4
REPORT – *Framework* para Cálculo de Reputações de Agentes de Software Baseado em Testemunhos

Segundo [16], um *framework* é um conjunto de objetos que colaboram com o objetivo de cumprir com um conjunto de responsabilidades para uma aplicação ou um domínio de um subsistema. Em uma outra definição proposta por [18], *framework* é uma arquitetura desenvolvida com o objetivo de se obter a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização. Sabendo da existência de pontos flexíveis e da possibilidade de se obter o reuso do mecanismo de reputação para diferentes aplicações, optamos pela criação de um *framework* para implementar a parte (semi-) centralizada do mecanismo proposto no capítulo 3.

O *framework* REPORT (REPutation-ORganization-Testimony) implementa a parte (semi-) centralizada do mecanismo de reputação e portanto deve ser utilizado para o cálculo de reputações de agentes de software baseado em testemunhos sobre violações de normas [11, 25]. Os sistemas de reputação instanciados a partir do *framework* REPORT deverão ser implementados nas organizações que compõem a aplicação. As organizações serão então capazes de avaliar e armazenar as reputações dos agentes com base nos testemunhos sobre violações de normas. As normas e os papéis que os agentes irão desempenhar são definidos pelas organizações da aplicação. Cada sistema de reputação deverá ser capaz de receber os veredictos dos testemunhos, calcular e manter as reputações dos agentes atualizadas, e disponibilizar a reputação para todos os agentes do sistema multi-agente.

Os agentes que compõem os sistemas instanciados a partir do *framework* REPORT (*Attendant*, *Evaluator* e *Auxiliary*) e suas respectivas ações estão ilustrados na figura 8. O agente *Attendant* recebe requisições do sistema de julgamento e dos agentes da aplicação solicitando a reputação de um determinado agente e fica responsável também pelo recebimento dos veredictos enviados pelo subsistema de julgamento. Ao receber um veredicto, o agente *Attendant* cria um

agente *Evaluator* que classifica o veredicto como uma violação de norma ou falso testemunho. Em seguida, este agente calcula e armazena a reputação do agente considerado culpado. O agente *Auxiliary* é o responsável por manter as reputações atualizadas periodicamente.O REPORT foi desenvolvido utilizando-se o *framework* ASF (*Agent Society Framework*) [27], que tem como propósito criar sistemas multi-agentes implementando o conceito de sociedade de agentes, onde agentes desempenham papéis em organizações.

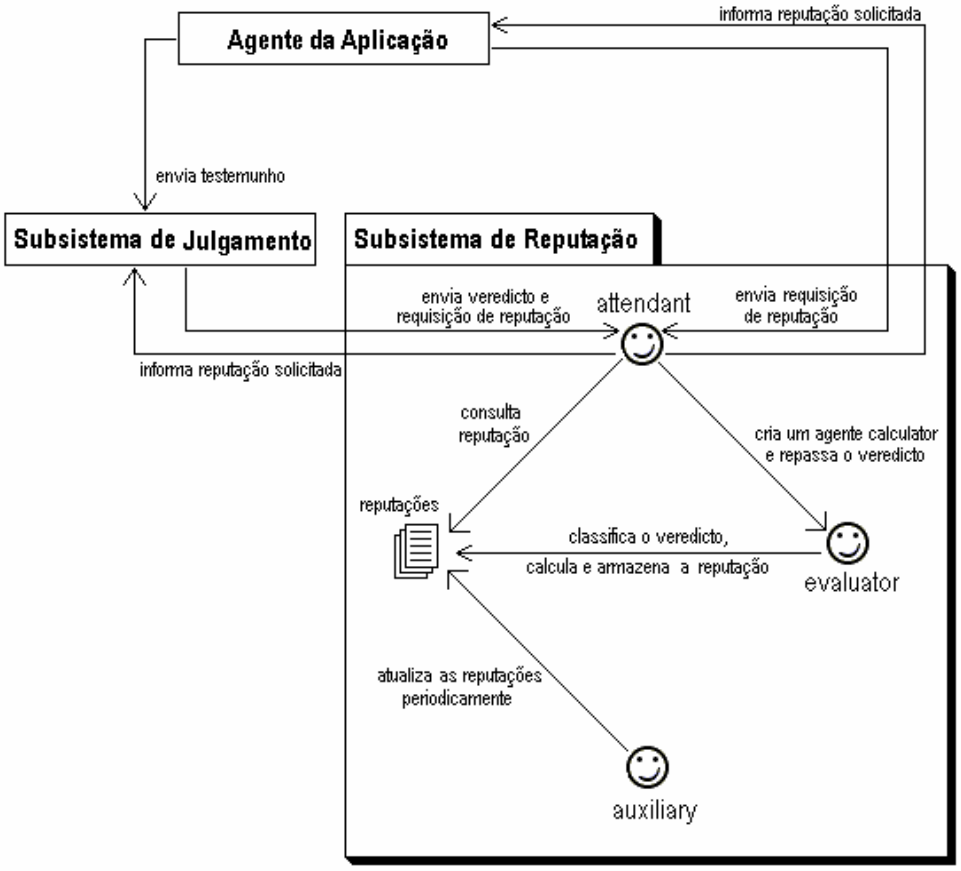

Figura 8. Ações dos agentes do subsistema de reputação, instância do *framework* REPORT.

Descrição dos Agentes do Framework REPORT

Attendant: O agente *Attendant* é responsável pelo recebimento de requisições dos agentes de software e/ou do subsistema de julgamento, que solicitam a reputação de um determinado agente. O agente *Attendant* também fica responsável pelo recebimento de veredictos enviados pelo subsistema de julgamento, sobre

testemunhos de normas violadas. Ao receber um veredicto o agente *Attendant* cria um agente *Evaluator*.

Evaluator: O agente *Evaluator* é responsável por receber o veredicto do agente *Attendant,* classificá-lo como uma violação de norma ou falso testemunho e então alterar a reputação do agente considerado culpado.

Auxiliary: O agente *Auxiliary* é responsável pela atualização de todas as reputações de todos os agentes de tempos em tempos. Sempre que o agente perceber que houve mudança no tempo, este agente irá atualizar a reputação de todos os agentes que violaram normas, desde que a violação ainda esteja influenciando no cálculo da reputação. O tempo é relativo, pode ser um dia, um minuto ou uma transação, por exemplo. O tempo é definido de acordo com a instância do *framework*.

4.1. *Framework* ASF

O *framework* ASF, apresentado na figura 9, é orientado a objetos, desenvolvido com a linguagem Java e deve ser utilizado para implementar sociedades de agentes. Sociedade de agentes são sistemas multi-agentes nos quais é necessário representar agentes desempenhando papéis em organizações.

Usando o *framework* ASF, é possível implementar vários agentes, cada um desempenhando um ou mais papéis em diferentes organizações. O *framework* ASF dá suporte a implementação de agentes, papéis, organizações e ambientes. O *framework* é composto por conjuntos de módulos, cada um representando uma entidade de um sistema multi-agente: módulo agente, módulo papel de agente, módulo organização e módulo ambiente.

Dentre os atributos de um agente estão os seus objetivos e suas crenças. Para alcançar um objetivo o agente deverá executar um dos seus planos. Cada plano é composto por um conjunto de ações e sempre está associado a um objetivo. As ações definem as tarefas de um agente e possibilitam ao agente por exemplo, enviar e receber mensagens, acessar recursos (chamar métodos de objectos) e alterar seu estado.

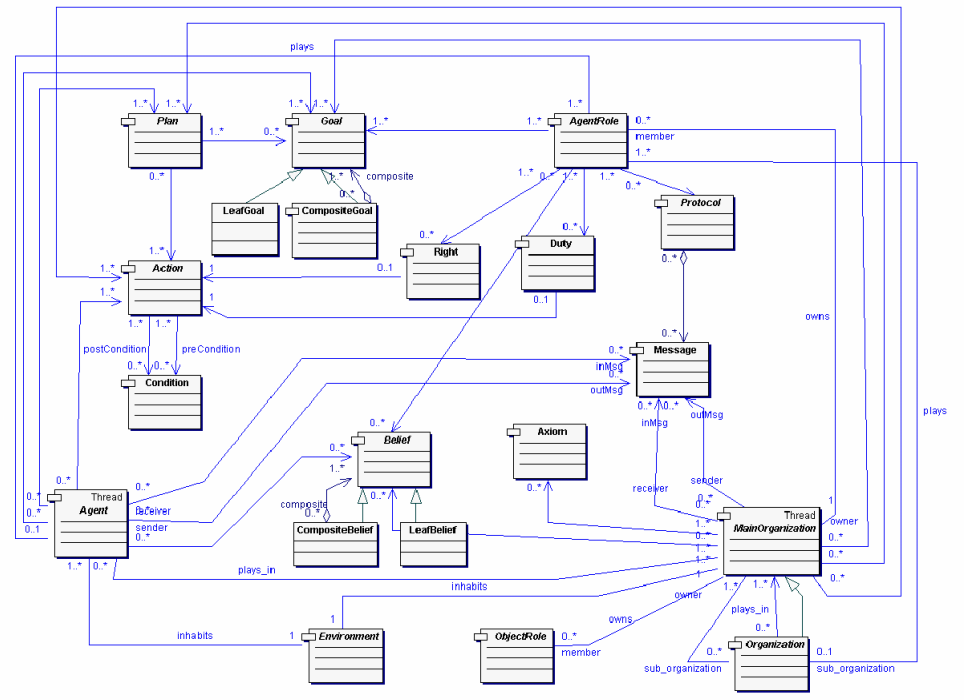

Figura 9. O *Framework* ASF [27]

A organização representa partições ou grupos de entidades, como por exemplo: departamentos, comunidades e sociedades. Todo agente e organização habitam em um ambiente e não podem habitar em mais de um ambiente ao mesmo tempo. No ASF, cada organização define o conjunto de papéis que pode ser desempenhado pelos agentes e estes podem desempenhar diferentes papéis em diferentes organizações. Um agente é implementado como uma entidade *multi-thread* onde cada *thread* está relacionada a um papel desempenhado pelo agente, possibilitando assim que o agente desempenhe diferentes papéis. A propriedade mais importante do papel desempenhado por um agente é a sua definição no contexto de uma organização.

4.2. *Framework* REPORT

O propósito do *framework* REPORT é implementar o mecanismo de reputação (semi-) centralizado para sistemas multi-agentes abertos de larga escala apresentado no capítulo 3 onde uma estrutura hierárquica de organizações seja utilizada. Cada organização do sistema deve ser capaz de definir um conjunto de

papéis e normas cujo objetivo é descrever as responsabilidades de cada papel. Assumimos que os agentes destes sistemas podem desempenhar um ou mais papéis dentro de diferentes organizações e que conhecem o conjunto de normas que devem respeitar durante a sua execução.

4.2.1. Requisitos do *Framework*

O *framework* deve possibilitar aos agentes de software acesso a reputação de outros agentes mesmo sem terem interagido anteriormente com eles. A reputação deverá refletir o comportamento dos agentes que é avaliado pelo sistema de julgamento (externo ao *framework* REPORT) com base nos testemunhos de violações de normas enviados por outros agentes.

Um algoritmo para o cálculo da reputação dos agentes deve estar disponível com base no modelo proposto apresentado no capítulo 3. Usuários do *framework* devem ter a possibilidade de escolher utilizar o algoritmo para cálculo da reputação disponível no *framework*, estender este algoritmo ou implementar outro mais adequado as suas necessidades.

O *framework* deve ser capaz de avaliar o comportamento do agente de acordo com o contexto da norma violada ou do falso testemunho. Sendo assim, esse deve fornecer diferentes tipos de reputação onde seja possível analisar o comportamento do agente em diferentes contextos. O *framework* fornecerá inicialmente três tipos diferentes de reputação por contexto dentro de uma organização e possibilitará que novos tipos sejam implementados. Os três tipos de reputação são: reputação do agente por norma violada, reputação do agente por papel desempenhado e reputação local de um agente, que considera todas as violações de normas e falsos testemunhos de um agente dentro de uma organização. Além disso, o *framework* é capaz de calcular os três tipos de reputação considerando todas as suborganizações de uma hierarquia.

Como o mecanismo de reputação descrito no capítulo 3 prevê que a reputação de um agente aumente periodicamente, o *framework* deve ser flexível o suficiente para permitir que diferentes definições de periodicidade sejam implementadas. A periodicidade básica implementada pelo *framework* se refere a minutos. Utilizando-se está periodicidade, as reputações dos agentes são

atualizadas freqüentemente a cada minuto.

O REPORT deve poder instanciar sistemas de reputação com as seguintes quatro responsabilidades básicas:

(i) Receber os veredictos dos testemunhos, enviados pelo subsistema de julgamento;

(ii) Fazer o atendimento ao subsistema de julgamento e aos agentes de software do sistema, respondendo as requisições sobre as reputações dos agentes;

(iii) Calcular e armazenar as reputações dos agentes de acordo com os veredictos dos testemunhos. Os veredictos são classificados e armazenados como violação de norma ou falso testemunho;

(iv) Atualizar as reputações periodicamente.

4.2.2. Pontos Flexíveis

Existem três pontos flexíveis modelados no REPORT que podem ser estendidos pelo usuário do *framework* permitindo a este fazer as adequações necessárias ao instanciá-lo:

(i) ***Fórmula de cálculo da reputação:*** O *framework* fornece algoritmos básicos para cálculo de reputação (representado pelas fórmulas (6) e (9) do capítulo 3) que podem ser estendidos pelo usuário do *framework*. Estes algoritmos implementam utilizam expressões lineares para fornecer a variação da reputação dos agentes de acordo com as normas violadas e os falsos testemunhos. Estes algoritmos podem ser redefinidos através da extensão da classe abstrata *EvaluateFormula* e implementação dos métodos *formulaReputationDef()* (fornece a reputação do agente levando em consideração as normas violadas) e *formulaReputationWit()* (fornece a reputação dos agentes levando em consideração os falsos testemunhos).

(ii) ***Tempo de atualização da reputação***: A influência de uma violação

na reputação do agente diminui ao longo do tempo. A periodicidade aplicada à reputação é um ponto flexível do *framework*. A implementação básica fornecida relaciona a periodicidade com o passar dos minutos. Porém periodicidades relacionadas, por exemplo, com a ocorrência de um evento ou com o passar dos dias também podem ser implementadas através da extensão da classe *UpdatingReputation* e da implementação do método *getTimeOfUpdating()*.

(iii) ***Tipos de reputação:*** O *framework* oferece três tipos de reputação pré-definidos (*LocalReputation*, *RoleReputation* e *NormaReputation*). Porém, é possível criar novos tipos de reputação estendendo a classe *AgentReputation*.

4.2.2.1. Criando a Fórmula de Cálculo da Reputação

O *framework* oferece uma expressão linear para cálculo da reputação, considerando as variáveis definidas no capítulo 2, descrita detalhadamente a seguir.

A equação (1) avalia a influência da norma n_i sobre a reputação do agente a_j considerando o poder da norma e a percentagem de culpa. Quanto maior o poder da norma e a percentagem de culpa, maior será o valor de $defRepInf(a_j,n_i)$, variável que representa a influência da violação na reputação. Quanto maior o valor desta variável, menor será a reputação do agente (como apresentado na equação 6).

$$defRepInf(a_j,n_i) = normPower(n_i) * blamePercentage(a_j, n_i) \quad (1)$$
$$\text{onde } 0 <= defRepInf(a_j,n_i) <= 1$$

Para exemplificar o cálculo da reputação, vamos utilizar o exemplo do capítulo 2. Agentes importadores e exportadores interagem regulados pelo conjunto de normas da tabela 1.

Normas	*Norm Power*	*Total Time*
n_1. Exportador tem que entregar a carga ao importador em perfeito estado.	0,3	10
n_2. Exportador tem que entregar a carga ao importador no prazo estabelecido.	0,5	30
N_3. Exportador tem que entregar a quantidade que foi combinada com o importador	0,3	15

Tabela 1. Normas reguladoras da interação entre agentes importadores e exportadores

Estamos supondo que um agente exportador foi acusado de violar a norma n_1 e o julgamento definiu o percentual de culpa em 80%, indicando a probabilidade do agente ter violado a norma. Utilizando a equação 1, a influência desta violação na reputação do agente será a seguinte:

$$\text{defRepInf}(\text{exportador}, n_1) = 0{,}3 * 0{,}80 = 0{,}24$$

A equação (2) avalia a influência de n_i de norma na reputação do agente a_j considerando o poder da norma, a percentagem de culpa e o número de dias restantes de influência da norma. Considerando a expressão (2), é possível verificar que *remainingDays*(a_j,n_i) diminuirá com o passar do tempo, diminuindo então o valor da variável *defRepInf*(a_j,n_i) e aumentando a reputação do agente a_j. Quanto mais próximo de 0 (zero) estiver o valor da variável *defRepInf*(a_j,n_i) menor será a influência da norma violada n_i na reputação do agente a_j.

$$\text{defRepInf}(a_j,n_i) = \text{normPower}(n_i) * \text{blamePercentage}(a_j, n_i) * \text{remainingDays}(a_j,n_i) \quad (2)$$

$$\text{where remainingDays}(a_j,n_i) = \frac{\text{totalTime}(n_i) - \text{passedDays}(a_j,n_i)}{\text{totalTime}(n_i)}$$

Considerando uma violação da norma n_1, sabemos que o tempo total de influência de uma violação desta norma na reputação de um agente será de 10 dias, conforme a tabela 1. Desta forma, o valor da variável *remainingDays* no primeiro dia da violação será a seguinte:

$$\text{remainingDays}(\text{exportador}, n_1) = \frac{10 - 0}{10} = 1$$

Aplicando a equação 2 vamos ter:

$$\text{defRepInf}(\text{exportador}, n_1) = 0{,}3 * 0{,}80 * 1 = 0{,}24$$

Após cinco dias, o valor da variável *remainingDays* será:

$$\text{remainingDays}(\text{exportador}, n_1) = \frac{10 - 5}{10} = 0{,}5$$

Desta forma a influência da violação na reputação do agente diminuirá com o passar do tempo. Após cinco dias a influência diminuirá de 0,24 para 0,12.

$$\text{defRepInf}(\text{exportador}, n_1) = 0{,}3 * 0{,}80 * 0{,}5 = 0{,}12$$

A equação (3) adapta o poder da norma considerando a *reincidência* de violação de uma norma. Ao violar uma mesma norma mais de uma vez o poder da norma é aumentado. Desta forma a influência de uma violação na reputação do agente é agravada.

$$\text{normPower}_r(n_i) = \text{normPower}(n_i) * (1/\text{relapse}(n_i)) \qquad (3)$$
$$\text{onde } 0 <= \text{normPower}_r(n_i) <= 1$$

Se um agente violou a norma n_1 mais de uma vez então a influência da violação irá considerar a reincidência adaptando o poder da norma com o fator de reincidência (*relapse*). Considerando que o fator de reincidência para a norma n_1 é de 0,1 podemos observar com os exemplos abaixo como o poder da norma aumenta quando as reincidências acontecem. A partir da oitava violação da norma n_1 o poder da norma atinge o valor máximo (1).

1ª violação → $\text{normPower}_r(n_1) = 0{,}3 * (1/\text{relapse}(n_1)) = 0{,}3 * 1 / 1{,}0 = 0{,}300$
2ª violação → $\text{normPower}_r(n_1) = 0{,}3 * (1/\text{relapse}(n_1)) = 0{,}3 * 1 / 0{,}9 = 0{,}333$
3ª violação → $\text{normPower}_r(n_1) = 0{,}3 * (1/\text{relapse}(n_1)) = 0{,}3 * 1 / 0{,}8 = 0{,}375$
4ª violação → $\text{normPower}_r(n_1) = 0{,}3 * (1/\text{relapse}(n_1)) = 0{,}3 * 1 / 0{,}7 = 0{,}429$
5ª violação → $\text{normPower}_r(n_1) = 0{,}3 * (1/\text{relapse}(n_1)) = 0{,}3 * 1 / 0{,}6 = 0{,}500$
6ª violação → $\text{normPower}_r(n_1) = 0{,}3 * (1/\text{relapse}(n_1)) = 0{,}3 * 1 / 0{,}5 = 0{,}600$
7ª violação → $\text{normPower}_r(n_1) = 0{,}3 * (1/\text{relapse}(n_1)) = 0{,}3 * 1 / 0{,}4 = 0{,}750$
8ª violação → $\text{normPower}_r(n_1) = 0{,}3 * (1/\text{relapse}(n_1)) = 0{,}3 * 1 / 0{,}3 = 1{,}100 = 1$

A equação (4) modifica o poder da norma considerando a *confissão*. Ao confessar a violação um agente poderá ser beneficiado com a diminuição do poder

da norma. Desta forma a influência de uma violação na reputação do agente é atenuada. O fator de confissão é representado por um número dentro do intervalo (0,1].

$$normPower_{rc}(n_i) = normPower_r(n_i) * confession(n_i) \quad (4)$$

A equação (5) fornece a informação parcial de como cada violação de norma irá influenciar a reputação de um agente. Já a equação (6) fornece a reputação do acusado (*defendant's reputation*) reunindo todas as influências parciais. A equação exemplifica a reputação de um agente a_j considerando que ele violou *k* normas. Note que a reputação do acusado pode ser igual a zero se o total de suas influências parciais for igual (ou maior que) um.

$$defRepInf(a_j, n_i) = normPower_{rc}(n_i) * blamePercentage(a_j,n_i) * remainingDays(a_j,n_i) \quad (5)$$

$$defendantRep(a_j) = 1 - \sum_{0<i<=k}[defRepInf(a_j,n_i)], \text{ if } \sum_{0<i<=k}[defRepInf(a_j,n_i)]<=1 \quad (6)$$

A reputação do acusado será igual a 1 se as violações não estiverem mais exercendo influência na reputação. Assim, os valores de reputação variam de 1 a 0 e nós consideramos reputações maiores que 0.5 reputações boas e menores que (ou igual a) 0.5 reputações ruins. Voltando ao nosso exemplo, para obter a influência parcial da violação da norma *n1* na reputação do agente exportador aplicamos a fórmula 5. Em seguida, para obter a sua reputação como réu aplicamos a fórmula 6. Estamos supondo que o agente exportador cometeu apenas esta violação.

$$defRepInf(exportador, n_1) = 0,3 * 0,80 * 1 = 0,24$$

$$defendantRep(a_j) = 1 - defRepInf(exportador, n_1) = 0,76$$

A equação (7) modifica o poder da norma considerando uma testemunha mentirosa, utilizada no cálculo da reputação *Witness Reputation*. Definimos uma testemunha mentirosa como um agente que envia testemunho de violação de

norma que foi considerado falso pelo subsistema de julgamento, i.e., a probabilidade do agente acusado ter violado a norma é menor do que 50%.

$$\text{normPower}_{rw}(n_i) = \text{normPower}_{r}(n_i) * \text{witnessFactor}(n_i) \quad (7)$$
$$\text{onde } 0 <= \text{normPower}_{rw}(n_i) <= 1$$

A equação (9) avalia a reputação do agente testemunha a_j considerando que ele mentiu em k testemunhos sobre violações de normas. A equação agrega todas as influências parciais referentes aos testemunhos mentirosos avaliados de acordo com a equação (8).

$$\text{witRepInf}(a_j, n_i) = \text{normPower}^{rw}(n_i) * \text{blamePercentage}(a_j,n_i) * \text{remainingDays}(a_j,n_i) \quad (8)$$

$$\text{witnessRep}(a_j) = 1 - \sum_{0<i<=k}[\text{witRepInf}(a_j,n_i)], \text{ if } \sum_{0<i<=k}[\text{witRepInf}(a_j,n_i)]<=1 \quad (9)$$

Considerando que o agente exportador nunca cometeu falso testemunho, a sua reputação como testemunha é a melhor possível (1).

A *reputação local* de um agente, equação (10), é aquela obtida pela média dos resultados providos por (6) e (9). A *reputação local* de um agente considera todas as normas violadas e todas as mentiras contadas por um agente em uma organização Org_n.

$$\text{localRep}_{Orgn}(a_j) = [\text{defendantRep}_{Orgn}(a_j) + \text{witnessRep}_{Orgn}(a_j)] / 2 \quad (10)$$

Desta forma, em nosso exemplo a reputação local do agente exportador será a seguinte:

$$\text{localRep}_{Org1}(\text{exportador}) = [0,76 + 1] / 2 = 0,88$$

A equação utilizada para avaliar a reputação por papel é similar à utilizada para calcular as reputações locais, mas agora nós consideramos apenas as normas violadas enquanto o agente está desempenhando um determinado papel *r*, como descrito na equação (11).

$$\text{roleRep}(a_j^r) = [\text{defendantRep}(a_j^r) + \text{witnessRep}(a_j^r)] / 2 \quad (11)$$

Para cada norma do sistema os agentes têm uma reputação por norma que é calculada através da média das equações (5) e (8) como está ilustrado na equação (12). N_i é a norma que está sendo considerada.

$$\text{normRep}(a_j, n_i) = [\text{defendantRep}(a_j,n_i) + \text{witnessRep}(a_j,n_i)] / 2 \qquad (12)$$

A reputação global de um agente é obtida pela média das reputações avaliadas em Org_x e em todas as suas suborganizações, conforme a equação (13). Já a reputação global por papel de um agente é obtida pela média das reputações avaliadas enquanto o agente está desempenhando um determinado papel em Org_x e em todas as suas suborganizações, conforme a equação (14).

A reputação global por norma de um agente é obtida pela média das reputações avaliadas de acordo com a violação de uma determinada norma em Org_x e em todas as suborganizações, conforme a equação (15). Nas organizações que não possuem suborganizações as reputações globais são iguais às reputações locais, conforme as equações (16, 17 e 18).

$$\text{globalRep}_{\text{Orgx}}(a_j) = \text{localRep}_{\text{Orgx}}(a_j) \qquad (16)$$

$$\text{globalRoleRep}_{\text{Orgx}}(a_j) = \text{roleRep}_{\text{Orgx}}(a_j) \qquad (17)$$

$$\text{globalNormRep}_{\text{Orgx}}(a_j, n_i) = \text{normRep}_{\text{Orgx}}(a_j, n_i) \qquad (18)$$

data	Violação 1: n1' 01/05/2006				Violação 2: n1'' 04/05/2006				parciais n1' +n1''	Defendant Rep
	relapse	blame Percent.	remaining Days	parcial n1' defReplnf	relapse	blame Percent.	remaining Days	parcial n1'' defReplnf		
30/04/06									0.000	1.000
01/05/06	1	0.8	1	0.400					0.400	0.600
02/05/06	1	0.8	0.9	0.360					0.360	0.640
03/05/06	1	0.8	0.8	0.320					0.320	0.680
04/05/06	1	0.8	0.7	0.280	0.9	0.7	1	0.389	0.669	0.331
05/05/06	1	0.8	0.6	0.240	0.9	0.7	0.9	0.350	0.590	0.410
06/05/06	1	0.8	0.5	0.200	0.9	0.7	0.8	0.311	0.511	0.489
07/05/06	1	0.8	0.4	0.160	0.9	0.7	0.7	0.272	0.432	0.568
08/05/06	1	0.8	0.3	0.120	0.9	0.7	0.6	0.233	0.353	0.647
09/05/06	1	0.8	0.2	0.080	0.9	0.7	0.5	0.194	0.274	0.726
10/05/06	1	0.8	0.1	0.040	0.9	0.7	0.4	0.156	0.196	0.804
11/05/06	1	0.8	0	0.000	0.9	0.7	0.3	0.117	0.117	0.883
12/05/06					0.9	0.7	0.2	0.078	0.078	0.922
13/05/06					0.9	0.7	0.1	0.039	0.039	0.961
14/05/06					0.9	0.7	0	0.000	0.000	1.000
15/05/06									0.000	1.000

Informações da norma n1: normPower=0.5, totalTime=10 dias, confession=1 e fator de reincidência=0.1

Tabela 2. Cálculo das influências parciais e *defendant's reputation* do agente *A* segundo as duas violações da norma *n1*

A tabela 2 apresenta um exemplo de duas violações da mesma norma, feitas por um agente em épocas diferentes (primeira violação no dia 01/05/06 e a segunda no dia 04/05/06). A tabela mostra as informações utilizadas para avaliar as influências parciais (equação 5) e a reputação do agente como acusado de violar normas (equação 6).

Em nosso exemplo o subsistema de julgamento decidiu que a probabilidade do agente acusado ser culpado é de 80% para a primeira violação e 70% para a segunda violação. O fator de reincidência que estamos utilizando é 0.1 e o tempo em que a norma influencia a reputação do agente é de 10 dias. Analisando a tabela nota-se que entre os dias 04/05 e 10/05 a reputação do agente está sofrendo influência das duas violações.

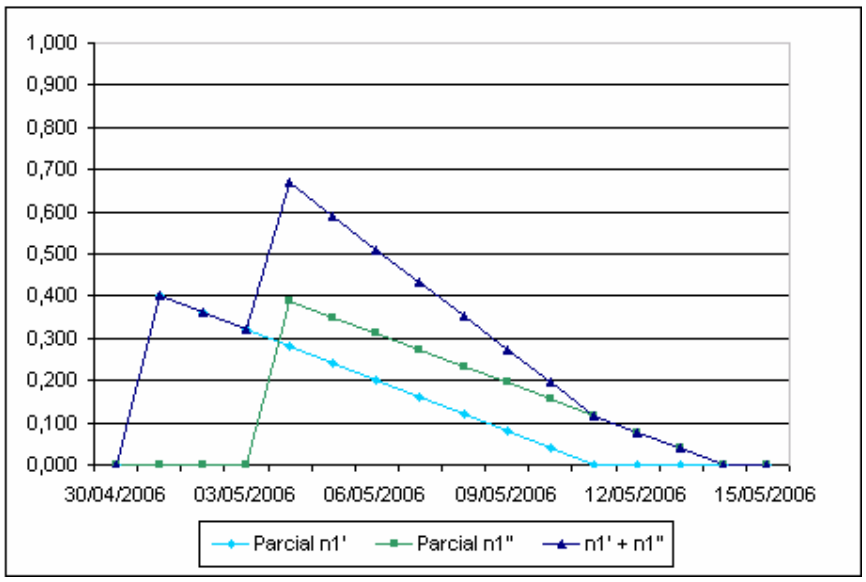

Figura 10. Influências Parciais do agente *A* segundo as violações da norma *n*

As influências parciais, que correspondem aos valores resultantes das violações de normas, podem ser observadas no gráfico da figura 10. Observe que as parciais diminuem na medida em que os dias passam. Note também que ao violar a norma *n1* pela segunda vez, no dia 04/05, a reputação do agente será modificada considerando as duas violações, através do somatório de *n1'* e *n1"*.

Para exemplificar os diferentes tipos de reputação vamos considerar as duas violações da norma *n1,* apresentadas na tabela 2, e supor que o agente cometeu a primeira violação enquanto desempenhava o papel *r1* e a segunda enquanto desempenhava o papel *r2*. Vamos supor também que o mesmo agente cometeu

uma terceira violação, da norma *n2,* enquanto desempenhava o papel *r2*. A tabela 3 exemplifica o cálculo da reputação do agente segunda esta violação.1

data	Violação 3: n2		07/05/2006	parcial n2 defRepInf	Defendant Rep
	relapse	blame Percentage	remaining Days		
30/04/2006				0.000	1.000
01/05/2006				0.000	1.000
02/05/2006				0.000	1.000
03/05/2006				0.000	1.000
04/05/2006				0.000	1.000
05/05/2006				0.000	1.000
06/05/2006				0.000	1.000
07/05/2006	1	0.6	1.00	0.180	0.820
08/05/2006	1	0.6	0.88	0.158	0.843
09/05/2006	1	0.6	0.75	0.135	0.865
10/05/2006	1	0.6	0.63	0.113	0.888
11/05/2006	1	0.6	0.50	0.090	0.910
12/05/2006	1	0.6	0.38	0.068	0.933
13/05/2006	1	0.6	0.25	0.045	0.955
14/05/2006	1	0.6	0.13	0.023	0.978
15/05/2006	1	0.6	0.00	0.000	1.000

Informações da norma n2: normPower=0.3, totalTime=8 dias e confession=1

Tabela 3. Cálculo da influência parcial e *Defendant's Reputation* do agente *A* segundo a violação da norma *n2*

A tabela 4 apresenta os tipos de reputações obtidos considerando contextos diferentes. No cálculo de reputação local do agente (equação 10), a reputação do agente enquanto testemunha é igual a 1, pois este agente nunca forneceu falso testemunho, enquanto que sua reputação como acusado depende das três violações de norma que cometeu. Até o sexto dia do mês a reputação local do agente só estava sendo influenciada pelas violações da norma *n1* e portanto o valor da reputação local era igual ao valor da reputação segundo a norma *n1*. Entre os dias 07 e 10 a reputação local do agente está sendo influenciada pelas 3 violações.

Outro ponto interessante de análise da tabela 4 está na relação entre a reputação local e a reputação por papel. Até o terceiro dia do mês, a reputação local é igual à reputação por papel *r1*, pois este foi o único papel sendo desempenhado enquanto o agente violava as normas até este dia.

Entre os dias 04 e o dia 10, a reputação local do agente sofre influência das três violações cometidas enquanto desempenhava dois papéis (*r1* e *r2*). A partir do dia 11, a reputação local é igual à reputação por papel *r2* pois este foi o único papel desempenhado enquanto o agente violava as normas *n1* (pela segunda vez) e

a norma *n2*. O gráfico da figura 11 ilustra as variações de cada tipo de reputação ao longo dos dias.

data	Local Rep (A)	Role Rep (A^{r1})	Role Rep (A^{r2})	Norm Rep (A, n1)	Norm Rep (A, n2)
30/04/2006	1,000	1,000	1,000	1,000	1,000
01/05/2006	0,800	0,800	1,000	0,800	1,000
02/05/2006	0,820	0,820	1,000	0,820	1,000
03/05/2006	0,840	0,840	1,000	0,840	1,000
04/05/2006	0,666	0,860	0,806	0,666	1,000
05/05/2006	0,705	0,880	0,825	0,705	1,000
06/05/2006	0,744	0,900	0,844	0,744	1,000
07/05/2006	0,694	0,920	0,774	0,784	0,910
08/05/2006	0,745	0,940	0,805	0,823	0,921
09/05/2006	0,795	0,960	0,835	0,863	0,933
10/05/2006	0,846	0,980	0,866	0,902	0,944
11/05/2006	0,897	1,000	0,897	0,942	0,955
12/05/2006	0,927	1,000	0,927	0,961	0,966
13/05/2006	0,958	1,000	0,958	0,981	0,978
14/05/2006	0,989	1,000	0,989	1,000	0,989
15/05/2006	1,000	1,000	1,000	1,000	1,000

Tabela 4. Valores dos diferentes tipos de reputação do agente *A*

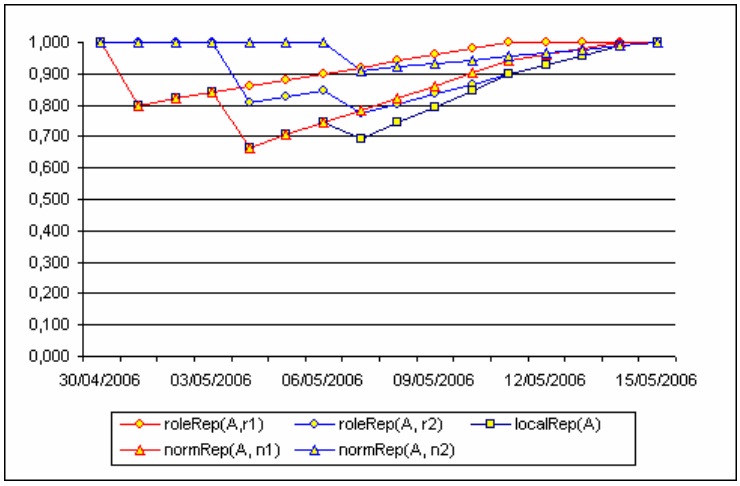

Figura 11. Os diferentes tipos de reputação do agente *A* ao longo dos dias

Criando uma nova fórmula

Se o usuário do *framework* não desejar utilizar a expressão linear para cálculo da reputação já fornecida pelo *framework* é necessário criar uma classe concreta estendendo a classe abstrata *EvaluateFormula* e implementar o cálculo da reputação do agente. O método *formulaReputation* define a reputação de um agente considerando um conjunto de normas violadas e mentiras contadas por um agente em uma organização, de acordo com as equações (10), (11) e (12)

apresentadas no capítulo 3. Os métodos *formulaReputationDef()* e *formulaReputationWit()* deverão ser implementados, informando as fórmulas de cálculo para *defendant* e *witness reputations*, respectivamente.

Como visto anteriormente, a reputação de um agente é avaliada através das influências exercidas pelas normas violadas e pelos falsos testemunhos. A influência de cada norma violada é calculada de acordo com a equação (5) apresentada no capítulo 3 e implementada pelo método *formulaInfluenceDefendant()* da classe *EvaluateFormula*. A influência de cada falso testemunho é calculada de acordo com a equação (8) apresentada no capitulo 3 e implementada pelo método *formulaInfluenceWitness()*.

Embora o *framework* já ofereça as implementações dos métodos *formulaInfluenceDefendant()* e *formulaInfluenceWitness()*, eles também podem ser modificados na implementação da nova classe que estenderá a classe *EvaluateFormula*. É possível, por exemplo, considerar novos dados sobre o comportamento dos agentes, acrescentando novas variáveis nas fórmulas.

Os agentes *Evaluator* e *Auxiliary* são responsáveis pelo cálculo da reputação. O primeiro altera as reputações do agente imediatamente ao receber um veredicto do subsistema de julgamento. O segundo mantém as reputações atualizadas recalculando-as de tempos em tempos. Para calcular a reputação de um agente, os agentes *Evaluator* e *Auxiliary* chamam o método *updateReputation* das extensões da classe *AgentReputation*, detalhada na seção 4.2.2.3. Estas extensões deverão utilizar as implementações da classe *EvaluateFormula* para obterem as influências das violações e dos falsos testemunhos e são responsáveis por fornecer a reputação do agente de acordo com um determinado contexto. O *framework* oferece três contextos distintos, *localReputation*, *RoleReputation* e *normReputation*, porém outros contextos podem ser definidos. As implementações dos métodos abstratos *isViolationInfluence()* e *isFalseTestimonyInfluence()* têm como objetivo reunir as influências de um agente de acordo com o contexto definido para cada tipo de reputação. A figura 12 apresenta o diagrama com as classes envolvidas no cálculo da reputação.

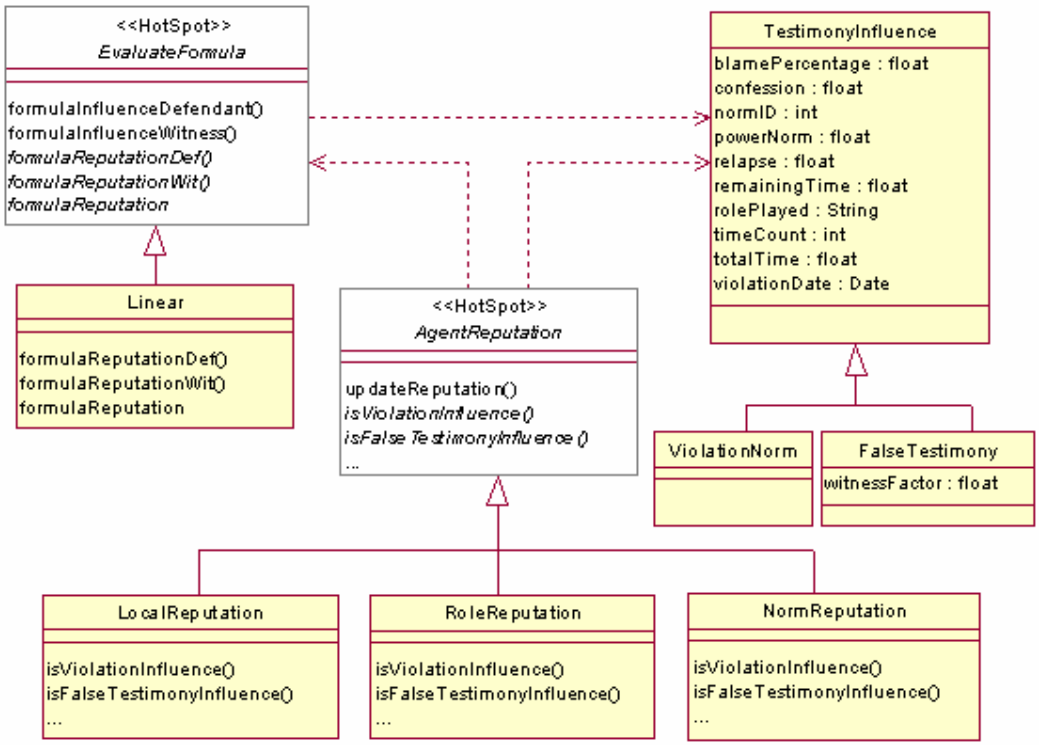

Figura 12. Classes envolvidas no cálculo da reputação

Com o objetivo de exemplificar as diferentes implementações dos métodos abstratos *formulaReputationDef()* e *formulaReputationWit()* considere as implementações das figuras 13 e 14.

```
public class Linear extends EvaluateFormula{
    public float formulaReputationDef(float totRepInfDef){
        if(totRepInfDef > 1){
            totRepInfDef = 1;
        }
        float defendantRep = (float)(1-totRepInfDef);
        return defendantRep;
    }
    public float formulaReputationWit(float totRepInfWit)  {
        if(totRepInfWit > 1){
            totRepInfWit = 1;
        }
        float witnessRep = (float)(1-totRepInfWit);
        return witnessRep;
    }
    public float formulaReputation(float defendantRep, float witnessRep) {
        float reputation = (((defendantRep + witnessRep) / 2));
        return reputation;
    }
}
```

Figura 13. Fórmula linear para o cálculo da reputação.

A classe *Linear*, implementação básica fornecida com o *framework*, estende a classe *EvaluateFormula* implementando os métodos abstratos

formulaReputationDef() e *formulaReputationWit()* como uma equação linear (figura 14). Já a classe *Exponencial* estende a classe *EvaluateFormula* implementando os mesmos métodos como uma equação exponencial (figura 14).

```
public class Exponencial extends EvaluateFormula{
   public float formulaReputationDef(float totRepInfDef){
       if(totRepInfDef > 1){
           totRepInfDef = 1;
       }
       float defendantRep = (float)(0.2 *
            (Math.pow(2.7182,(1.792*((1-totRepInfDef) + 0))))-0.2);
       return defendantRep;
   }
   public float formulaReputationWit(float totRepInfWit) {
       if(totRepInfWit > 1){
           totRepInfWit = 1;
       }
       float witnessRep = (float)(0.2 *
            (Math.pow(2.7182,(1.792*((1-totRepInfWit) + 0))))-0.2);
       return witnessRep;
   }
   public float formulaReputation(float defendantRep, float witnessRep) {
       float reputation = (((defendantRep + witnessRep) / 2));
       return reputation;
   }
}
```

Figura 14. Fórmula exponencial para o cálculo da reputação.

No gráfico da figura 15 exemplificamos a diferença da variação da reputação de um agente considerando as duas implementações do método *formulaReputationDef()*. As mesmas informações de violações de normas foram fornecidas para as duas implementações. É possível notar que a fórmula exponencial faz com que a reputação do agente seja mais drasticamente modificada quando ocorre uma violação. Na primeira violação a reputação do agente baixa de 1 para 0,142, utilizando-se a fórmula exponencial, e de 1 para 0,300 utilizando-se a fórmula linear. Além disso, é possível notar também que a recuperação da reputação do agente é mais lenta utilizando-se a fórmula exponencial que a fórmula linear. Após a primeira violação e considerando o mesmo espaço de tempo, a fórmula linear proporciona um aumento de 0,140 e a exponencial um aumento de apenas 0,098. Sendo assim a fórmula exponencial é mais adequada que a fórmula linear quando se deseja aplicar punições severas a violações de normas. Em sistemas críticos, por exemplo, a fórmula exponencial é mais adequada do que a fómula linear pois imporia mais rigor na inibição de violações.

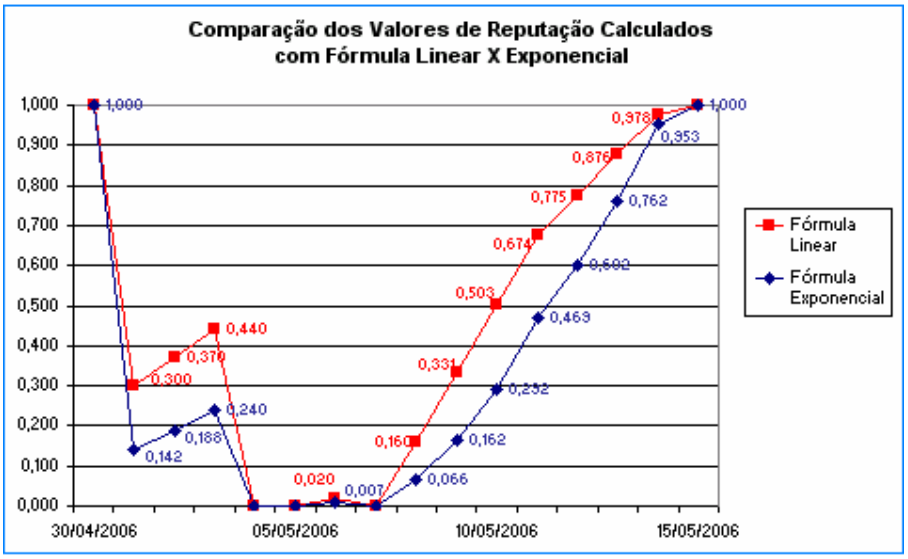

Figura 15. Comparação dos valores de reputação: fórmula linear X exponencial

4.2.2.2. Definindo o Tempo de Atualização da Reputação

Cada norma é responsável por informar por quanto *tempo* a violação desta norma ou o falso testemunho irá influenciar a reputação dos agentes através do atributo *totalTime*. Se o valor informado para este atributo for zero significa que, se houver uma violação desta norma ou falso testemunho, ela não irá influenciar a reputação do agente. Se o valor for diferente de zero, este corresponderá ao tempo total de influência da violação ou falso testemunho na reputação do agente.

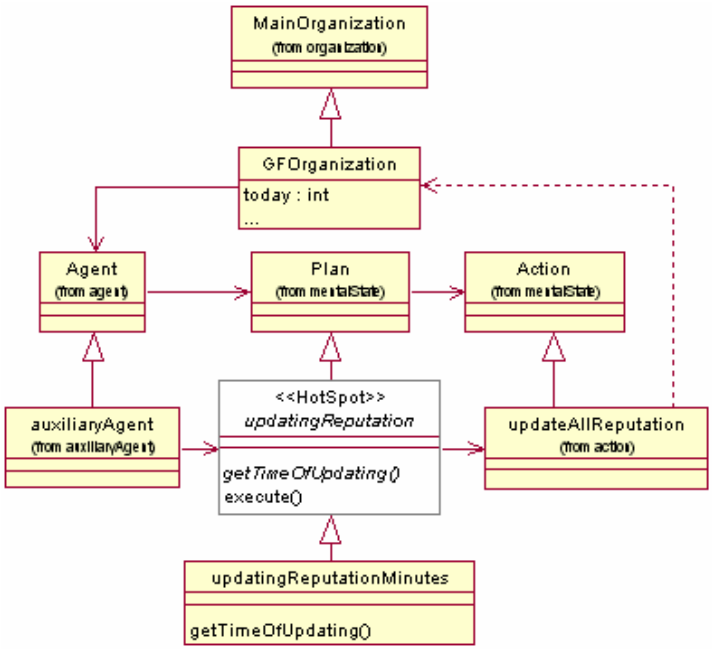

Figura 16. Classes envolvidas na definição do tempo de atualização da reputação.

Por exemplo, se o tempo total de uma norma for igual a trinta (*TotalTime* = 30) significa que uma violação desta norma ou falso testemunho irá influenciar a reputação do agente durante 30 *tempos*. É a variável *RemainingTime* que armazena a informação de quanto tempo a norma ainda influenciará a reputação do agente.

A definição do tempo deverá representar o momento mais adequado para um agente recuperar a sua reputação. Um *tempo* pode corresponder a um dia, uma hora, um minuto, uma sessão, um evento ou uma medida qualquer definida pela instância do *framework*, para efetuar a atualização das reputações. O diagrama da figura 16 apresenta as classes envolvidas na definição do tempo de atualização da reputação.

O agente *Auxiliary* é quem determina este *tempo*, ou melhor, a periodicidade de atualização. A organização conhece o tempo atual através do atributo *today*. Um *tempo* termina quando o valor deste atributo for diferente do valor retornado pelo plano *UpdatingReputation*. A figura 17 apresenta a implementação deste plano. Ao instanciar o *framework* é necessário estender a classe *updatingReputation()* implementando o método *getTimeOfUpdating()*. Este método determina a periodicidade na qual as reputações deverão ser atualizadas.

```
public abstract class UpdatingReputation extends Plan {

   public abstract int getTimeOfUpdating(GFOrganization org);

   public void execute (AgentRole role)
   {
      //Fica em loop tentando atingir o objetivo
      GFOrganization org = (GFOrganization)role.getOwner();
      int timeOfUpdating = this.getTimeOfUpdating(org);
      if (org.today != timeOfUpdating)
      {
          org.setLastUpdate(timeOfUpdating);
          UpdateAllReputation action = new UpdateAllReputation();
          action.execute(org, timeOfUpdating);
      }

    }
}
```

Figura 17. Código do plano *updatingReputation* do agente *AuxiliarAgent*.

A classe concreta *updatingReputationMinutes* implementa o método *getTimeOfUpdating*. Este método retorna o tempo atual, representado pela

variável *timeOfUpdating*. Neste exemplo, conforme a figura 18, determinamos que o tempo para a atualização das reputações deve ser a cada minuto. Quando o valor retornado por este método mudar então o agente *AuxiliaryAgent* irá executar a ação *updateAllReputation*, procedendo a atualização de todas as reputações de todos os agentes.

```
public class updatingReputationMinutes extends UpdatingReputation {

    public int getTimeOfUpdating(GFOrganization org) {
        Calendar calendar = Calendar.getInstance();
        int timeOfUpdating = calendar.get(Calendar.MINUTE);
        return timeOfUpdating;
    };

}
```

Figura 18. Código do método *getTimeOfUpdating* da classe concreta *updatingReputationMinutes*

4.2.2.3. Criando Novos Tipos de Reputação

Embora o *framework* já ofereça as reputações por contexto *LocalReputation*, *RoleReputation* e *NormaReputation*, é possível criar novos tipos de reputação. Pode ser importante para uma aplicação, por exemplo, definir uma reputação por norma para um determinado papel. Outro exemplo seria criar uma reputação relacionada a um subconjunto de normas.

Os novos tipos de reputações podem ser especificados ao criar uma organização, conforme o exemplo de implementação da figura 19. No exemplo foi criada uma reputação denominada *DeliveryReputation*, que está relacionada às duas normas definidas na organização. Um *array* contendo as normas relacionadas, o nome da classe que contém as fórmulas de cálculo para este tipo de reputação e o nome da classe concreta referente ao novo tipo de reputação (que especializa a classe abstrata *AgentReputation*), são os atributos que devem ser informados na organização (e definidos no objeto *PersonalizedReputationInformation*). Através destas informações o *framework* será capaz de calcular as reputações e mantê-las atualizadas periodicamente.

Ao especializar a classe *AgentReputation*, os métodos abstratos *isViolationInfluence* e *isFalseTestimonyInfluence* deverão ser implementados. O

algoritmo utilizado nestes métodos tem como objetivo grupar as violações de normas e os falsos testemunhos, respectivamente, que deverão ser consideradas no cálculo deste tipo de reputação. No exemplo da figura 19 o método percorre o *array* contendo a relação de normas que influenciam este novo tipo de reputação. Se o agente violou a norma ou prestou falso testemunho então o algoritmo retorna *true*.

```
public class DeliveryReputation extends AgentReputation {
    protected boolean isViolationInfluence(ViolationNorm VN){

        List CRNormList = getPrClass().getNormsList();
        for(Iterator i = CRNormList.iterator(); i.hasNext();)
        {
            Norm CR = (Norm)i.next();
            if(VN.getviolatedNorm()==CR.getId())
            {
                return true;
            }
        }
        return false;
    }
    protected boolean isFalseTestimonyInfluence(FalseTestimony FT){
        List CRNormList = getPrClass().getNormsList();
        for(Iterator i = CRNormList.iterator(); i.hasNext();)
        {
            Norm CR = (Norm)i.next();
            if(FT.getviolatedNorm()==CR.getId())
            {
                return true;
            }
        }
        return false;
    }
}
```

Figura 19. Implementação de novo tipo de reputação.

4.2.3.
Diagrama de Casos de Uso

A figura 20 apresenta o diagrama de casos de uso do *framework*. Os detalhes de cada caso de uso são apresentados em seguida.

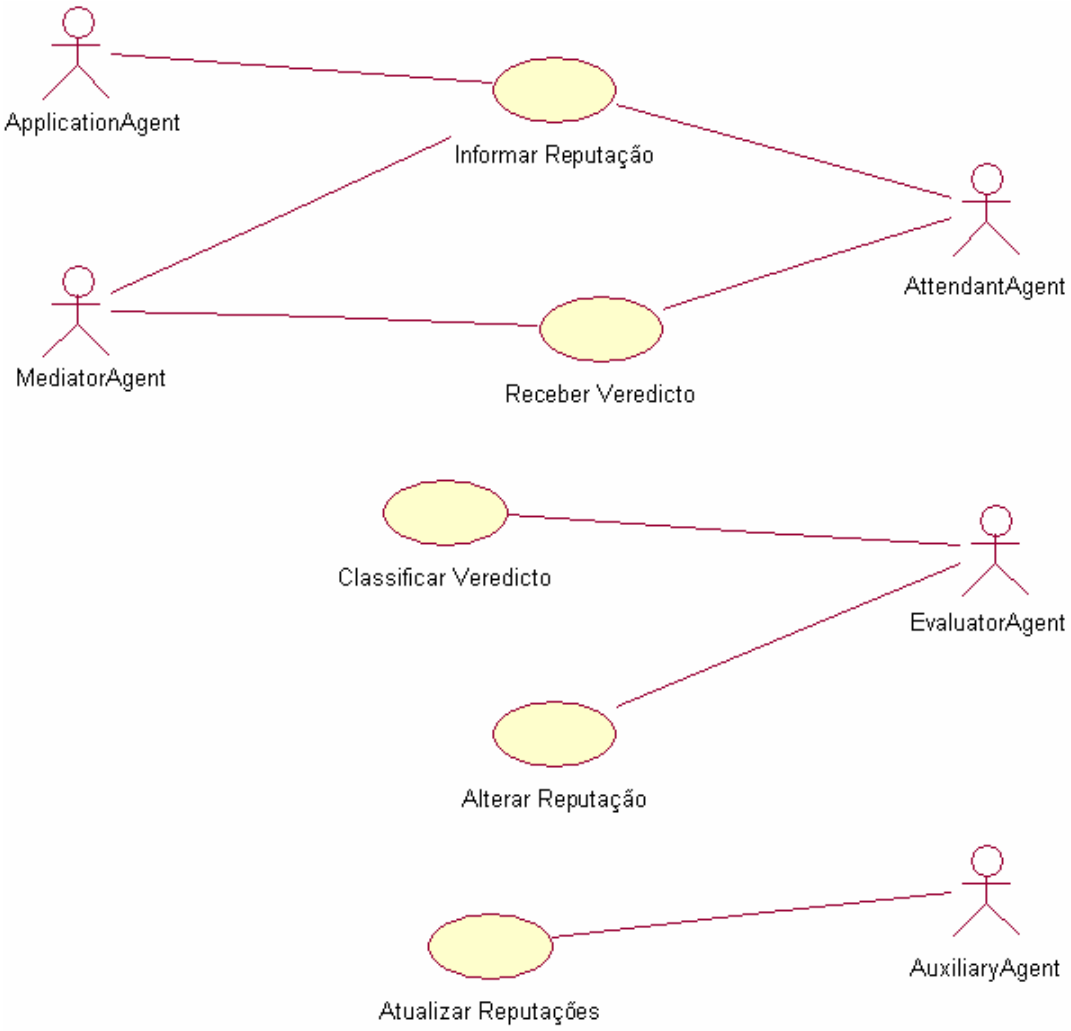

Figura 20. Diagrama de Casos de Uso do *Framework* REPORT

4.2.3.1. Informar Reputação

Ator Primário: *MediatorAgent* ou *ApplicationAgent*

Objetivo: O objetivo deste caso de uso é fornecer a reputação de um determinado agente para quem solicitou, ou seja, qualquer agente de software da aplicação ou o agente *MediatorAgent* do sistema de julgamento. Podem ser solicitadas as reputações: *LocalReputation*, *RoleReputation*, *NormReputation* e também as reputações personalizadas criadas especificamente para a aplicação instanciada. Além dessas reputações também é possível solicitar reputações que envolvem outras organizações, representadas pelas reputações *GlobalReputation*, *GlobalRoleReputation* e *GlobalNormReputation*. O agente responsável pela execução desta ação é o *Attendant*.

Pré-condições: Nenhuma.

Fluxo Normal:

1. Este caso de uso é iniciado quando o agente *MediatorAgent*, do subsistema de julgamento, ou qualquer outro agente da aplicação instanciada envia uma mensagem ao agente *Attendant*, solicitando a reputação de um agente. Existe uma *performative* diferente para cada tipo de reputação. A mensagem deverá conter um objeto da classe correspondente ao tipo de reputação desejado.
2. O agente *Attendant* recebe a mensagem e identifica, de acordo com a *performative*, qual o tipo de reputação que foi solicitado.
3. O agente *Attendant* faz a pesquisa na lista de reputações e envia uma resposta ao agente, através de uma mensagem com um objeto referente ao tipo de reputação anexado, contendo a reputação (um valor entre 0 e 1, inclusive), a data da última violação, a quantidade de violações e a lista de todas as violações deste agente. Se a solicitação for de uma reputação global são enviadas mensagens aos agentes *Attendants* das suborganizações.
4. O agente *Attendant* fica aguardando novas requisições.

Fluxo Alternativo: Quando o agente não possui reputação.

A.3. *Attendant* faz a pesquisa na lista de reputações e retorna nulo. Se não existe nenhuma reputação para um determinado agente, significa que ele nunca violou normas, nem prestou falsos testemunhos. O valor da reputação é –1.

A.4. *Attendant* retorna reputação com valor máximo, 1.

Atores Secundários: *ApplicationAgent*.

4.2.3.2. ReceberVeredicto

Ator Primário: *MediatorAgent*

Objetivo: O agente *Attendant*, além de fornecer a reputação dos agentes, também fica responsável pelo recebimento de veredictos vindo do sistema de julgamento, sobre testemunhos de violações de normas do sistema. O veredicto é gerado pelo subsistema de julgamento. O agente *MediatorAgent* envia uma mensagem para o agente *Attendant* informando o resultado do julgamento. Tal mensagem contém informações sobre o testemunho e o veredicto. O agente *Attendant* recebe as informações e cria um agente *Evaluator* que ficará responsável pelo armazenamento destas informações e pelo cálculo da reputação do agente acusado ou do agente que testemunhou a violação, dependendo do resultado do julgamento. A criação de um agente *Evaluator* libera o agente *Attendant* para que ele possa atender outras solicitações.

Pré Condições: Nenhuma.

Fluxo Normal:

1. Este caso de uso é iniciado quando o agente *MediatorAgent* envia uma mensagem ao agente *Attendant* contendo o testemunho e o veredicto, ou seja, um resultado de um julgamento sobre uma violação de norma.
2. O agente *Attendant* recebe a mensagem e cria um agente *Evaluator* repassando as informações.
3. O agente *Attendant* fica aguardando novas mensagens

Fluxo Alternativo:

Nenhum.

4.2.3.3. Classificar Testemunho

Ator Primário: *Attendant*

Objetivo: Este caso de uso tem como objetivo classificar o testemunho recebido do subsistema de julgamento baseado no veredicto.

Precondições: Agente *Attendant* deverá ter recebido uma mensagem do agente *MediatorAgent*, do subsistema de julgamento, contendo o resultado de um julgamento.

Fluxo Normal:

1. Este caso de uso é iniciado quando o agente *Evaluator* é criado, imediatamente após o agente *Attendant* receber um veredicto.
2. O agente *Evaluator* recebe as informações do veredicto, que contém a data, a norma que foi violada, o agente acusado e o agente enviou o testemunho, o veredicto, se houve confissão e o percentual de culpa.
3. Se o agente acusado foi considerado culpado então o testemunho é classificado como uma violação de norma. Se o agente foi considerado inocente, então o testemunho é classificado como falso.
4. O agente *Evaluator* recupera através da identificação da norma, todos os parâmetros utilizados no cálculo da reputação, *NormPower*, *TotalTime*, *RemainingTime* e cria um objeto da classe *ViolationNorm* ou *FalseTestimony*.

4.2.3.4. Alterar Reputação

Ator Primário: *Evaluator*

Objetivo: Este caso de uso tem como objetivo alterar a reputação do agente acusado de violar uma norma ou do agente que enviou o testemunho, de acordo com o resultado do julgamento.

Precondições: Nenhuma.

Fluxo Normal:

1. Este caso de uso é iniciado após o agente *Evaluator* classificar o veredicto.
2. O agente *Evaluator* verifica se o agente condenado já violou normas ou prestou falso testemunho anteriormente.
3. Se for a primeira violação então o agente condenado é incluído nas listas de reputações.
4. A violação é incluída na lista de violações do agente.
5. O fator de reincidência é calculado, contando o número de vezes que o agente violou esta norma.
6. A reputação do agente condenado é atualizada.
7. Agente *Evaluator* atingiu seu objetivo e encerra sua execução (“morre”)

4.2.3.5. Atualizar Reputações

Ator Primário: *Auxiliary*

Objetivo: Este caso de uso é executado pelo agente *Auxiliary*. Este agente ficará verificando se é o momento de fazer a atualização das reputações dos agentes que já violaram normas, considerando somente as violações ou falsos testemunhos que ainda influenciam na reputação dos agentes.

Precondições: Nenhuma.

Fluxo Normal:

1. Agente *Auxiliary* executa o método *getTimeOfUpdating* que retorna o *tempo* atual. O *tempo* é definido de acordo com a aplicação.
2. Se o valor do *tempo* mudou então o agente *Auxiliary* inicia o processo de atualização da reputação, descrito abaixo, para cada tipo de reputação:
3. O agente *Auxiliary* obtém a lista de reputação.
4. Em seguida obtém o nome da classe que contém as fórmulas para cálculo da reputação.
5. Para cada agente da lista é feito o somatório de todas as violações que

influenciam o tipo de reputação que está sendo calculado.

6. Para cada violação é feito o cálculo dos dias restantes.
7. A reputação do agente é atualizada.
8. O processo é repetido para as demais listas de reputação.

4.2.4.
Diagrama de Classes

Na figura 21 estão ilustradas todas as classes que compõem o *framework* REPORT.

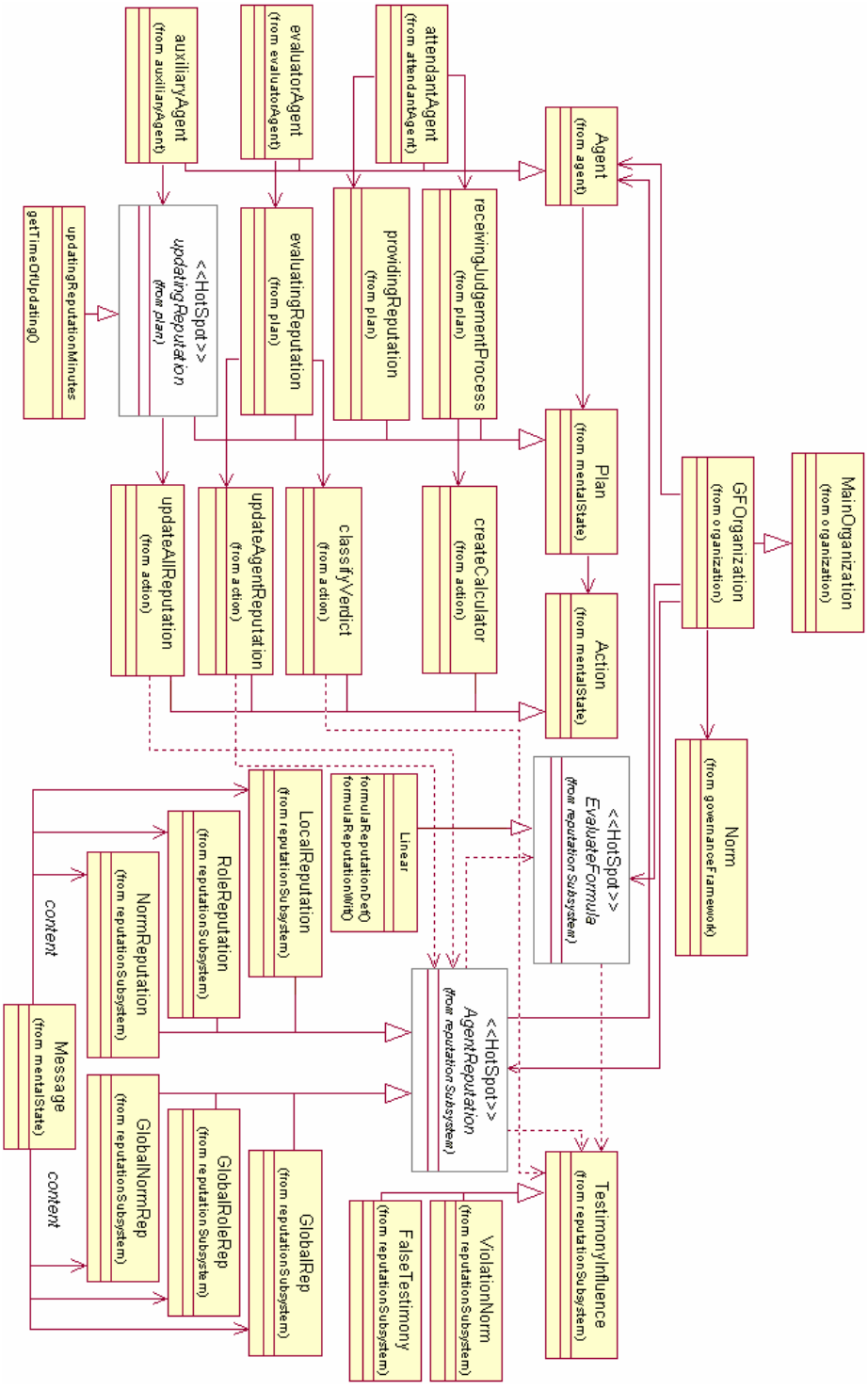

Figura 21. Diagrama de Classes do *Framework* REPORT

As classes que representam os pontos flexíveis do *framework* estão marcadas com o esteriótipo <<hot-spot>>. As classes que estendem os pontos flexíveis são as implementações básicas dos já oferecidas pelo *framework*.

4.2.5. Diagramas de Seqüência

Nesta seção apresentamos diagramas de seqüência para quatro dos cindo casos de uso do *framework*: informar reputação, classificar veredicto, receber veredicto e atualizar reputação.

4.2.5.1. Informar Reputação

Os agentes da aplicação podem enviar mensagens solicitando a reputação de outro agente. O diagrama de seqüência da figura 22 apresenta os métodos utilizados pelo *framework* para responder a uma requisição feita por um agente solicitando a reputação local de outro agente, enquanto que o diagrama da figura 23 apresenta uma requisição feita por um agente solicitando a reputação global de outro agente. Neste caso as suborganizações também são consultadas.

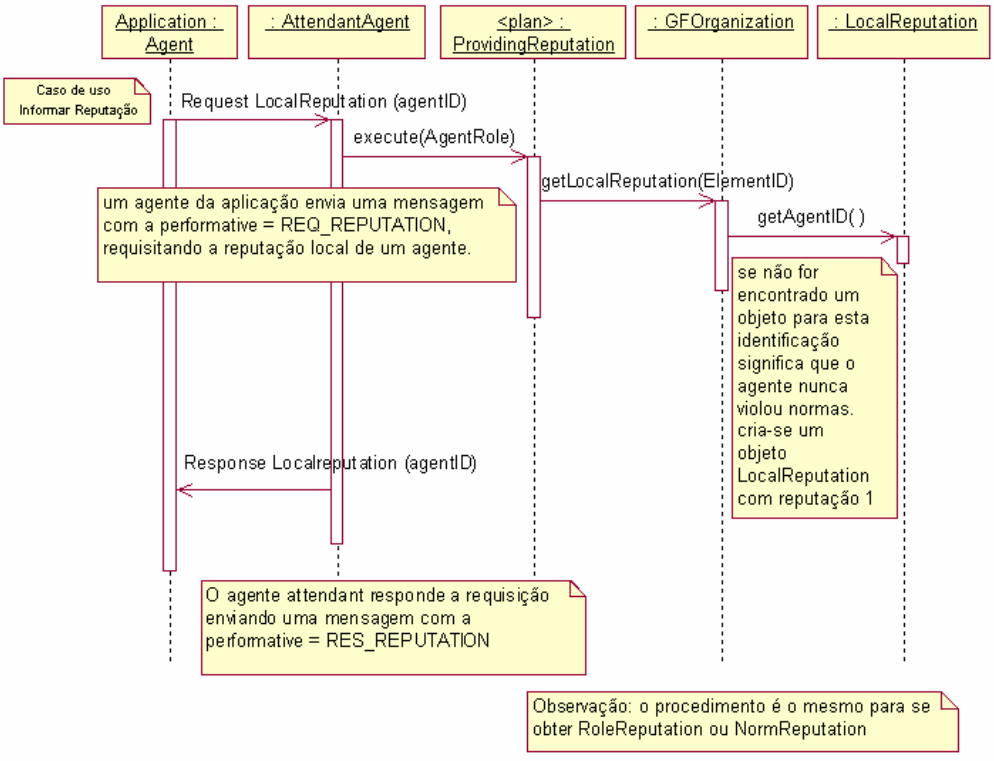

Figura 22. Diagrama de Seqüência – Informar Reputação Local

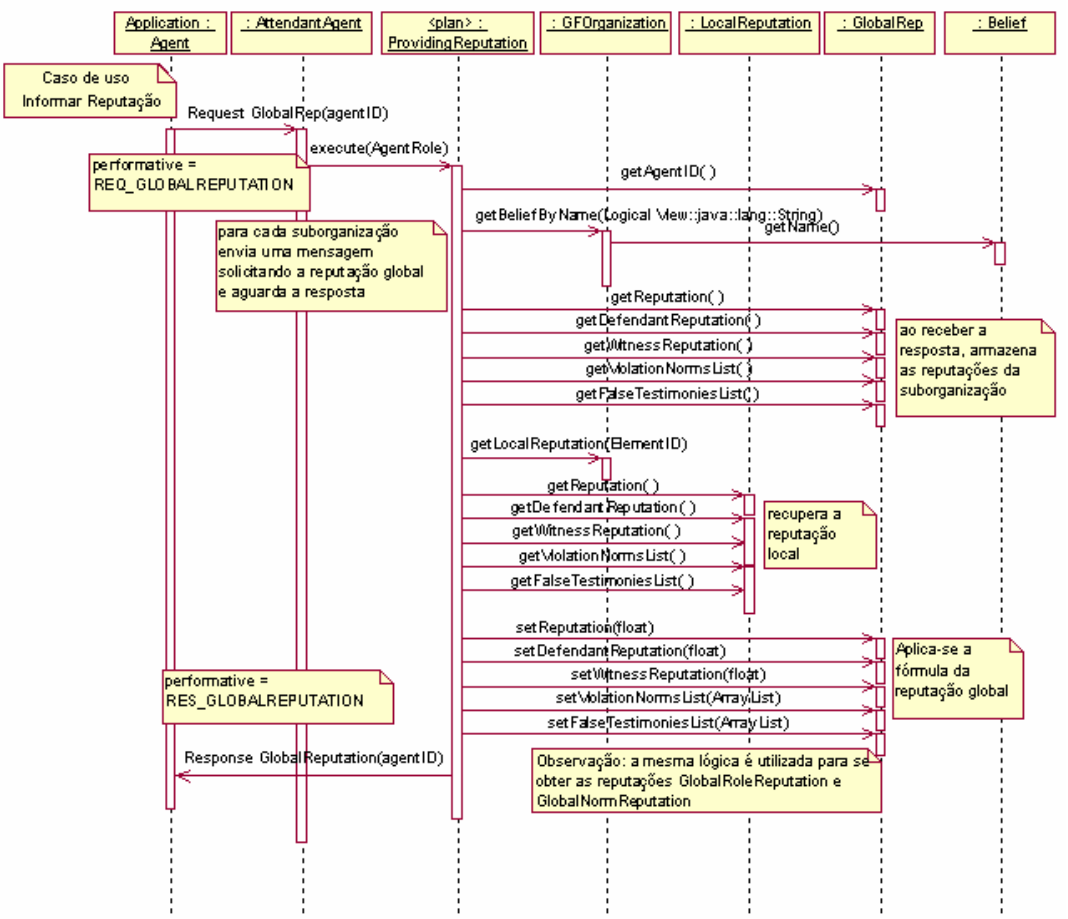

Figura 23. Diagrama de Seqüência – Informar Reputação Global

4.2.5.2. Classificar e Receber Veredicto

Ao receber um veredicto do subsistema de julgamento o agente *Attendant* cria um agente *Evaluator*, repassando o resultado do julgamento como parâmetro. Este agente irá executar duas ações: na primeira ele classifica o testemunho como uma violação de norma ou falso testemunho, conforme o diagrama da figura 24 (caso de uso Classificar veredicto), e na segunda ação altera a reputação do agente condenado conforme o diagrama da figura 25.

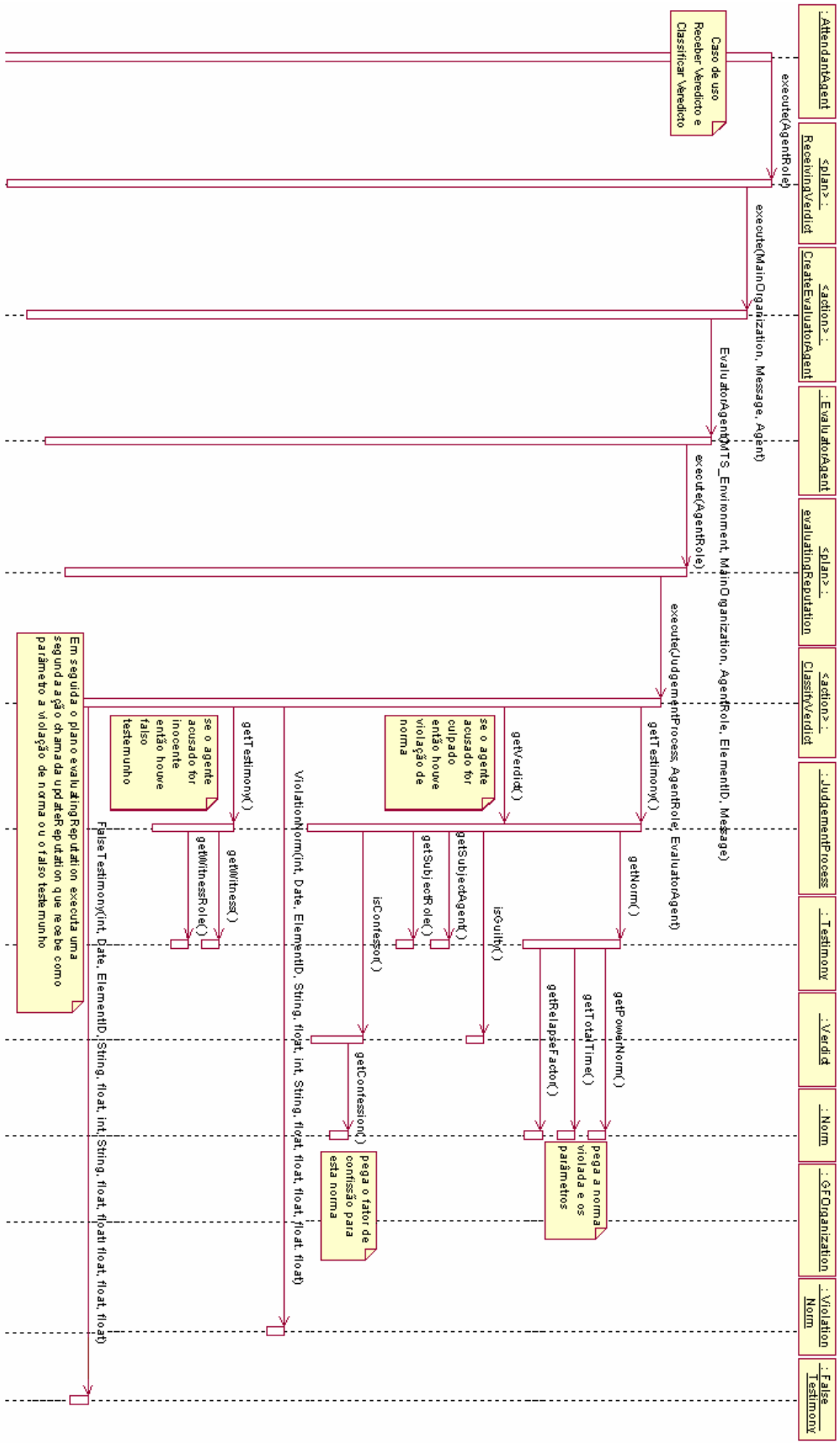

Figura 24. Diagrama de Seqüência – Classificar Veredicto

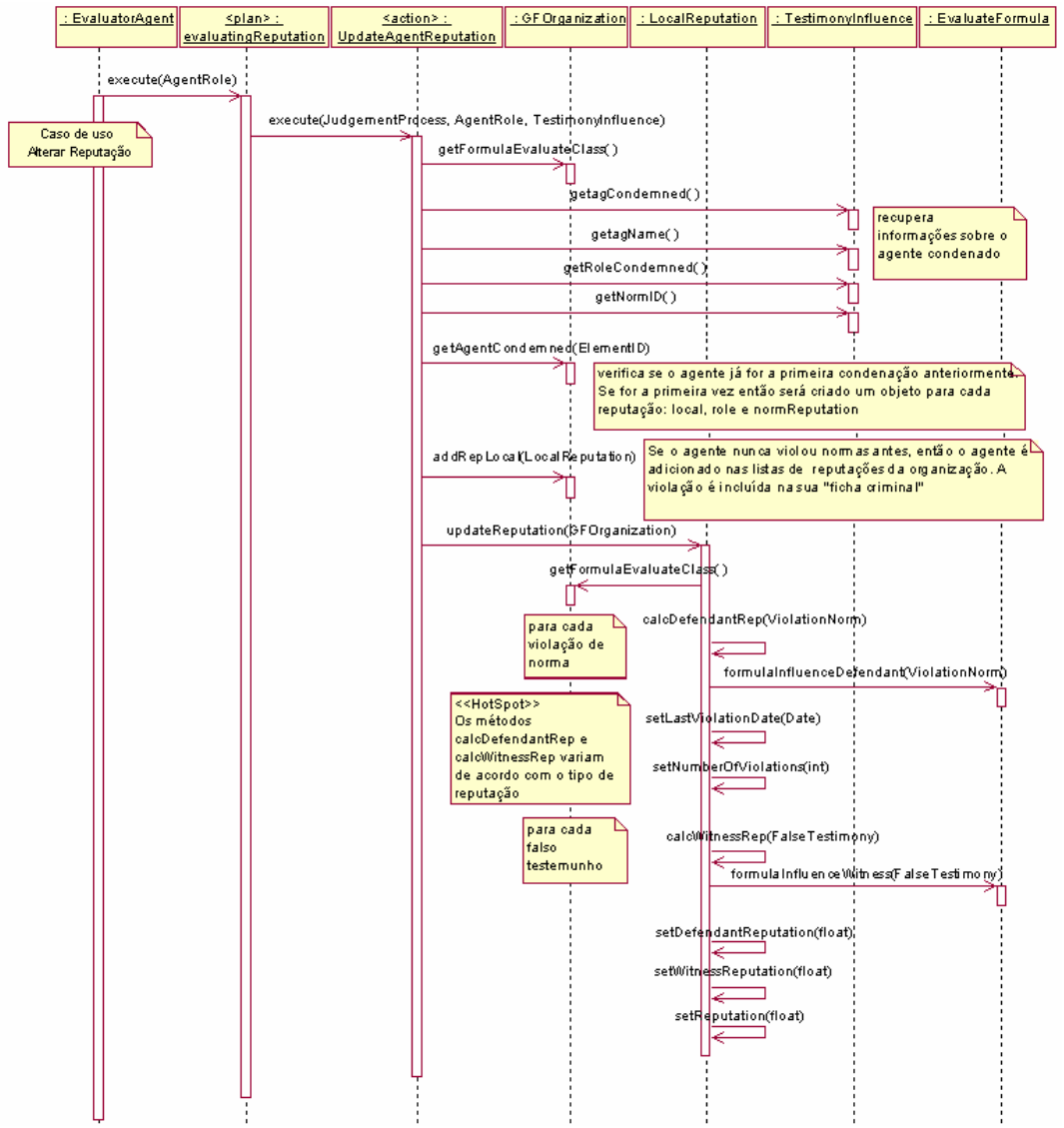

Figura 25. Diagrama de Seqüência - Alterar Reputação

4.2.5.3. Atualizar Reputações

O diagrama da figura 26 apresenta os métodos acionados para atualizar periodicamente todas as reputações dos agentes.

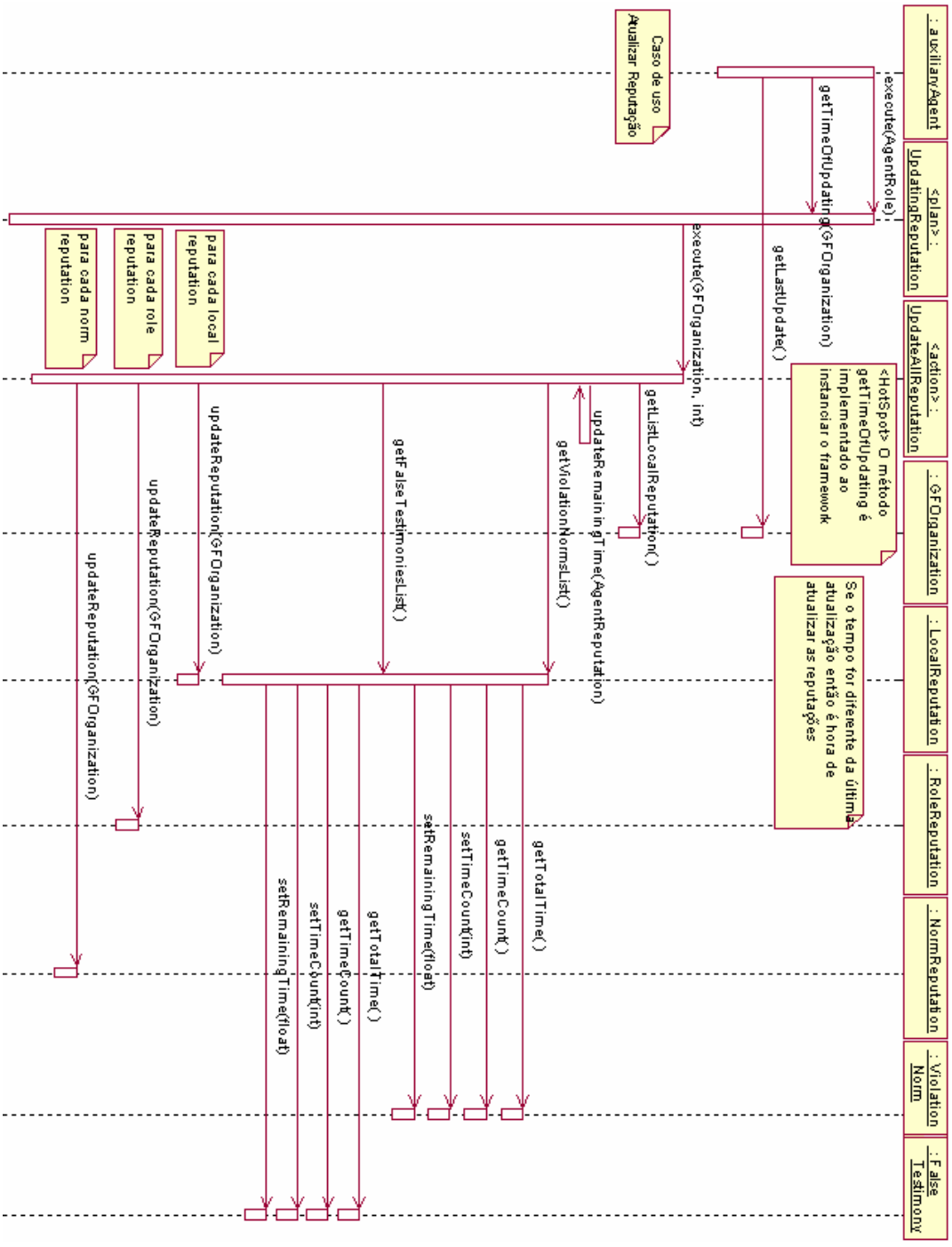

Figura 26. Diagrama de Seqüência – Atualizar reputações

4.2.6. Instanciando o *Framework* REPORT

Para instanciar o *framework* REPORT o desenvolvedor deverá seguir os passos abaixo:

1 Estender a classe *EvaluateFormula* para criar as fórmulas que serão utilizadas pelo *framework* para calcular as reputações dos agentes, se necessário, conforme a seção 4.2.2.1.

2 Estender a classe *UpdatingReputation* para definir o tempo de atualização das reputações, se for diferente do tempo por minutos, conforme a seção 4.2.2.2.

3 Estender a classe *AgentReputation* para criar as reputações personalizadas específicas da aplicação, se houver, implementando os métodos abstratos *isViolationInfluence* e *isFalseTestimonyInfluence*, conforme a seção 4.2.2.3.

4 Instanciar uma organização, seguindo os seguintes passos.

- Criar a organização estendendo a classe *GFOrganization*;
- Atribuir ao campo *formulaEvaluateClass* o nome da classe que implementa as fórmulas de cálculo das reputações (Exemplo: *formulaEvaluateClass = Linear.class*) ;
- Atribuir ao campo *updatingReputationPlan* o nome da classe que implementa a definição do tempo e atualização das reputações (Ex: *updatingReputationPlan = updatingReputationDaily.class*);
- Se houver algum tipo de reputação personalizado, deverá ser criado um objeto *PersonalizedReputation*. Este objeto deverá conter as seguintes informações para cada reputação personalizada: a relação de normas que influenciam este tipo de reputação, o nome da classe que contém as fórmulas de cálculo e o nome da classe que representa este tipo de reputação (nome da classe que especializa a classe *AgentReputation*);
- Especificar as normas que definem as responsabilidades de cada papel que poderá ser desempenhado dentro da organização e seus parâmetros, utilizados no cálculo da reputação: *normPower*,

totalTime, *witnessFactor*, *relapseFactor* e *confession*, definidos na tabela 5. Está tabela descreve todos os atributos de uma norma relevantes para o sistema de reputação. As normas da aplicação são objetos da classe *Norm*. A figura 27 apresenta parte do código de uma organização, com um exemplo de especificação de uma norma;

- Em um sistema multi-agentes podem existir várias organizações. Para definir a hierarquia organizacional, deverão ser informadas quais são as suborganizações de cada organização através da instrução: *Organization.addSubOrganization(SubOrganization)*.

```
/* NORM 2  */
// Consolidator needs to delivery the cargo to the importer(s)
// in the determined location and in the established deadline
norm = new Norm(2);
norm.setSubjectRole(Consolidator.class);
norm.setJudgePlan(Judging.class);
norm.setNormPower((float)0.5);
norm.setTotalTime(30);
norm.setWitnessFactor((float)0.5);
norm.setRelapseFactor((float)0.1);
norm.setConfession((float)0.5);
this.addNorm(norm);
```

Figura 27. Especificação de uma norma dentro de uma organização

Consultando a reputação de um agente:

Os agentes da aplicação podem solicitar a reputação de outro agente enviando uma mensagem para o agente *Attendant*. Cada organização tem pelo menos um agente responsável pelo atendimento. A figura 28 apresenta um código exemplificando a solicitação da reputação. Ao requisitar a reputação de um agente deverá ser anexado a mensagem um objeto da classe correspondente a reputação desejada.

```
// procurando atendente da organização
Collection attendants = AMS.getInstance().searchAgentsbyRole(org,
                    ((GFOrganization)subOrg).getRoleByClass(Attendant.class));
AttendantAgent attendant = (AttendantAgent)attendants.iterator().next();
// criando um objeto LocalReputation
LocalReputation localReputation = new LocalReputation(consolidator.getAgentID());
// enviando a mensagem para o agente attendant da organização
Message outMsg;
outMsg = new Message("LocalReputation", localReputation,
                      agent.getAgentName(), attendant.getAgentName());
outMsg.setPerformative("request:reputation");
outMsg.setReplyTo(agent.getAgentName());
mtp = new MessageTransportProtocol();
mtp.activateClient(1500, outMsg);
```

Figura 28. Exemplo de código para requisição da reputação local de um agente

No exemplo foi criado o objeto *localReputation* com a identificação do agente de quem se deseja saber a reputação. Este objeto foi anexado a mensagem com a performativa *request:reputation*. A relação das performativas definidas para fazer a requisição das reputações pode ser observada na tabela 6.

Variável	Descrição	Intervalo
Blame Percentage	Representa o percentual de culpa de um agente. Definido pelo subsistema de julgamento. Ex: blamePercentage=0,8 indica que o agente foi considerado 80% culpado	[0,1]
Norm Power	Normas diferentes influenciam as reputações de modos diferentes. Ex: *normPower* = 1,0 indica norma muito importante *normPower* = 0,1 indica norma pouco importante	(0,1]
TotalTime	Determina o tempo em que a violação influenciará a reputação de um agente. Valor zero significa que a violação desta norma não será esquecida com o tempo. Ex: *totalTime* = 30 significa que a violação desta norma irá influenciar a reputação do agente durante 30 tempos.	Qtde de *tempos*
Confession	Determina quanto uma confissão de uma violação desta norma diminuirá o seu efeito na reputação. Ex: *confession* = 0,5 diminuirá em 50% a influência da violação desta norma na reputação do agente.	(0,1]
Relapses	A reincidência de violação de uma mesma norma aumenta o poder da norma, aumentando a influência da violação na reputação de um agente. Ex: *relapse* = 1,0 indica norma pouco importante *relapse* = 0,1 indica uma norma muito importante, a reincidência aumentará a influência da violação na reputação porque utilizamos a inversa (1/relapse).	(0,1]
Remaining Time	Determina os dias restantes dentro do tempo total no qual a norma violada influenciará a reputação. Ex: *remainingTime* = 1,0 significa que a violação é recente *remainingTime* = 0,5 significa que já se passaram 50% do tempo total de influência desta violação na reputação do agente. *remainingTime* = 0 significa que a violação não está afetando mais a reputação do agente	[0,1]

Tabela 5. Variáveis utilizadas no cálculo da reputação

Requisitando a reputação	Resposta	Classe Correspondente
request:reputation	response:reputation	LocalReputation.class
request:rolereputation	response:rolereputation	RoleReputation.class
request:normreputation	response:normreputation	NormReputation.class
request:globalreputation	response:globalreputation	GlobalRep.class
request:globalrolereputation	response:globalrolereputation	GlobalRoleRep.class
request:globalnormreputation	response:globalnormreputation	GlobalNormRep.class
request:personalizedreputation	response:personalizedreputation	*Classe personalizada*

Tabela 6. Performativas para requisitar reputações