**Atira::p**	
R3	Auxiliar	**Encher::o**	**Encher::o**	**Encher::o**			
R3	Morador	**Encher::r**	**Encher::r**	**Encher::r**			

TABELA 4: MARAO – TABELA ATIVIDADE X REGRA X ATOR X OBJETO

A tabela MARAO se distingue das demais, devido à inclusão da dimensão atividade que é qualificada com os operadores normativos (‘right’(::r), ‘prohibition’(::p), ’obligation’(::o), ‘dispensation’(::d)) opcionalmente tingindo a atividade em cores diferentes (‘amarelo’, ‘vermelho’, ‘verde’, ‘azul’)

3.4 VERIFICAÇÃO INTRA-REGRAS

Para verificar interações entre as dimensões dentro do corpo de uma única regra a técnica utiliza modelos de apoio (tabelas tridimensionais, quadrimensional e diagramas) mapeando as interações entre as dimensões presentes.

Com base no conjunto de regras do escopo, escrito em linguagem L-Fórum (CAMOLESI & MARTINS, 2006), foi desenhado o diagrama das regras (figura 15) e posteriormente montadas as tabelas.

Dentre as possíveis verificações nessa etapa temos:

- Garantir que uma regra envolva pelo menos um ator e um objeto, ou outro ator: a consistência é realizada utilizando o diagrama (figura 13) para visualizar se cada ação da regra possui os elementos (figura 12) ator e objeto descritos;

- Mapear se duas ou mais atividades envolvem um mesmo objeto: o mapeamento é feito através da tabela MARAO, onde além de relacionar as atividades que envolvem um objeto é representado os operadores normativos (‘right’(::r), ‘prohibition’(::p), ’obligation’(::o), ‘dispensation’(::d)) que regem a atividade;

- Mapear se duas ou mais atividades são executadas por um mesmo ator: o mapeamento é feito através da tabela MARAO, onde além de relacionar as atividades que envolvem um ator é representado os operadores normativos (‘right’(::r), ‘prohibition’(::p), ’obligation’(::o), ‘dispensation’(::d)) que regem a atividade;

- Se as ações dos atores nos objetos ou no tempo são seqüenciais ou paralelas: normalmente no processo colaborativo as ações são paralelas, quando a política necessita ordenar um grupo de atividades podemos verificar a seqüência de ações na regra utilizada no modelo de diagrama (figura 15) que utiliza um índice nas inter-regras.

- Se existe aplicabilidade repetida dentro de uma regra: através do modelo de apoio do diagrama podemos verificar a repetição das aplicabilidades de uma regra, caso isso ocorra devemos excluir uma das aplicabilidades e inserir as atividades da aplicabilidade excluída em um único corpo de regra, unificando as atividades e analisando seu impacto ou ratificar a aplicabilidade.

- Verificação dos parâmetros: analisando as atividades de uma regra no diagrama é possível verificar se todos os parâmetros utilizados nas atividades estão retratados no parâmetro da regra.

3.4.1 ETAPAS DO SUB-PROCESSO

Para verificar interações entre as dimensões dentro do corpo de uma regra a técnica utiliza modelos de apoio (tabelas tridimensionais, quadrimensional e diagramas) mapeando as interações entre as dimensões presentes. As etapas do sub-processo foram divididas em 5 passos:

- Passo 1: iniciar o diagrama de regras documentando as dimensões do conjunto de regras (Atores, Objetos, Atividades, Tempo e Espaço), os Agrupamentos e as Abstrações com base no escopo descrito em linguagem L-Forum, também os analistas da verificação e a data.

- Passo 2: desenhar uma a uma as regras e suas interações no diagrama. No desenho das atividades, devem-se enfatizar as dimensões ator e objeto, depois de desenhadas as atividades, completar a caixa dos parâmetros com as dimensões e registrá-las no grafo, verificando se são as mesmas descritas inicialmente na regra. O mapeamento da aplicabilidade deve possuir sempre um atributo valorado. Nesse passo é possível identificar:

    1. Se todos os atributos presentes na regra foram passados no parâmetro;

    2. Se existe aplicabilidade repetida dentro da regra;

3. A seqüência de execução, quando aplicável.

- Passo 3: identificar um possível encadeamento de regras no diagrama, onde no corpo de uma regra possui uma chamada à uma outra regra, e se assim for, verificar a existência dos atributos nos parâmetros da última para a primeira regra.

- Passo 4: montar as tabelas MAAR (Tabela Ator x Ator x Regra) e MAOR (Tabela Ator x Objeto x Regra), nesse passo é possível:

  1. Garantir que uma regra envolva pelo menos um ator e um objeto, ou outro ator;

- Passo 5: montar a tabela MARAO (Tabela Atividade x Regra x Ator x Objeto), nesse passo é possível:

  1. Mapear se duas ou mais atividades envolvem um mesmo objeto e identificar se os operadores normativos (‘right’(::r), ‘prohibition’(::p), ’obligation’(::o), ‘dispensation’(::d)) que regem a atividade estão em conflito;

  2. Mapear se duas ou mais atividades envolvem um mesmo ator e identificar se os operadores normativos (‘right’(::r), ‘prohibition’(::p), ’obligation’(::o), ‘dispensation’(::d)) que regem a atividade estão em conflito;

  3. Analisar duas ou mais atividades que envolvem um mesmo objeto e verificar se são conflitantes;

  4. Analisar duas ou mais atividades que envolvem um mesmo ator e verificar se são conflitantes.

3.4.2 RESULTADOS ESPERADOS

Com o uso da técnica de verificação de conflito utilizando modelos de apoio (tabelas tridimensionais, quadrimensional e diagramas) mapeando as

interações entre as dimensões do modelo M-Forum (CAMOLESI & MARTINS, 2006), espera-se um refinamento das especificações mapeadas na linguagem L-Forum proporcionando um menor retrabalho e uma maior agilidade nas verificações das regras incrementais, uma vez que os modelos de apoio permitam visualização mais fácil das possíveis inconsistências.

3.5 VERIFICAÇÃO INTER-REGRAS

Para verificar as regras com a mesma aplicabilidade, as chamadas que uma regra faz às outras, a técnica utiliza modelos de apoio (tabelas tridimensionais, quadrimensional e diagramas), mapeando as interações entre as dimensões presentes.

Com base no conjunto de regras do escopo, escrito em linguagem L-Fórum (CAMOLESI & MARTINS, 2006), foi desenhado o diagrama das regras (figura 15), montado as tabelas da verificação intra-regra e posteriormente a tabela MAR (Tabela Aplicabilidade x Regra).

Dentre as possíveis verificações nessa etapa temos:

- Visualizar um conjunto de regras que possuem uma mesma aplicabilidade e a associação entre as aplicabilidades durante a chamada de uma regra, que em sua estrutura não conste a aplicabilidade da qual foi chamada (regra interna).

- Através da tabela MAR, é possível verificar a possibilidade de unificar as aplicabilidades comuns em uma regra, como exemplo a regra R2 (tabela MAR), que contem duas aplicabilidades;

- A verificação dos parâmetros de regras chamadas: analisando as atividades de uma regra no diagrama, é possível verificar se todos os parâmetros utilizados nas atividades estão retratados no parâmetro da regra.

- No caso das inter-regras chamadas, a verificação dos parâmetros deve

iniciar da última regra a ser executada para a primeira, pois os atributos de todas as regras devem estar presentes no parâmetro desde a primeira regra (a regra principal, a que engloba as demais).

3.5.1 ETAPAS DO SUB-PROCESSO

Para verificar as regras com a mesma aplicabilidade e as chamadas que uma regra faz a outras, a técnica utiliza modelos de apoio (tabelas tridimensionais, quadrimensional e diagramas) mapeando as interações entre as dimensões presentes. As etapas do sub-processo foram divididas em 2 passos:

- Passo 1: efetuar a verificação intra-regra que oferece uma consistência das atividades nas regras.

- Passo 2: montar a tabela MAR (Tabela Aplicabilidade x Regra), nesse passo é possível:

    1. Visualizar um conjunto de regras que possuem uma mesma aplicabilidade e a associação entre as aplicabilidades durante a chamada de uma regra, que em sua estrutura não conste a aplicabilidade da qual foi chamada (regra interna).

    2. Verificar a possibilidade de unificar as regras de mesma aplicabilidade em uma única regra, como exemplo a regra R2 (tabela MAR), que contem duas aplicabilidades e uma das aplicabilidades poderia migrar para a outra regra existente (R1);

    3. Filtrar as regras chamadas, para possibilitar a verificação dos parâmetros das inter-regras mapeados no diagrama. O processo deve iniciar pela última regra a ser executada até a primeira, pois os atributos de todas as regras devem estar presentes no parâmetro, desde a primeira regra (a regra principal, a que engloba as demais).

3.5.2 RESULTADOS ESPERADOS

Com o uso da técnica de verificação, utilizando modelos de apoio (tabelas tridimensionais, quadrimensional e diagramas) mapeando as interações entre as dimensões do modelo M-Forum (CAMOLESI & MARTINS, 2006).

Espera-se um refinamento das especificações mapeadas na linguagem L-Forum, verificando as possíveis unificações de regras e rastreio dos parâmetros passados através das regras chamadas.

3.6 CONSIDERAÇÕES FINAIS

A técnica de verificação propõe modelos de apoio para facilitar a análise do testador, porém a habilidade com a técnica e a experiência pode influenciar diretamente no resultado.

4. ESTUDO EMPÍRICO DO PROCESSO

4.1 CONSIDERAÇÕES INICIAIS

Para aplicar da técnica de verificação utilizando modelos de apoio (tabelas tridimensionais, quadrimensional e diagramas) mapeando as interações entre as dimensões no modelo M-Forum (CAMOLESI & MARTINS, 2006). Foi escrito em linguagem L-Forum as regras de um sistema de *Service Desk*, posteriormente inseridos sete erros variados no conjunto de regras e distribuídos para cinco testadores, utilizarem a técnica de verificação para saná-los.

4.2 ESTUDO DE CASO

Um sistema de *Service Desk,* que possui uma estrutura colaborativa nos grupos de atendentes, é composto de três níveis de suporte. Elaborado primeiramente para registrar os atendimentos dos suportes aos clientes, foi modificado para dinamizar o atendimento e criar um banco de conhecimento para registrar soluções de incidentes recorrentes.

Para tanto, as abribuições dos níveis de suporte e coordenação seguem as principais regras a seguir:

FIGURA 16 – TICKET ABERTO

- O primeiro nível recebe um *ticket* por telefone ou e-mail, verifica a prioridade negociada para o aplicativo, registra-o e encaminha-o ao segundo nível do suporte;

- O segundo nível verifica no banco de conhecimento uma solução para o ticket, caso não encontre trabalha na resolução por até metade dos dias acordados para o final do atendimento. Passado o período e não tendo uma solução para o ticket esse é encaminhado para o terceiro nível;

- O terceiro nível trabalha na solução dos *tickets* ainda não resolvidos, com a maior prioridade registrada;

- Um gerente monitora o processo e atualiza o banco de conhecimento com as soluções úteis geradas nos atendimentos finalizados.

FIGURA 17 – GESTÃO DOS TICKETS

Para a gestão dos tickets, o modelo disponibiliza um gerador de consultas e relatórios que facilita as consultas repetidas. Pode se parametrizado, enviar automaticamente uma seleção de tickets para o grupo responsável pelo atendimento.

4.2.1 DESCRIÇÃO DA POLÍTICA DE REGRAS

As regras que regem o sistema de Service Desk composto de três níveis de suporte e um de coordenação foram escritos em linguagem L-Forum (CAMOLESI & MARTINS, 2006). O conjunto das regras e sua seqüência de execução formam a política de regras que gerência o ambiente.

Antes de descrever a política de regras, a linguagem L-Forum, descreve as dimensões da regra, quais são os atores, objetos, atividades, tempo e espaço e

suas abstrações reconhecidas no ambiente:

Atores:	Clientes, Atendente nível 1, Atendente nível 2, Atendente nível 3, Gerente
Objetos:	Ticket, Banco de Conhecimento, SLA negociado por Módulo
Atividades:	Atualiza, Analisa, Identifica, Encaminha, Trabalha
Tempo:	SLA negociado por Módulo
Espaço:	Empresa
Grupo:	Grupo de Atendente nível 1, Grupo de Atendente nível 2, Grupo de Atendente nível 3
Abstrações:	Atendente nível 1.compõe.Grupo de Atendente nível 1
	Atendente nível 2.compõe.Grupo de Atendente nível 2
	Atendente nível 3.compõe.Grupo de Atendente nível 3
	Cliente.possui.Cadastro
	SLA negociado por Módulo.possui.Prioridade
	SLA negociado por Módulo.possui.Tempo atendimento
	Banco de Conhecimento.possui.Solução
	Ticket.possui.Prioridade
	Ticket.possui.Status
	Ticket.possui.Nivel
	Ticket.possui.Histórico
	Ticket.possui.Solução
	Ticket.possui.Data criação
	Ticket.possui.Analista

A regra Registra Ticket (R1), trata as atividades do grupo de atendentes do primeiro nível que recebe um ticket por telefone ou e-mail, verifica a prioridade negociada para o aplicativo, efetua o registro e encaminha ao segundo nível do suporte;

Rule	Registra Ticket (R1) [ACTIVE]
{	
Parameters::	(c:Cliente, gn1: Grupo de Atendente nível 1, gn2: Grupo de Atendente nível 2, t:Ticket, s:SLA negociado por Módulo.prioridade)
Applicability::	(t.prioridade is = 0)
Body::	Execute Action [1] (gn1 obligation analisa c.cadastro t.historico);
	Execute Action [2] (gn1 obligation identifica t.historico s.prioridade);
	Execute Action [3] (gn1 obligation encaminha t gn2 t.data criação);
}	

A regra Atende NV2 (R2), trata as atividades do grupo de atendentes do segundo nível que verifica no banco de conhecimento uma solução para o ticket, caso não encontre, trabalha na resolução por até metade dos dias acordados para o final do atendimento. Passado o período e não tendo uma solução para o ticket esse é encaminhado para o terceiro nível;

```
Rule            Atende NV2 (R2)        [ACTIVE]
{
Parameters::    (gn2: Grupo de Atendente nível 2, t:Ticket, b: Banco de Conhecimento, s:SLA)
Applicability:: (t.nivel = 2 and t.prioridade > 0 and t.status = 'Pendente' )
Body::          If ( data atual - t.data criação > s.tempo atendimento / 2 )
                 Then {  Execute Action  [ 1 ] (gn2 obligation encaminha  t gn3  t.data atual)};
                Execute Action [ 2 ] (gn2 obligation analisa t.historico b );
                If ( b.solução is aceita )
                    Then { Execute Action   [ 3 ] (gn2 obligation atualiza t.status t.solução) };
                    Else   { Execute Action  [ 3 ] (gn2 obligation trabalha t ) };
}
```

A regra Atende NV3 (R3), trata as atividades do grupo de atendentes do terceiro nível, que trabalha na solução dos tickets ainda não resolvidos, com a maior prioridade registrada;

```
Rule            Atende NV3 (R3)        [ACTIVE]
{
Parameters::    (gn3: Grupo de Atendente nível 3, t:Ticket, b: Banco de Conhecimento)
Applicability:: (t.nivel = 3 and t.prioridade is > 0 and t.status = 'Pendente' )
Body::          Execute Action [ 1 ] (gn3 obligation analisa t.historico b );
                If ( b.solução is aceita )
                    Then { Execute  Action  [ 2 ] (gn3 obligation atualiza t.status t.solução) };
                    Else  {  Execute Action  [ 2 ] (gn3 obligation trabalha t )};
}
```

A regra Gerência (R4), trata as atividades gerente monitora o processo, reclassifica os tickets e atualiza o Banco de Conhecimento com as soluções úteis geradas nos atendimentos finalizados;

```
Rule            Gerência (R4)           [ACTIVE]
{
Parameters::    (g:Gerente, gn3: Grupo de Atendente nível 3, t:Ticket, b:Banco de
Conhecimento)
Applicability::  (t.status = 'Pendente')
Body::
                If ( t.prioridade is Alta )
                    Then { Execute  Action  [ 1 ] (g obligation encaminha t.data atual gn3)};
Applicability::  (t.status = 'Concluido')
Body::
                If ( t.solução is aceita in b )
                    Then { Execute  Action  [ 1 ] (g obligation atualiza t.solução b) };
}
```

4.2.2 PERFIL DOS TESTADORES

O primeiro testador é estagiário na empresa cursando o último semestre de Tecnologia da Informação; o segundo é analista de sistemas com mais de quinze anos de profissão e com formação em Administração; o terceiro e o quarto são analistas de sistemas com mais de dez anos de experiência e com formação na área de Exatas; o quinto cursa mestrado em Ciências da Computação e possui mais de oito anos de experiência profissional.

Todos os testadores utilizaram uma versão beta do *Service Desk* com a estrutura de três níveis (nivel 1 recebe o *ticket*, nivel 2 efetua um primeiro atendimento e quando não consegue resolver encaminha para o nivel 3), porém sem o tempo de atendimento e a geração do banco de conhecimento que compõe a proposta de melhoria para o sistema.

4.2.3 A INJEÇÃO DE ERROS

Para testar a aplicabilidade do método de verificação de regras por testadores, foram injetados sete erros de 6 tipos, distribuídos entre as principais

possibilidades do modelo:

- Na seção das intra-regras, possibilita a verificação dos parâmetros: analisando as atividades de uma regra no diagrama, é possível verificar se todos os parâmetros utilizados nas atividades estão retratados no parâmetro da regra.

- Se as ações dos atores nos objetos ou no tempo, são seqüenciais ou paralelas: normalmente, no processo colaborativo, as ações são paralelas. Quando a política necessita ordenar um grupo de atividades, podemos verificar a seqüência de ações na regra, utilizando o modelo de diagrama.

Na existência da prática condenável de copiar uma parte do código em informática, principalmente pelos profissionais mais experientes, a verificação dos parâmetros passados contra os utilizados se mostrou bastante útil.

Para possibilitar a verificação dos parâmetros e a seqüência de execução das atividades foram inseridos erros na regra Atende NV2 abaixo:

```
Rule            Atende NV2 (R2)        [ACTIVE]
{
Parameters::    (c:Cliente, gn2: Grupo de Atendente nível 2, t:Ticket, b: Banco de
                Conhecimento)
Applicability:: (t.nivel = 2 and t.prioridade > 0 and t.status = 'Pendente' )
Body::          If ( data atual - t.data criação > s.tempo atendimento / 2 )
                 Then {  Execute Action  [ 1 ] (gn1 obligation encaminha  t gn3  t.data atual)};
                Execute Action [ 2 ] (gn2 obligation analisa t.historico b );
                If ( b.solução is aceita )
                     Then { Execute Action   [ 3 ] (gn2 obligation atualiza t.status t.solução) };
                     Else   { Execute Action  [ 4 ] (gn2 obligation trabalha t ) };
}
```

Na regra acima, quatro erros foram inseridos:

1. Ator c:Cliente não utilizado em uma atividade na regra e presente no parâmetro;

2. Atores gn3 e gn1 utilizados na regra e não estão presentes no parâmetro, no caso do gn1 o correto para a atividade é o gn2;

3. A seqüência de execução 4 deveria ser 3 por se tratar de uma derivação de fluxo, onde independente da ação a ser executada será a terceira.

O modelo apresentado na figura 18, possibilita a verificação dos parâmetros e a seqüência das atividades da regra como uma das primeiras verificações, isso é possível através do desenho das regras no diagrama.

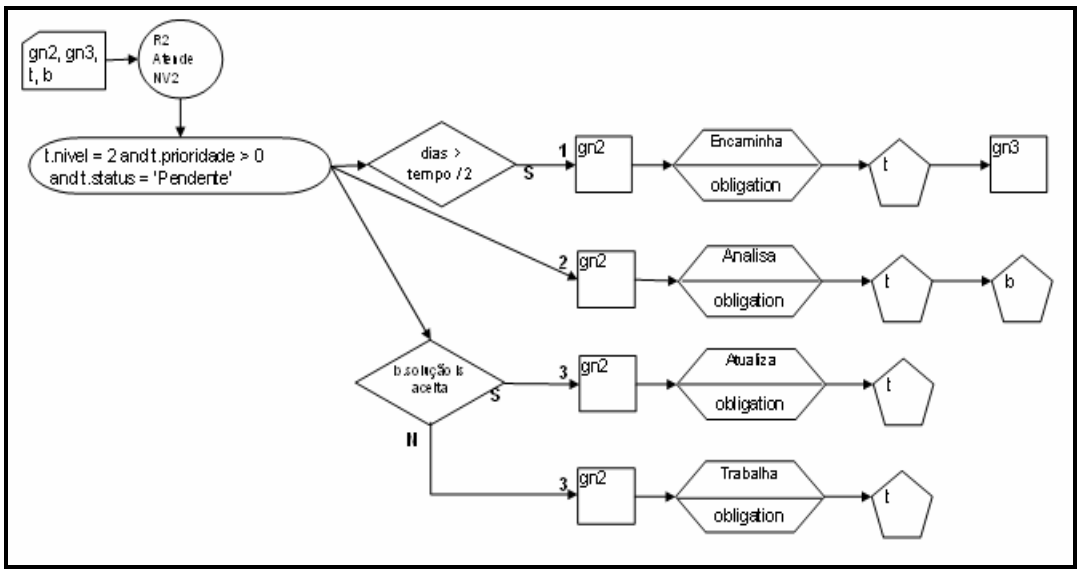

FIGURA 18 – VERIFICAÇÃO DOS PARÂMETROS DAS INTRA-REGRAS

A figura 18 retrata a regra R2 Atende NV2 correta, sendo o passo seguinte alteração da regra, na linguagem L-Forum (CAMOLESI & MARTINS, 2006).

- Se existe aplicabilidade repetida dentro de uma regra: através do modelo de apoio do diagrama, podemos verificar a repetição das aplicabilidades de uma regra. Caso isso ocorra, devemos excluir uma das aplicabilidades e inserir as atividades da aplicabilidade excluída em um único corpo de regra, unificando as atividades e analisando seu impacto, ou ratificar a aplicabilidade.

Para verificar aplicabilidade repetida, foi utilizada a regra Gerência (R4), com a inserção de um erro:

```
Rule            Gerência (R4)          [ACTIVE]
{
Parameters::    (g:Gerente, gn3: Grupo de Atendente nível 3, t:Ticket, b:Banco de Conhecimento)
Applicability:: (t.status = 'Pendente')
Body::
                If ( t.prioridade is Alta )
                    Then { Execute  Action  [ 1 ] (g obligation encaminha t.data atual gn3)};
Applicability:: (t.status = 'Pendente')
Body::
                If ( t.solução is aceita in b )
                    Then { Execute  Action  [ 1 ] (g obligation atualiza t.solução b) };
}
```

Na regra anterior, foi inserido o quinto erro no conjunto de regras, na forma de aplicabilidade repetida dentro da regra:

4. Aplicabilidade (t.status = 'Pendente') está repetida na regra, muito provavelmente por um descuido do analista, o correto para o caso é outra aplicabilidade, como (t.status = 'Concluído');

O modelo possibilita a verificação da aplicabilidade de duas maneiras, nas intra-regras, a través da tabela MAR, é registrada a interação das regras. Possibilita facilmente visualizar um conjunto de regras que possuem uma mesma aplicabilidade.

Outra possibilidade, é a associação entre as aplicabilidades durante a chamada de uma regra, que em sua estrutura não consta a aplicabilidade da qual foi chamada (regra interna).

Nas inter-regras, a verificação se dá através do diagrama, como é mostrado na figura 19.

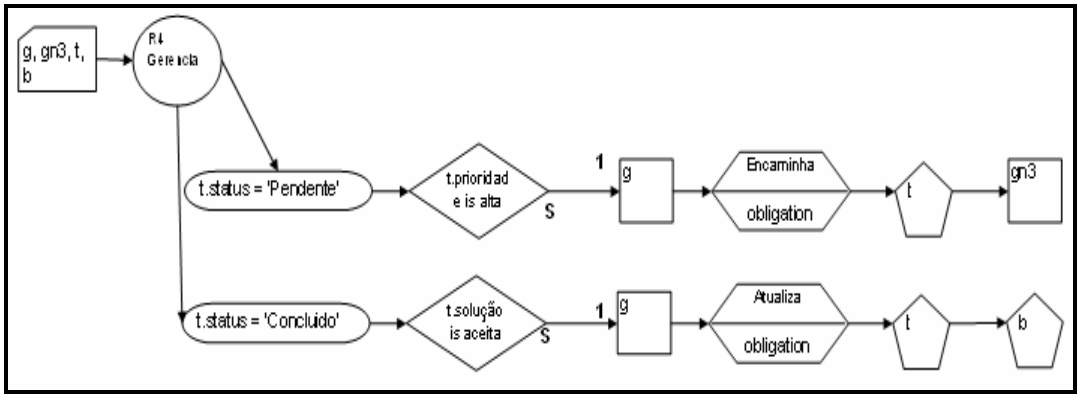

FIGURA 19 – VERIFICAÇÃO DAS APLICABILIDADES DAS INTRA-REGRAS

A figura 19 retrata a regra R4 Gerência correta, sendo o passo seguinte alteração da regra na linguagem L-Forum (CAMOLESI & MARTINS, 2006).

- Mapear se duas ou mais atividades são executadas por um mesmo ator: o mapeamento é feito através da tabela MARAO, onde além de relacionar as atividades que envolvem um ator, é representado os operadores normativos (‘right’(::r), ‘prohibition’(::p), ’obligation’(::o), ‘dispensation’(::d)) que regem a atividade.

Para relacionar as atividades dos atores nas regras, foi utilizada a regra Atende NV3 (R3), com a inserção de um erro:

```
Rule            Atende NV3 (R3)        [ACTIVE]
{
Parameters::    (gn3: Grupo de Atendente nível 3, t:Ticket, b: Banco de Conhecimento)
Applicability:: (t.nivel = 3 and t.prioridade is > 0 and t.status = ‘Pendente’ )
Body::          Execute Action [ 1 ] (gn3 obligation analisa t.historico b );
                If ( b.solução is aceita )
                    Then { Execute  Action  [ 2 ] (gn3 obligation analisa t.status t.solução) };
                    Else  {  Execute Action  [ 2 ] (gn3 obligation trabalha _ )};
}
```

Na regra R3, foi inserido o sexto e o sétimo erro no conjunto de regras, o sexto repetindo a atividade dentro da regra e o sétimo na omissão de um ator ou objeto na atividade trabalha:

5. A atividade (gn3 obligation analisa...) está repetida na regra, o correto para o caso é a atividade (gn3 obligation atualiza t.status t.solução).

		Objeto		
Regra	**Ator**	Ticket	SLA	Banco de Conhecimento
R3	Grupo Nível 3	**Analisa::o**		**Analisa::o**
R3	Grupo Nível 3	**Atualiza::o**		
R3	Grupo Nível 3	**Trabalha::o**		

TABELA 5: MARAO – REGRA ATENDE NV3

Durante o quinto passo da verificação das inter-regras, é feito o mapeamento da tabela MARAO (Tabela Atividade x Regra x Ator x Objeto), ao registrar a segunda atividade da regra, é possível perceber o erro na regra e corrigi-la. A tabela acima mostra as atividades corrigidas na regra R3.

6. A atividade (gn3 obligation trabalha _) está faltando um ator ou objeto, o correto para o caso é a atividade (gn3 obligation trabalha t), sendo t o ticket.

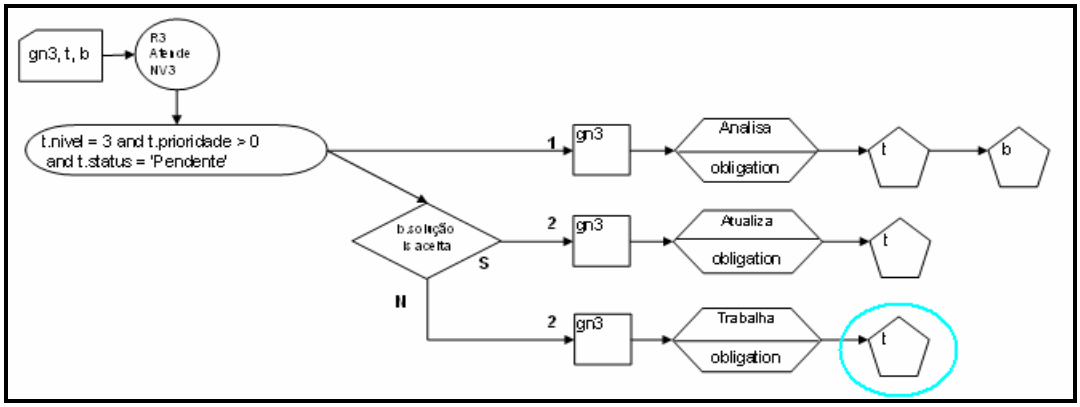

FIGURA 20 – VERIFICAÇÃO DAS ATIVIDADES DAS INTRA-REGRAS

A figura 21 retrata a regra R3 Atende NV3 correta, mostrando em destaque o objeto que faltava na grafia da atividade. O Modelo M-Forum (CAMOLESI & MARTINS, 2006) define que uma regra envolva pelo menos um ator e um objeto, ou outro ator.

4.2.4 RESULTADOS OBTIDOS

Analisando os resultados da tabela 6, obtidos no estudo empírico de um método de verificação para o modelo M-Forum (CAMOLESI & MARTINS, 2006). Foi diagnosticado que o modelo atendeu satisfatoriamente a verificação dos parâmetros das intra-regras, utilizados nas atividades presentes na regra, independente do avaliador.

ERRO INSERIDO	VERIFICADOR					ACERTO	
	1	2	3	4	5	Total	%
Ator c não utilizado na regra R1 e presente no parâmetro	1	1	1	1	1	5	**100**
Ator gn1 utilizado na regra R1 e não presentes no parâmetro	1	1	1	1	1	5	**100**
Ator gn3 utilizado na regra R1 e não presentes no parâmetro	1	0	1	1	1	4	**80**
Seqüência de execução 4 deveria ser 3 na R1 por se tratar de uma derivação de fluxo	0	1	1	1	1	4	**80**
Aplicabilidade (t.status = 'Pendente') está repetida na regra R4	1	0	1	1	0	3	**60**
Atividade (gn3 obligation analisa...) está repetida na regra R3	0	1	0	1	1	3	**60**
Atividade (gn3 obligation trabalha _) está faltando um ator ou objeto	0	1	1	0	1	3	**60**
Total de acertos	4	5	6	6	6	**27**	
Percentual	**57**	**71**	**86**	**86**	**86**		**77**

TABELA 6: RESULTADO DA VERIFICAÇÃO DE REGRAS

Os diagramas são ferramentas muito úteis na proposta de verificação das intra-regras, praticamente com ele, é possível diagnosticar os problemas de estrutura da regra, já as tabelas, são mais utilizadas quando necessitamos filtrar um grupo para uma análise refinada.

Apesar do grupo de testadores conhecerem o negócio retratado em regras, suas experiências e técnica com análise de sistema são distribuídos em quatro grupos: iniciante: verificador 1; experiente sem formação na área de Ciências Exatas: verificador 2; experientes com formação na área referida: verificadores 3 e 4; experiente com pós-graduação na área referida: verificador 5.

A verificação com tabelas só é eficaz quando primeiro validamos as regras no

diagrama, as tabelas são novas verificações ou uma confirmação da primeira.

O método de verificação segue uma seqüência de passos, montagem do diagrama das regras internas, matrizes das intra-regras, matrizes das inter-regras, porém um erro no desenho do diagrama reflete na montagem das matrizes, o fato ocorreu de forma variada em três erros inseridos que gerou uma taxa de acerto de 60% da população dos testadores.

4.3 CONSIDERAÇÕES FINAIS

Com um índice de acerto na verificação entre 57% e 86%, mostra que a figura do testador tem um papel fundamental na verificação e sua experiência com o método proposto e com o negócio mapeado exerce uma grande influência no resultado.

As verificações inter-regras, são utilizadas como um refino das verificações intra-regras e muito importantes no desenvolvimento incremental, onde uma regra validada e estável, com a inclusão ou alteração de uma regra no conjunto de regras, pode alterar o equilíbrio da política de regras.

5 CONCLUSÃO

5.1 CONSIDERAÇÕES INICIAIS

O objetivo deste trabalho é definir uma técnica de verificação de regras durante a especificação incremental em um ambiente colaborativo.

Esse trabalho parte da premissa de que não existe uma estratégia de resolução de conflitos que seja adequada a qualquer tipo de sistema, porém os conflitos detectados podem ser classificados e a utilização de uma estratégia específica em função do tipo de conflito pode conduzir a resultados mais satisfatórios.

Com o uso da técnica proposta, foi possível identificar grande parte das situações de conflito de regras para análise de um testador, agindo como um pré-teste de especificação, para aprimorar o desenvolvimento espiral.

As verificações inter-regras são utilizadas como um refino das verificações intra-regras e são muito importantes no desenvolvimento incremental.

5.2 CONTRIBUIÇÕES DA PESQUISA

Atualmente, a demanda de CSCW, está voltada para a especificação e desenvolvimento de sistemas, para a coordenação de processos de trabalho (workflow), para a telecooperação, no Ensino, na Arquitetura, em projetos de Engenharia e na Telemedicina.

Na área de especificação e desenvolvimento de sistemas, a técnica possibilita que vários profissionais interajam durante a especificação de um sistema e que a inserção do conhecimento do negócio, seja facilitada para todos os participantes, possibilitando uma melhoria na qualidade do projeto.

Para tanto, a técnica proposta, identifica os tipos de conflitos para um projetista

ou testador tratá-los, tendo na fase de especificação, a possibilidade de um pré-teste de conceito, agilizando o processo e reduzindo os custos do projeto.

A normatização do processo de verificação para detecção de conflitos, traz a redução de erros e consequentemente de custo, a velocidade de desenvolvimento necessária para aproveitar as oportunidades de mercado, a otimização dos recursos humanos e não-humanos e a reciclagem, visando melhoramento e disseminação do conhecimento.

5.3 TRABALHOS FUTUROS

Futuros trabalhos podem facilitar o desenvolvimento e emprego da técnica, entre eles, o desenvolvimento de uma ferramenta colaborativa para a escrita e gestão das regras estruturadas em linguagem L-Forum (CAMOLESI & MARTINS, 2006), uma vez que o modelo atende satisfatoriamente as parametrizações das intra-regras.

Acrescer as dimensões tempo e espaço nas tabelas, é também uma contribuição importante, pois o modelo atual de verificação ainda não as trata, no caso do tempo, existe um índice numérico que rege a seqüência das atividades, porém não atende plenamente.

Outra contribuição de grande valia, é o controle das atividades nas regras englobando os atributos valorados das dimensões Ator e Objeto, exemplo: Execute Action [1] (gn1 obligation encaminha t.analista = gn3 t.data atual).

5.4 CONSIDERAÇÕES FINAIS

A técnica demonstrou ser eficaz principalmente na verificação das intra-regras, independente da qualificação do testador, trazendo essa verdade para o âmbito empresarial.

As empresas necessitam aproveitar as oportunidades de mercado e os projetos tendem a ter cronogramas mais otimizados, com a aplicação da técnica de

verificação de conflitos, teremos uma melhoria das especificações.

Em grandes empresas de software, o ambiente da elaboração de projetos de sistemas possui uma divisão de funções e até de locais físicos. Uma equipe é responsável pelo desenho, outra pela criação do código e outra para o teste / implantação.

Porém erros de especificação são comumente encontrados na fase de implantação, a aplicação da técnica de verificação de conflitos de regra no primeiro estágio, garante uma melhoria na cadeia do projeto.

Hoje existe pouca literatura disponível para validação de regras. Esse trabalho de verificação de conflito de regras pode contribuir para trabalhos futuros de validação.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSIS, R. L. **Facilitando a percepção em ambiente virtuais de aprendizado através da tecnologia groupware**, dissertação de mestrado, Departamento de Informática, PUC - Rio, Abril 2000.

AHMED, T. & TRIPATHI, A. R. **Static Verification of Security Requirements in Role Based CSCW Systems**, *SACMAT'03*, June 1-4, 2003, Como, Italy. - ACM 1581136811/03/0006…, 2003.

BORDEAU, J. and B. WASSON. **Orchestrating collaboration in collaborative telelearning. Artificial intelligence in education**. B. Boulay and R. Mizoguchi, IOS Press: 565-567, 1997.

BREHMER, B. **Distributed decision making: some notes on the literature. Distributed decision making : cognitive models for cooperative work**. J. Rasmussen, B. Brehmer and J. Leplat. Chichester, England; New York, 1991.

BRINCK, T. & MCDANIEL, S. E. **Awareness in Colaborative Systems**, *Workshop Report*, SIGCHI Bulletin, 1997.

BROOKE, J. **User interfaces for CSCW systems, in CSCW in practice: an Introduction and case studies**, Dan Dapier e Colston Sanger (eds.), Springer-Verlag, 1993.

CAMOLESI JR, L. & MARTINS, L. E. G. **A Model for Interaction Rules to Define Governance Policies in Collaborative Environments**, Lecture Notes in Computer Science - LNCS, Springer-Verlag, Berlin Heidelberg, Vol. 3865, Computer Support Cooperative Work in Design 2, W. Shen and Kuo-Ming Chao et al.(Eds.). 11-20. 2006.

CAMOLESI JR, L. & MARTINS, L. E. G. ***FORUM*: Modelo e Linguagem para Especificação de Regras em Ambientes Colaborativos**, 2006.

CHANG, C. K., VORONTSOV, A., ZHANG, J., QUEK, F. **Rule-Mitigated Collaboration Technology**, IEEE Workshop on Future Trends of Distributed Computing Systems, 137-142. 1999.

DILLENBOURG, P., M. Baker, A. Blaye and C. O'Malley. **The Evolution of Research on Collaborative Learning.** Learning in humans and machines. Towards an interdisciplinary learning science. P. Reimann and H. Spada. London, Pergamon: 189-211. 1995.

EGENHOFER, M. J., MARK, D. **Modeling Conceptual Neighborhoods of Topological Line-Region Relationships**, Int. Journal Geographical Inf. Systems. 9:5 555-565. 1995.

FUKS, H & ASSIS, R. L. **Facilitating Perception on Virtual Learningware-Based Environments**, *Journal on Systems and Information Technology*, 2001, in press. 2001.

FUKS, H., LAUFER, C., CHOREN, R., & BLOIS, M. **Communication, coordination and cooperation in distance education**, *Proceedings of Americas Conference on Information Systems*, pp 130-132, 1999.

HOLZMANN, G. J. **The Model Checker SPIN**. IEEE Transactions on Software Engineering, 23(5):279-295. 1997.

JENNINGS, N. and CAMPOS, J.R. **Towards a Social Level Characterization of Socially Responsible Agents**, IEE Proceedings on Software Engineering, 144 (1), pp. 11-25. 1998.

KAGAL, L., FININ, T., JOHSHI, A. **A Policy Language for Pervasive Computing Environment**. IEEE Int. Workshop on Policy for Distributed Systems and Networks. 63-76. 2003.

LEWIN, K **Conflito.** Disponível em**: <http://pt.wikipedia.org/wiki/Conflito>** Acesso em: 21 Dez. 2007.

LIU, T.H. and others. **Classification and Representation of Conflict in Multi-Agent Systems**, Technical Report, University of Texas, 1998.

MOECKEL, Leila C. F. **Implementação de ambiente web para informação e interação em um programa de pós-graduação**. Curitiba, Dissertação (Mestrado em Tecnologia) - PPGTE, CEFET-PR. 2001.

MOK, A. K., LEE, C., WOO, H. **The Monitoring of Timing Constraints on Time Intervals.** IEEE Real-Time Systems Symposium, 1-10. 2002.

PASCHKE, Adrian **ECA Rule Markup Language,** *Journal on Systems and Information Technology*. 2005.

PLACCA, J.A., GARCIA, A.C.B. & DUTRA, I.C. **Um Modelo de Resolução de Conflitos em Sistemas Multiagentes Aplicado ao Projeto de uma Plataforma Off-Shore de Petróleo,** Sociedade Brasileira de Engenharia Naval. 2003.

SHANKAR, C. S., RANGANATHAN, A. & CAMPBELL, R. **An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments, Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services** (MobiQuitous'05) IEEE 0-7695-2375-7/05, 2005.

SHOHAM, Y., TENNENHOLTZ, M. **On social laws for artifical agente societies: off-line design**, Artificial Intelligence Magazine. 1996.

TAKADA, H., SHIMAKAWA, H., HORIIKE, S. **The Time/Place /Object Model for Tracking and History Management in Manufacturing Line Control**, IEEE Int. Symposium on Database Applications in Non-Traditional Environments,

385-394. 1999.

TUROFF, M. & HILTZ, S. R. **Computer Support for Group versus Individual Decisions**, *IEEE Transactions on Communications*, 30, (1), pp 82-91, 1982.

APÊNDICE

<rule> ::=	´Rule´ <rule name> ´(´<status>´)´ ´{´ (<context>) ´Body ::´<definition> (<regime>) ´}´
<context>::=	(´Parameters: (´<parameters>´)´) (<applicability>)
<definition> ::=	<condition> | <action> | <rule call> (<definition>)
<regime> ::=	<survivability> (´Priorities:´ <priority>)
<parameters> ::=	<identifier>´:´<element> (´,´ <parameters>)
<element> ::=	<actor> | <group> | <object> | <space> | <time> | <association> | <activity> | <operation>
<type> ::=	´actor´ | ´group´ | ´object´ | ´space´ | ´time´ | ´association´ | ´activity´ | ´operation´
<applicability> ::=	´Applicability::´ <condition expression>
<survivability> ::=	´Survivability::´ <condition expression>
<condition> ::=	´If ´ <condition expression> ´then {´ <definition> ´}´ (´else {´ <definition> ´}´)
<action> ::=	´Action: (´ <supreme action> | <definition action> | <attribution action> | (<actor> <normative operator> {<activity> (<actor> | <object>)} (<space attribution operation> <space>) (<time attribution operation> <time>)) (´);´ <action>) ´);´
<supreme action> ::=	<actor> <normative operator> <primitive operator> (<element>|<domain>|(´is part of´|´is a´) <element>) | <actor> <normative operator> <primitive group operator> <group> <element>)
<definition action> ::=	<actor> ´set´ <status>
<attribution action> ::=	<actor> ´attribute´ (<value>|<formula>| ((next | prior) (<value domain>|<domain name>)) <attribute>
<condition expression> ::=	´(´ (<attribute><attribute condition operator>(<value>| ((´all´|´any´) (<value domain>|<domain name>)) | ((<element>|<rule name>) <status condition operator> <value>) | (<element>) <element group operator> <group>) | (<element>) <group group operator> <group>) | ((not) ´exist´<association>) | (<element> <space condition operator> (<space> | (<group>|<domain name>))) | (<attribute><space condition operator>(<value>|´here´|((´all´|´any´) (<value domain>|<domain name>)))) | (<attribute><time condition operator>(<value>|´now´|((´all´|´any´) (<value domain>|<domain name>)))) | (<actor> <activity condition operator> <activity>) |(<condition expression> (and | or)) ´)´
<rule call> ::=	´Rule (´ <rule name> ´ (´<parameters>´)´ <normative operator> (´);´ <rule call>) ´);´
<priority> ::=	<name> (´,´ <priority>)
<group> ::=	((<identifier>) |´any´ | ´all´| <integer value>) (<association>) (´:´<name>) (´:Group´)
<actor> ::=	((<identifier>) |´any´ | ´all´| <integer value>) (<association>) (<group>´.´) (´:´<name>) (´:Actor´)
<activity> ::=	(<activity>´.´) <name> (´:Activity´)
<operation> ::=	<activity>´.´<name> (´:Operation´)
<object> ::=	((<identifier>) |´any´ | ´all´| <integer value>) (<association>) (<object>´.´) (´:´<name>) (´:Object´)
<space> ::=	((<identifier>) |´any´ | ´all´| <integer value>) (<association>) (<space>´.´) (´:´<name>) (´:Space´)

<time> ::=	((<identifier>) |´any´ | ´all´| <integer value>) (<association>) (´:´<name>) (´:Time´)
<association> ::=	<element>´.´<name> (´.´<association>) (´:Association´)
<attribute> ::=	<element>´.´<name> (´:Attribute´)
<domain> ::=	<name> (<value domain> | <grouping>)
<value domain> ::=	´(´ (<numeric value> {´,´ <numeric value>}) | (<string> {´,´ <string>}) ´)´
<grouping> ::=	(<type> <name> <attribute condition operator> (<value>|((´all´| ´any´)(<value domain>|<domain name>))) {(and| or) <grouping>}) | (<element> {´,´ <element>})
<status> ::=	<element> <status attribution operator> <value>
<primitive group operator> ::=	´insert´ | ´delete´ | ´update´
<primitive operator> ::=	´create´ | ´destroy´
<group element operator> ::=	´∈´ | ´∉´ |
<group group operator> ::=	´⊆´ | ´⊄´ | ´⊂´
<activity condition operator> ::=	(´not´) ´has´
<normative operator> ::=	´right´ | ´prohibition´ | ´obligation´ | ´dispensation´
<activity attribution operator> ::=	´receive´
<status attribution operator> ::=	´put on´ | ´move to´
<space attribution operator> ::=	´==´ | ´inside´ | ´outside´ | ´north´ | ´south´ | ´east´ | ´west´
<time attribution operator> ::=	´in´ | ´on´ | ´at´
<attribute condition operator> ::=	´<´ | ´<=´ | ´>´ | ´>=´ | ´=´ | ´<>´
<time condition operator> ::=	´<´ | ´<=´ | ´>´ | ´>=´ | ´=´ | ´<>´ | ´precedes´ | ´succeeds´ | ´directly precedes´ | ´directly succeeds´| ´overlaps´| ´is overlapped by´| ´occurs during´| ´contains´| ´starts´ | ´is stated with´| ´finishes´| ´is finished with´| ´coincides with´
<space condition operator> ::=	´<´ | ´<=´ | ´>´ | ´>=´ | ´=´ | ´<>´ | ´not equal´ | ´inside´ | ´outside´ | ´intersect´ | ´meet´ | ´overlap´ | ´north´ | ´south´ | ´east´ | ´west´
<arithmetic operator> ::=	´+´ | ´-´ | ´/´ | ´*´
<status condition operator> ::=	(´not´) ´is´
<rule name> ::=	<string>
<name> ::=	<string>
<domain name>::=	<string>
<string> ::=	<letter>,{<letter>}
<integer value> ::=	<digit>,{<digit>}
<real value> ::=	<digit>,{<digit>},´.´,<digit>,{<digit>}
<numeric value> ::=	<integer value> | <real value>
<digit> ::=	´1´ | ´2´ | ´3´ | ´4´ | ´5´ |´6´ | ´7´ | ´8´ | ´9´ | ´0´
<letter> ::=	´a´ | ´b´ | ´c´ | ´d´ | ´e´ | ´f´ | ´g´ | ´h´ | ´I´ | ´j´ | ´k´ | ´l´ | ´m´ | ´n´ | ´o´ | ´p´ | ´q´ | ´r´ | ´s´ | ´t´ | ´u´ | ´v´ | ´x´ | ´w´ | ´y´ | ´z´
<identifier> ::=	<letter>|<identifier> (,<letter> | <integer value>)
<formula>::=	´(´(<attribute>|<numeric value>|<formula>) (<arithmetic operator> <formula> ´)´
<value>::=	<string> | <numeric value>