4
Cenários de Uso

Neste capítulo são descritos dois cenários de uso onde o *framework* JAAF-S foi aplicado. Inicialmente é apresentado um cenário no domínio de escorregamentos de massa. A seguir, um cenário no domínio de agência de viagens.

4.1
Escorregamentos de Massa

Escorregamentos de massa é um fenômeno natural difícil de ser previsto pois ele depende de muitos fatores e da relação entre eles. O número de óbitos e os custos associados à perda de bens materiais e recuperação de áreas degradadas por processos de escorregamentos no Estado do Rio de Janeiro continua a ser expressivos. Como exemplo, na Cidade do Rio de Janeiro, entre 1938 e 2001 foi registrada uma média de vinte e oito escorregamentos por ano, envolvendo 516 óbitos. Custos associados a obras de contenção de encostas ultrapassaram, nos últimos 35 anos, US 300.000.000,00 (Amaral and Feijo 2004).

Diante do apresentado, nota-se que existe uma necessidade de lidar com tais fatores, um dos principais desafios enfrentados por especialistas é encontrar modelos, conhecidos como Modelo de Susceptibilidade (MS), capazes de detectar áreas com maior risco de deslizamento.

Neste contexto, no âmbito do projeto GeoRisc (Lucena and et. al. 2009), nós implementamos uma aplicação baseada em sistemas multiagentes onde, diante de informações sobre novas áreas em análise no município do Rio de Janeiro, agentes implementados utilizando o JAAF-S adaptam seu comportamento em busca de MS que informem o risco de deslizamento da área em questão. Tais modelos são disponibilizados como WS que são descritos através de *Profiles* OWL-S.

4.1.1
Idéia Principal

O sistema multiagente mencionado, recebe dados contendo informações sobre uma área em análise, tais como o tipo de vegetação, solo, quanti-

dade de chuva acumulada, declividade, etc. Quando tais dados são fornecidos, um agente *Manager* é responsável por recebê-los e enviá-los para um grupo de Agentes Geradores de Susceptibilidade (**AGS**) que diante do tipo de dados fornecidos são capazes de encontrar adequados MS. O grupo de **AGS** é dividido em duas categorias que dizem respeito ao tipo de análise aplicada: análise estatística (Soeters and Wsloesten 2007) que utiliza diferentes métodos para comparar a distribuição espacial dos escorregamentos com os fatores (vegetação, declividade, etc.) considerados e análise determinística (Soeters and Wsloesten 2007) onde as principais propriedades de uma área são quantificadas e aplicadas em modelos matemáticos específicos.

Quando um **AGS** recebe dados ele tenta utilizar o MS em uso atualmente para calcular o risco de escorregamento da área em análise. Neste caso, um AGS pode adaptar seu comportamento escolhendo um outro serviço se algum problema ocorre durante a execução, o tempo para geração do risco é alto ou o risco informado pelo MS não é aceitável, a veracidade de um risco pode ser verificada a partir da comparação do mesmo com informações contidas num inventário [1] da área em análise. Se o resultado da comparação retorna um percentual acima de setenta porcento, então o resultado é considerado aceitável.

A fim de detectar os problemas supracitados e realizar adaptações, cada AGS possui: (i) Uma base de regras a partir da qual é possível detectar a existência de problemas utilizando os dados coletados a partir do monitoramento do WS e os dados providos pelo agente *Manager* , (ii) Uma base de casos onde cada um deles (representado pelo par problema-solução - ver Seção 3.3.2) o problema descreve características de áreas que foram analisadas no passado e a solução é um *Profile* OWL-S descrevendo o serviço que disponibiliza o MS utilizado para geração do risco de deslizamento da área descrita no problema, (iii) Um conjunto de *Profiles* OWL-Ss que descrevem serviços atualmente *on-line* e, finalmente, (iv) Uma base compartilhada entre os AGS contendo a reputação de vários serviços.

Cada AGS aplica o *control-loop* de auto-adaptação, provido pelo JAAF-S, como descrito na seção seguinte.

4.1.2 Agentes Geradores de Susceptibilidade

Na atividade *Collect* (Ver Seção 3.3.1) os monitores para interceptação de eventos (Ver método *sensor* da classe *MessageMonitor* na Figura 3.5)

[1] Um inventário armazena informações sobre os últimos escorregamentos ocorridos em um determinada região

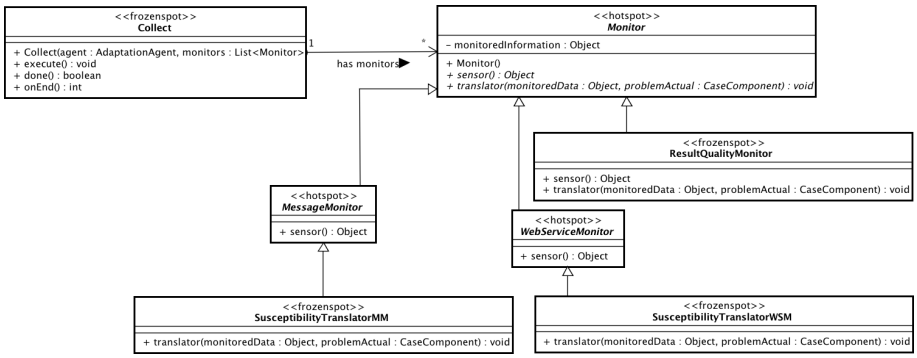

Figura 4.1: Atividade *Collect* de um AGS

e monitoramento de WS (Ver método *sensor* da classe *WebServiceMonitor* na Figura 3.5) ambos disponibilizados pelo JAAF-S foram utilizados. O primeiro irá verificar se existe novas mensagens enviadas pelo agente *Manager* e o segundo irá monitorar o funcionamento do WS obtendo informações sobre a execução do mesmo. Adicionalemente, uma outra classe, chamada *ResultQualityMonitor* (Ver Figura 4.1), que estende a classe *Monitor* (Ver Figura 3.5) e implementa os seus métodos abstratos *sensor* e *translator* foi criada. No método *sensor* foi definida uma implementação que calcula a diferença percentual entre o resultado gerado pelo MS disponibilizado pelo WS e as informações contidas no inventário da área em análise. Já o método *translator* formata o resultado do *sensor* de forma que ele possa ser entendido pelas atividades seguintes.

Por fim, duas classes, *SusceptibilityTranslatorMM* e *SusceptibilityTranslatorWSM* (Ver Figura 4.1), foram criadas como extensões das classes *MessageMonitor* e *WebServiceMonitor*, respectivamente, a fim de implementar o método abstrato *translator* para formatar e estruturar os dados: (i) enviados pelo agente *Manager* a partir do método *translator* da classe *SusceptibilityTranslatorMM*; e (ii) coletados a partir do monitoramento do WS, ou seja, quantidade de falhas ocorridades durante a execução e tempo para geração do risco, a partir do método *translator* da classe *SusceptibilityTranslatorWSM*.

Após a execução da atividade *Collect*, na atividade *Analyze* (Seção 3.3.2) o RBR (Ver Seção 3.3.2) é aplicado utilizando as regras **R1**, **R2** e **R3** (especificadas no método *importBase* da classe *SusceptibilityFowardChaining* apresentada na Figura 4.2) apresentadas abaixo, a fim de detectar problemas sobre a execução do serviço. **R1** detecta se algum problema ocorreu durante a execução do serviço, onde a variável *serviceFailure* representa a quantidade de falhas ocorridas durante a execução do serviço, **R2** verifica se o tempo para geração do risco é alto, onde a variável *responseTime* representa o tempo

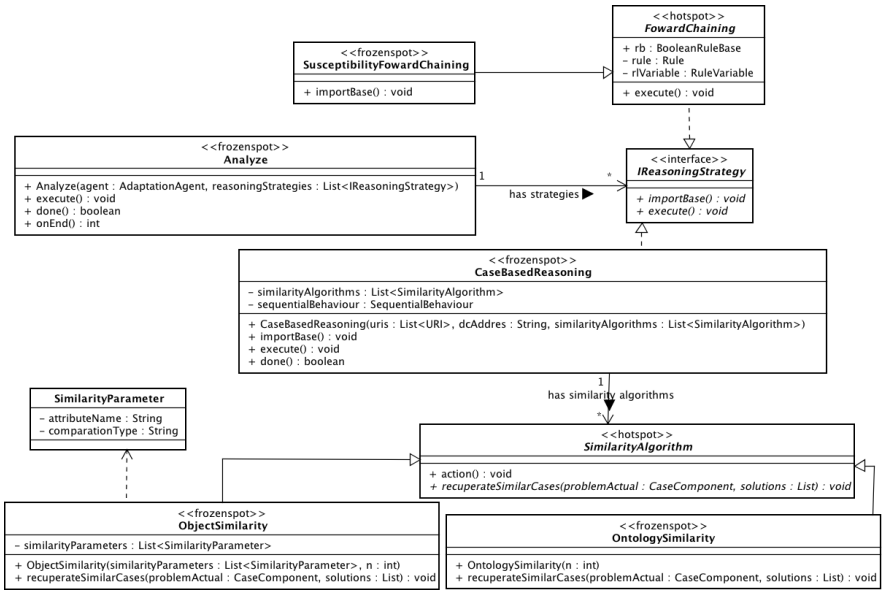

Figura 4.2: Atividade *Analyze* de um AGS

para geração do risco, e **R3** se o risco gerado não é aceitável, onde a variável *resultQuality* representa a diferença percentual entre o resultado gerado pelo MS disponibilizado pelo WS e as informações contidas no inventário da área em análise.

Regra R1 : If *serviceFailure* $>$ 0 Then Problem = "Service Fail"

Regra R2 : If Problem != "Service Fail" and *responseTime* $>$ 20s Then Problem = "High Response Time"

Regra R3 : If Problem != "Service Fail" and *resultaQuality* $<$ 70 Then Problem = "Low Result Quality"

Se nenhum problema foi detectado a partir da aplicação das regras supracitadas, nenhuma adaptação é necessária, o risco gerado pelo MS é enviado para o agente *Manager* que irá atualizar o mapa de susceptibilidade com o risco informado. Caso contrário, a atividade *Analyze* continuará executando.

Neste caso, o algoritmo que verifica similaridade entre objetos (Ver Seção 3.3.2) irá ser aplicado a fim de calcular a similaridade entre as características da área em análise juntamente com os problemas detectados a partir da aplicação do RBR e os casos armazenados na base de caso (Ver Seção 4.1.1). Estes casos são compostos pelo par "problema-solução" onde o problema descreve as características de áreas analisadas no passado e a solução contém *Profiles* OWL-S descrevendo o MS aplicado na área descrita no problema. Para melhor entendimento, considere que na BC tenhamos os seguintes casos:

Caso (1)

Descrição da Área em Análise :

Vegetação : Floresta Degradada

Declividade : 10 porcento

Chuva Acumulada : 25 mm

Problema : "Service Failure"

Solução do Caso (1) – Esta solução contém o *Profile* OWL-S, apresentado na Figura 4.3, que descreve o serviço provedor do MS "Combinação Qualitativa", descrito em (Neto et al. 2009a), com as seguintes características:

Nome Combinação Qualitativa;

Entradas Os objetos dos tipos *Vegetacao* (representa o tipo de vegetação), *Declividade* (representa a declividade da área em análise e o tipo de medida adotada) e *ChuvaAcumulada* (representa a quantidade de chuva acumulada e o tipo de medida);

Saídas O objeto do tipo *Risco* que representa características da área analisada e o risco associado;

Propriedades Tempo de resposta, custo e escalabilidade com qualidades medio, medio e excelente;

Caso (2)

Descrição da Área em Análise :

Chuva Acumulada : 10 mm

Problemas : "Service Failure"

Solução do Caso (2) – Esta solução contém o *Profile* OWL-S, apresentado na Figura 4.4, que descreve o serviço provedor do MS "GeoRio", apresentado em (GeoRio 2009), com as seguintes características:

Nome GeoRio;

Entradas O objeto do tipo *ChuvaAcumulada* (representa a quantidade de chuva acumulada e o tipo de medida);

Saídas O objeto do tipo *Risco* que representa características da área analisada e o risco associado;

Propriedades Tempo de resposta, custo e escalabilidade com qualidades ruim, bom e bom;

```
– <rdf:RDF xml:base="http://139.82.24.82:8080/OWLRepository/CQ.owl">
  – <service:Service rdf:ID="Service">
    – <service:presents>
        <profile:Profile rdf:ID="Profile"/>
      </service:presents>
    – <service:describedBy>
        <process:AtomicProcess rdf:ID="Process"/>
      </service:describedBy>
    </service:Service>
  – <profile:Profile rdf:about="#Profile">
      <profile:serviceName>Combinacao Qualitativa</profile:serviceName>
    – <profile:hasInput>
      – <process:Input rdf:ID="Declividade">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/georisc.owl#Declividade
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="Vegetacao">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/georisc.owl#Vegetacao
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="ChuvaAcumulada">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/georisc.owl#ChuvaAcumulada
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasOutput>
    – <process:Output rdf:ID="Risco">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/georisc.owl#Risco
        </process:parameterType>
      </process:Output>
    </profile:hasOutput>
  – <j.1:serviceParameter>
    – <j.1:ServiceParameter>
        <j.1:serviceParameterName>Custo</j.1:serviceParameterName>
      – <j.1:sParameter>
          <j.0:Qualidade rdf:about="georisc.owl#medio"/>
        </j.1:sParameter>
      </j.1:ServiceParameter>
    </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Tempo de Resposta</j.1:serviceParameterName>
            <j.1:sParameter rdf:resource="georisc.owl#medio"/>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Escalabilidade</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="georisc.owl#excelente"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      </profile:Profile>
   – <process:AtomicProcess rdf:about="#Process">
      <process:hasOutput rdf:resource="#Risco"/>
      <process:hasInput rdf:resource="#ChuvaAcumulada"/>
      <process:hasInput rdf:resource="#Declividade"/>
      <process:hasInput rdf:resource="#Vegetacao"/>
      <service:describes rdf:resource="#Service"/>
    </process:AtomicProcess>
  </rdf:RDF>
```

Figura 4.3: OWL-S : Modelo de Combinação Qualitativa

```
– <rdf:RDF xml:base="http://139.82.24.82:8080/OWLRepository/GeoRio.owl">
  – <service:Service rdf:ID="Service">
    – <service:presents>
        <profile:Profile rdf:ID="Profile"/>
      </service:presents>
    – <service:describedBy>
        <process:AtomicProcess rdf:ID="Process"/>
      </service:describedBy>
    </service:Service>
  – <profile:Profile rdf:about="#Profile">
      <profile:serviceName>GeoRio</profile:serviceName>
    – <profile:hasInput>
      – <process:Input rdf:ID="ChuvaAcumulada">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/georisc.owl#ChuvaAcumulada
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasOutput>
      – <process:Output rdf:ID="Risco">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/georisc.owl#Risco
          </process:parameterType>
        </process:Output>
      </profile:hasOutput>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Custo</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="georisc.owl#bom"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Escalabilidade</j.1:serviceParameterName>
            <j.1:sParameter rdf:resource="georisc.owl#bom"/>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Tempo de Resposta</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="georisc.owl#ruim"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
    </profile:Profile>
  – <process:AtomicProcess rdf:about="#Process">
      <process:hasOutput rdf:resource="#Risco"/>
      <process:hasInput rdf:resource="#ChuvaAcumulada"/>
      <service:describes rdf:resource="#Service"/>
    </process:AtomicProcess>
  </rdf:RDF>
```

Figura 4.4: OWL-S : Modelo GeoRio

Adicionalmente, considere que as informações enviadas pelo agente *Manager* sobre a área em análise juntamente com o problema detectado pelo RBR são como descritas abaixo:

Problema Atual

Características da Área em Análise :

Vegetação : Floresta Degradada

Solo : Solo exposto

Evidência : Cicatrizes

Declividade : 25 porcento

Chuva Acumulada : 25 mm

Problemas : "Low Result Quality"

Então, a similaridade entre o **Problema Atual** e os problemas dos casos **Caso (1)** e **(2)** é calculada como apresentado nas tabelas 4.1 e 4.2, respectivamente. Na tabela 4.1 é apresentada a similaridade entre o **ProblemaAtual** e o problema do **Caso (1)**, a coluna **Similaridade Local** apresenta nas linhas 2, 3 ,4 e 7 o resultado da comparação do tipo *Equal* (Ver Seção 2.4) aplicada nos atributos **Vegetação**, **Solo**, **Evidência** e **Problemas**, nas linhas 5 e 6 o resultado da comparação do tipo *Interval* (Ver Seção 2.4), sendo o intervalo entre 0 e 100, aplicada nos atributos **Declividade** e **Chuva Acumulada** e, por fim, a linha **Similaridade Global** apresenta o valor 0,36 que é o retorno do algoritmo de similaridade global (Ver Seção 2.4) aplicado sobre as similaridades locais. A mesma lógica é aplicada na tabela 4.2 onde a similaridade global entre o **Problema Atual** e o problema do **Caso (2)** é de 0,025.

Tabela 4.1: Similaridade entre o Problema Atual e o Problema do Caso (1)

1	**Atributos**	**Problema Atual**	**Problema do Caso (1)**	**Similaridade Local**
2	**Vegetação**	Floresta Degradada	Floresta Degradada	1
3	**Solo**	Solo exposto	Nao existe	0
4	**Evidência**	Cicatrizes	Nao existe	0
5	**Declividade**	25 porcento	10 porcento	0,15
6	**Chuva Acumulada**	25mm	25mm	1
7	**Problemas**	"Low Result Quality"	"Service Failure"	0
Similaridade Global				0,36

Tabela 4.2: Similaridade entre o Problema Atual e o Problema do Caso (2)

1	**Atributos**	**Problema Atual**	**Problema do Caso (2)**	**Similaridade Local**
2	**Vegetação**	Floresta Degradada	Nao existe	0
3	**Solo**	Solo exposto	Nao existe	0
4	**Evidência**	Cicatrizes	Nao existe	0
5	**Declividade**	25 porcento	Nao existe	0
6	**Chuva Acumulada**	25mm	10mm	0.15
7	**Problemas**	"Low Result Quality"	"Service Failure"	0
Similaridade Global				0.025

Por fim, a solução do caso mais similar (neste caso, Caso (1)) será enviada para o segundo algoritmo de similaridade, o qual verifica a similaridade entre o *Profile* OWL-S especificado pela solução do *Caso 1* e a lista de *Profiles* OWL-S descrevendo serviços atualmente *on-line* passadas como parâmetro ao instanciarmos a classe *CaseBasedReasoning* (Figura 3.6). A fim de encontrar os dois serviços *on-line* mais semelhantes com a solução especificada no Caso (1).

A exemplo de *Profiles* OWL-S descrevendo serviços *on-line* temos os apresentados nas figuras 4.5, 4.6 e 4.4(Figura 4.4 apresenta o *Profile* da solução do **Caso (2)**), onde:

– O *Profile* apresentado na Figura 4.5 descreve o MS Fator de Segurança, apresentado em (Iverson 2000), disponibilizado pelo serviço com as seguintes caraceterísticas:

Nome Fator de Segurança;

Entradas Os objetos dos tipos *Declividade* (representa a declividade da área em análise e o tipo de medida adotada) e *ChuvaAcumulada* (representa a quantidade de chuva acumulada e o tipo de medida);

Saídas O objeto do tipo *Risco* representa características da área analisada e o risco associado;

Propriedades Tempo de resposta, custo e escalabilidade com qualidades medio, excelente e medio;

– O *Profile* apresentado na Figura 4.6 descreve o MS Gusmão Filho, apresentado em (Gusmao and Alheiros 1992), disponibilizado pelo serviço com as seguintes caraceterísticas:

Nome Gusmao Filho;

Entradas Os objetos dos tipos *Vegetacao* (representa o tipo de vegetação), *Declividade* (representa a declividade da área em análise e o tipo de medida adotada) e *ChuvaAcumulada* (representa a quantidade de chuva acumulada e o tipo de medida);

Saídas O objeto do tipo *Risco* representa características da área analisada e o risco associado;

Propriedades Tempo de resposta, custo e escalabilidade com qualidades medio, medio e excelente;

Então, aplicado o algoritmo de similaridade entre ontologias apresentado na Seção 3.3.2, temos na tabela 4.3 a coluna **Similaridade** apresentando a similaridade entre os atributos do *Profile* apresentado na solução do Caso (1) e o da Figura 4.5 tendo como similaridade total 0.5.Na tabela 4.4 a coluna **Similaridade** apresenta a similaridade entre os atributos do *Profile* apresentado na solução do Caso (1) e o da Figura 4.6 tendo como similaridade total 0.875. Na tabela 4.5 a coluna **Similaridade** apresenta a similaridade entre os atributos do *Profile* apresentado na solução do Caso (1) e o da Figura 4.4 tendo como similaridade total 0.25.

Tabela 4.3: Similaridade a solução do Caso (1) e a OWL-S da Figura 4.5

Atributos	**Solução do Caso (1)**	**OWL-S da Figura 4.5**	**Similaridade**
Name	Combinação Qualitativa	Fator de Segurança	0
Parameter	TempoResposta:medio	TempoResposta:medio	1
	Custo:medio	Custo:excelente	0
	Escalabilidade:excelente	Escalabilidade:medio	0
Inputs	Vegetacao	Nao Existe	0
	Declividade	Declividade	1
	ChuvaAcumulada	ChuvaAcumulada	1
Outputs	Risco	Risco	1
Similaridade Total			0.5

Considerando que os três *Profiles* mais semelhantes devem ser enviados para atividade *Decision* (Ver Seção 3.3.3). E que os três mais semelhantes são os apresentados nas Figuras 4.4, 4.5 e 4.6 então os mesmos serão enviados para atividade *Decision* e os mecanismos de função de utilidade (Ver Seção 3.3.3), o qual irá retornar os dois serviços com maior importância, e reputação (Ver Seção 3.3.3), o qual irá retornar o serviço com maior reputação, serão aplicados.

```
– <rdf:RDF xml:base="http://139.82.24.82:8080/OWLRepository/FS.owl">
  – <service:Service rdf:ID="Service">
    – <service:presents>
        <profile:Profile rdf:ID="Profile"/>
      </service:presents>
    – <service:describedBy>
        <process:AtomicProcess rdf:ID="Process"/>
      </service:describedBy>
    </service:Service>
  – <profile:Profile rdf:about="#Profile">
      <profile:serviceName>Fator de Seguranca</profile:serviceName>
    – <profile:hasInput>
      – <process:Input rdf:ID="ChuvaAcumulada">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/georisc.owl#ChuvaAcumulada
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="Declividade">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/georisc.owl#Declividade
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
  – <profile:hasOutput>
    – <process:Output rdf:ID="Risco">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/georisc.owl#Risco
        </process:parameterType>
      </process:Output>
    </profile:hasOutput>
  – <j.1:serviceParameter>
    – <j.1:ServiceParameter>
        <j.1:serviceParameterName>Custo</j.1:serviceParameterName>
      – <j.1:sParameter>
          <j.0:Qualidade rdf:about="georisc.owl#excelente"/>
        </j.1:sParameter>
      </j.1:ServiceParameter>
    </j.1:serviceParameter>
  – <j.1:serviceParameter>
    – <j.1:ServiceParameter>
        <j.1:serviceParameterName>Escalabilidade</j.1:serviceParameterName>
      – <j.1:sParameter>
          <j.0:Qualidade rdf:about="georisc.owl#medio"/>
        </j.1:sParameter>
      </j.1:ServiceParameter>
    </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Tempo de Resposta</j.1:serviceParameterName>
            <j.1:sParameter rdf:resource="georisc.owl#medio"/>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      </profile:Profile>
    – <process:AtomicProcess rdf:about="#Process">
        <process:hasOutput rdf:resource="#Risco"/>
        <process:hasInput rdf:resource="#ChuvaAcumulada"/>
        <process:hasInput rdf:resource="#Declividade"/>
        <service:describes rdf:resource="#Service"/>
      </process:AtomicProcess>
    </rdf:RDF>
```

Figura 4.5: OWL-S: Modelo de Fator de Segurança

```
– <rdf:RDF xml:base="http://139.82.24.82:8080/OWLRepository/CV.owl">
  – <service:Service rdf:ID="Service">
    – <service:presents>
        <profile:Profile rdf:ID="Profile"/>
      </service:presents>
    – <service:describedBy>
        <process:AtomicProcess rdf:ID="Process"/>
      </service:describedBy>
    </service:Service>
  – <profile:Profile rdf:about="#Profile">
      <profile:serviceName>Gusmao Filho</profile:serviceName>
    – <profile:hasInput>
      – <process:Input rdf:ID="Declividade">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/georisc.owl#Declividade
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="Vegetacao">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/georisc.owl#Vegetacao
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="ChuvaAcumulada">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/georisc.owl#ChuvaAcumulada
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasOutput>
    – <process:Output rdf:ID="Risco">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/georisc.owl#Risco
        </process:parameterType>
      </process:Output>
    </profile:hasOutput>
  – <j.1:serviceParameter>
    – <j.1:ServiceParameter>
        <j.1:serviceParameterName>Tempo de Resposta</j.1:serviceParameterName>
      – <j.1:sParameter>
          <j.0:Qualidade rdf:about="georisc.owl#medio"/>
        </j.1:sParameter>
      </j.1:ServiceParameter>
    </j.1:serviceParameter>
        – <j.1:serviceParameter>
          – <j.1:ServiceParameter>
              <j.1:serviceParameterName>Custo</j.1:serviceParameterName>
              <j.1:sParameter rdf:resource="georisc.owl#medio"/>
            </j.1:ServiceParameter>
          </j.1:serviceParameter>
        – <j.1:serviceParameter>
          – <j.1:ServiceParameter>
              <j.1:serviceParameterName>Escalabilidade</j.1:serviceParameterName>
            – <j.1:sParameter>
                <j.0:Qualidade rdf:about="georisc.owl#excelente"/>
              </j.1:sParameter>
            </j.1:ServiceParameter>
          </j.1:serviceParameter>
        </profile:Profile>
      – <process:AtomicProcess rdf:about="#Process">
          <process:hasOutput rdf:resource="#Risco"/>
          <process:hasInput rdf:resource="#ChuvaAcumulada"/>
          <process:hasInput rdf:resource="#Declividade"/>
          <process:hasInput rdf:resource="#Vegetacao"/>
          <service:describes rdf:resource="#Service"/>
        </process:AtomicProcess>
      </rdf:RDF>
```

Figura 4.6: OWL-S: Modelo de Gusmão Filho

Tabela 4.4: Similaridade a solução do Caso (1) e a OWL-S da Figura 4.6

Atributos	**Solução do Caso (1)**	**OWL-S da Figura 4.6**	**Similaridade**
Name	Combinação Qualitativa	Gusmao Filho	0
Parameter	TempoResposta:medio	TempoResposta:medio	1
	Custo:medio	Custo:medio	1
	Escalabilidade:excelente	Escalabilidade:excelente	1
Inputs	Vegetação	Vegetacao	1
	Declividade	Declividade	1
	ChuvaAcumulada	ChuvaAcumulada	1
Outputs	Risco	Risco	1
Similaridade Total			0.875

Tabela 4.5: Similaridade a solução do Caso (1) e a OWL-S da Figura 4.4

Atributos	**Solução do Caso (1)**	**OWL-S da Figura 4.4**	**Similaridade**
Name	Combinação Qualitativa	GeoRio	0
Parameter	TempoResposta:media	TempoResposta:ruim	0
	Custo:medio	Custo:bom	0
	Escalabilidade:excelente	Escalabilidade:bom	0
Inputs	Vegetação	Nao Existe	0
	Declividade	Nao Existe	0
	ChuvaAcumulada	ChuvaAcumualda	1
Outputs	Risco	Risco	1
Similaridade Total			0.25

Função de Utilidade: Considerando que as dimensões de qualidade tempo de resposta (t), custo (c) e escalabilidade (e) são classificadas como *ruim*, *medio*, *bom* e *excelente*, com utilidades 0, 0.5, 0.75 e 1, respectivamente. Então, as utilidades referentes aos MS Gusmao Filho, Fator de Segurança e GeoRio são como descritas abaixo:

Gusmao Filho : Possui tempo de resposta *medio*, custo *medio* e escalabilidade *excelente* , com grau de utilidades $u_t(medio) = 0.5$, $u_c(medio) = 0.5e$ e $u_e(excelente) = 1$, respectivamente;

Fator de Segurança : Possui tempo de resposta *medio*, custo *excelente* e escalabilidade *medio* , com grau de utilidades $u_t(medio) = 0.5$, $u_c(excelente) = 1e$ e $u_e(medio) = 0.5$, respectivamente;

GeoRio : Possui tempo de resposta *ruim*, custo *bom* e escalabilidade *bom* , com grau de utilidades $u_t(ruim) = 0$, $u_c(bom) = 0.75$ e $u_e(bom) = 0.75$, respectivamente;

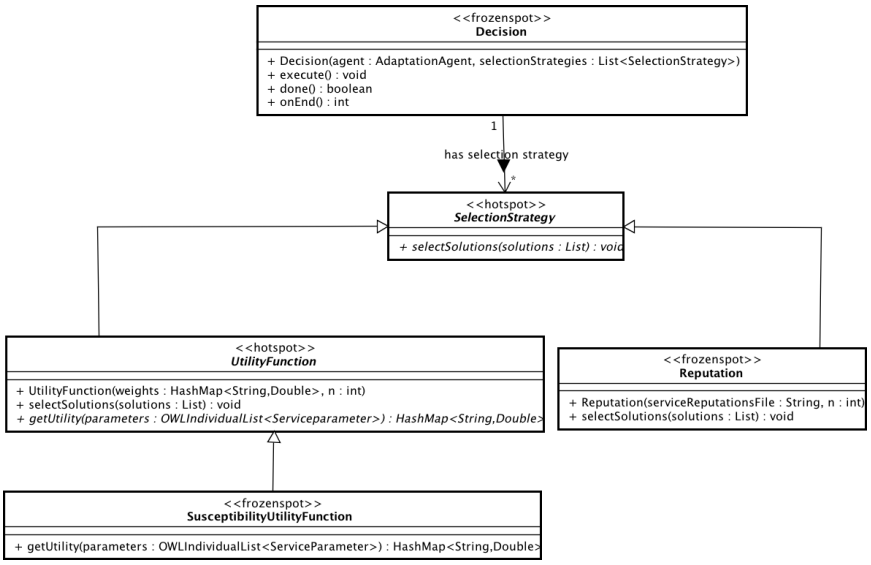

Figura 4.7: Atividade *Decision* de um AGS

Desta forma, tomando que as dimensões tempo de resposta, custo e escabilidade, tem pesos $w_t = 0.4$, $w_c = 0.3$, $w_e = 0.3$, podemos calcular a importância de cada um dos MS seguindo a equação apresentada na Seção 3.3.3, como descrito abaixo:

Gusmao Filho : 0.4*0.5 + 0.3*0.5 + 0.3*1 = 0.53;

Fator de Segurança : 0.4*0.5 + 0.3*1 + 0.3*0.5 = 0.53;

GeoRio : 0.4*0 + 0.3*0.75 + 0.3*0.75 = 0.45;

Então, considerando que os dois serviços mais semelhantes devem ser enviados para o mecanismos de reputação, como mencionado anteriormente. Os *Profiles* OWL-S descrevendo os serviços dos MS Gusmao Filho e Fator de Segurança serão enviados para o mecanismo de reputação.

Para implementar o mecanismos de função de utilidade a classe *SusceptibilityUtilityFunciton* (Ver Figura 4.7) foi criada, como extensão da classe abstrata *UtilityFunction*, e o método *getUtility* foi implementado a fim de criar uma *HashMap* que tem como *Key* o nome das dimensões de qualidade supracitadas (tempo de resposta, custo e escalabilidade) e *Value* os seus respectivos graus de utilidade. Adicionalmente, ao instanciarmos a classe *SusceptibilityUtilityFunciton* dois parâmetros são fornecidos: (i) uma *HashMap* onde a *Key* é uma das dismensões de qualidade supracitadas e *Value* o peso dela, e (ii) o número de serviços a serem enviados para o próximo mecanismos foi definido como dois.

```
– <Services>
  – <Service>
      <URI>http://139.82.24.82:8080/OWLRepository/CV.owl</URI>
      <Reputation>0.9</Reputation>
    </Service>
  – <Service>
      <URI>http://139.82.24.82:8080/OWLRepository/FS.owl</URI>
      <Reputation>0.5</Reputation>
    </Service>
  </Services>
```

Figura 4.8: Reputação

Reputação: O mecanismos de reputação irá verificar o nome de cada serviço e verificar a reputação do mesmo, retornando o serviço de maior reputação. Então, considerando os serviços enviados pelo mecanismo anteriormente aplicado e o arquivo XML apresentado na figura 4.8, onde verificamos que a reputação dos MS Gusmao Filho e Fator de Segurança é 0.9 e 0.5, respectivamente. Como o MS Gusmao Filho possui uma maior reputação, ele será enviado para a atividade *Effector* (Ver Seção 3.3.4).

Para implementar o mecanismos de reputação a classe *Reputation* (Ver Figura 3.10), disponibilizada pelo JAAF-S, foi instanciada passando como parâmetro o endereço do arquivo XML supracitado e definindo o número de serviços a serem enviados para atividade *Effector* como um.

Na última atividade, chamada *Effector*, o AGS envia uma mensagem para o agente *Manager* informando sobre o novo serviço, disponibilizá-o, configura o mecanismos de monitoramento para que o mesmo esteja apto à monitorar o serviço selecionado e executa o compotamento *WaitRequest* (Ver Seção 3.3) com os dados enviados pelo agente *Manager*. Ou seja, os atuadores representados pelas classes *Notification* e *ServiceExecution* foram utilizados (Ver Seção 3.3.4).

4.2 Agência de Viagens

Uma grande dificuldade para usuários que desejam viajar é procurar diferentes serviços que satisfaçam suas necessidades relacionadas a viagens, por exemplo se uma pessoa resolve tirar suas férias viajando para Europa. Para conseguir realizar tal tarefa ele terá de seguir um conjunto de passos como: fazer uma busca por serviços que efetuem reservas de hoteis que tenham o nível de acomodação desejado (5, 4, 3, 2 ou 1 estrela), procurar por serviços de companhias aéreas para a compra das passagens e, por fim, procurar serviços que realizem reserva de carros para disfrutar do destino como um nativo.

Considerando que estas tarefas possam ser satisfeitas a partir da invo-

cação de serviços que estão anotados semanticamente, seria possível o usuário configurar um agente de software capaz de, sem intervenção humana, buscar e selecionar WS que atendam aos requisitos do usuário, e invocá-los de maneira automática.

Neste contexto, nós implementamos uma aplicação baseada em sistemas multiagentes que tem como objetivo fazer reserva de pacotes de viagens (PV) a partir de um conjunto de informações fornecidas pelo usuário.

Tais PV são disponibilizados através de WS (descritos através de OWL-S) e os agentes implementados utilizando JAAF-S são capazes de adaptar seu comportamento em busca de serviços que melhor satisfaçam as necessidades dos usuários.

4.2.1 Idéia Principal

O sistema multiagente mencionado, recebe informações fornecidas pelo usuário, tais como: nome, cpf, idade, profissão, *hobbie* principal, informações sobre o cartão de crédito (incluindo número do cartão de crédito e data de validade), companhia aérea que deseja viajar, cidade de origem, cidade de destino, mês que pretende viajar, dia da partida, dia do retorno, tipo de acomodação (5, 4, 3, 2 ou 1 estrela), tipo de carro para locação e evento que deseja participar. Dependendo de quais dados foram fornecidos pelo usuário, o sistema retorna um pacote com informações sobre as reservas do hotel, passagens aéreas, ingressos do evento e do carro reservado para locação a fim de passear na cidade destino. Por exemplo, se usuário fornece informações como companhia aérea, cidade de origem, cidade de destino, mês da viagem, dia da partida e retorno, cartão de crédito e evento, então, o sistema retornará informações sobre as passagens aéreas e ingressos no evento.

Adicionalmente o sistema apresenta para o usuário uma descrição do serviço em uso atualmente. Desta forma, o usuário pode optar por não utilizar o serviço e escolher que o sistema multiagente procure outro serviço.

O sistema é composto por um agente *Manager* e um grupo de Agentes de Pacotes de Viagens (APV) que conhecem serviços de partes do mundo: América do Sul, América do Norte e Europa. Desta forma, quando tais dados são fornecidos, o agente *Manager* é responsável por recebê-los , a partir da cidade de destino especificada pelo usuário verificar qual APV abrange tal cidade, e enviar os dados recebidos para o APV selecionado, o qual é capaz de encontrar o serviço que possa satisfazer as necessidades do usuário.

Cada APV, está apto a executar o serviço em uso atualmente com os dados enviados pelo agente *Manager*. Se ocorreu alguma falha durante a

execução do serviço, não foi possível realizar autenticação do usuário ou o usuário optou por não utilizar o serviço atualmente em uso, o agente pode adaptar seu comportamento escolhendo um outro serviço.

A fim de detectar os problemas supracitados e realizar adaptações cada APV tem disponível: (i) uma base de regras que utiliza os dados coletados a partir do monitoramento do WS em uso e dos dados providos pelo agente *Manager* para detectar a existência de problemas, (ii) uma base de casos onde cada um deles (representado pelo par problema-solução) o problema descreve necessidades de usuários que precisaram de serviços relacionados a viagens no passado e a solução contém um *Profile* OWL-S que descreve o serviço utilizado para satisfazer as necessidades do usuário, (iii) um conjunto de OWL-Ss que descrevem serviços *on-line* atualmente e, finalmente, (iv) Uma base compartilhada entre agentes contendo a reputação de vários serviços.

Em suma, tais bases são utilizadas pelos APV, seguindo o *control-loop* de auto-adaptação provido pelo JAAF-S, como será descrito na seção seguinte.

4.2.2 Agentes de Pacotes de Viagens

Na atividade *Collect* (Ver Seção 3.3.1) os monitores para interceptação de eventos (Ver método *sensor* da classe *MessageMonitor* na Figura 3.5) e monitoramento de WS (Ver método *sensor* da classe *WebServiceMonitor* na Figura 3.5) ambos disponibilizados pelo JAAF-S foram utilizados. O primeiro irá verificar se existe novas mensagens enviadas pelo agente *Manager* e o segundo irá monitorar o funcionamento do WS obtendo informações sobre a execução do mesmo.

Por fim, duas classes *TravelTranslatorMM* e *TravelTranslatorWSM* (Ver Figura 4.9) foram criadas como extensões das classes *MessageMonitor* e *WebServiceMonitor* (Ver Figura 3.5), respectivamente, a fim de implementar os seus respectivos métodos abstratos *translator* para formatar e estruturar os dados coletados: (i) através da interceptação da mensagem enviada pelo agente *Manager*, a partir do monitor *MessageMonitor*, e (ii) a partir do resultado do monitoramento do WS, utilizando o monitor *WebServiceMonitor*.

Após a execução da atividade *Collect*, a atividade *Analyze* (Seção 3.3.2) é executada. O RBR (Ver Seção 3.3.2) é aplicado utilizando as regras **R1**, **R2** e **R3** (especificadas no método *importBase* da classe *TravelFowardChaining* (Ver Figura 4.10), que é extensão da classe *FowardChaining* apresentada na Figura 3.6) apresentadas abaixo a fim de detectar problemas durante a execução do serviço. **R1** detecta se algum problema ocorreu durante a execução do serviço, onde a variável *serviceFailure* representa o número de falhas ocorridas durante

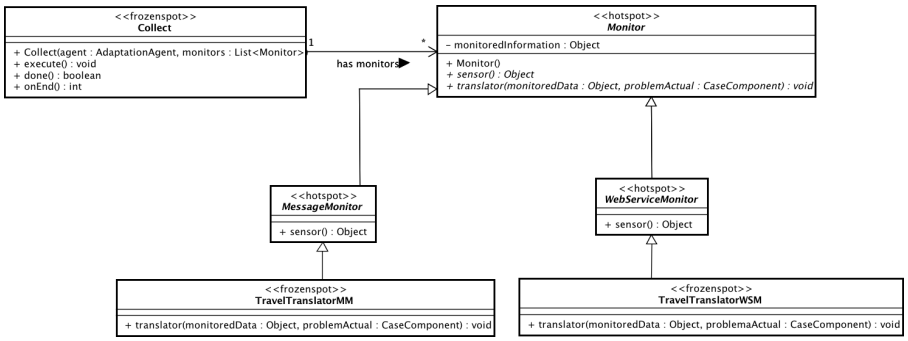

Figura 4.9: Atividade *Collect* de um APV

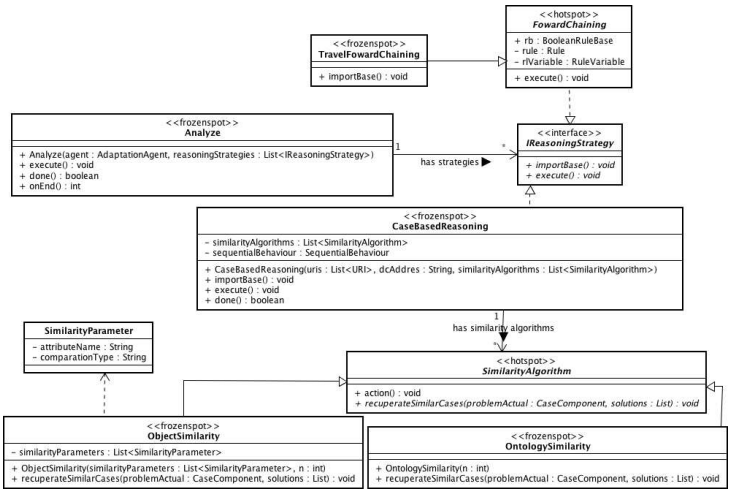

Figura 4.10: Atividade *Analyze* de um APV

a execução do serviço, **R2** detecta se ocorreu algum erro de autenticação, onde a variável *authenticationError* representa se ocorreu algum erro de autenticação e **R3** se o usuário optou por não utilizar o serviço selecionado, neste caso a variável *selectService* armazena opção escolhida pelo usuário.

Regra R1 : If *serviceFailure* > 0 Then Problem = "Service Fail"

Regra R2 : If Problem != "Service Fail" and *authenticationError* = true Then Problem = "Authentication Erro"

Regra R3 : If *selectService* == true Then Problem = "Selecting Other Service"

Se nenhum problema foi detectado a partir da aplicação das regras supracitadas, nenhuma adaptação é necessária, as informações sobre as reservas do pacote de viagem serão enviadas para o agente *Manager* que irá apresentar para o usuário. Caso contrário, a atividade *Analyze* continuará executando.

O algoritmo, disponibilizado pelo JAAF-S, que verifica similaridade entre objetos (Ver Seção 3.3.2) será aplicado visando calcular a similaridade entre as informações enviadas pelo agente *Manager* juntamente com os problemas detectados a partir da aplicação do RBR, e os casos armazenados na base de caso (Ver Seção 4.2.1). Sendo cada caso, como mencionado na seção 4.2.1, descrito pelo par problema-solução onde o problema descreve necessidades de usuários que precisaram de serviços relacionados à viagens no passado e a solução contém um *Profile* OWL-S que descreve o serviço utilizado para satisfazer as necessidades do usuário.

Por exemplo, considere que na BC tenhamos os seguintes casos:

Caso (1) Problema :

- Idade: 30
- Profissão: Jogador
- *Hobbie*: Futebol
- Cidade Origem: Maceió
- Cidade Destino: Rio de Janeiro
- Companhia Aérea: TAM
- Mês: Fevereiro
- Dia Partida: 10
- Dia Retorno: 15
- Transporte: Ferrari
- Evento: Rock in Rio
- Problemas: "Service Failure"

Solução :

- Esta solução contém o *Profile* OWL-S, apresentado na Figura 4.11, que descreve um serviço com as seguintes características:

  Nome Pacote Viagem Eventos;

  Entradas Os objetos dos tipos *Trecho* (representa as cidade de origem e destino), *Carro* (representa o tipo de carro), *Pessoa* (representando as informações do usuário como nome, cpf e *hobbie*), *Cartao-Credito* (representando informações sobre o cartão de crédito do usuário), *Evento* (representando informações sobre o evento que o usuário deseja participar) e *Periodo* (representando o mes, dia de chegada e retorno da viagem);

  Saídas O objeto do tipo *Confirmacao* que representa as informações sobre as passagens aéreas, hotel, locação de carro e ingressos de eventos;

  Propriedades Tempo de resposta, segurança e escalabilidade com qualidades *bom*, *excelente* e *excelente*;

Caso (2) Problema :

- Idade: 28
- Profissão: Político
- *Hobbie*: Leitura
- Cidade Origem: Maceió
- Cidade Destino: Madri
- Companhia Aérea: TAM
- Acomodação: 5 estrelas
- Mês: Fevereiro
- Dia Partida: 10
- Dia Retorno: 15
- Problemas: "Service Failure"

Solução :

- Esta solução contém o *Profile* OWL-S, apresentado na Figura 4.12, que descreve um serviço com as seguintes características:

  Nome Pacote Viagem Resort;
  Entradas Os objetos dos tipos
  Trecho (representa as cidade de origem e destino), *Pessoa* (representando as informações do usuário como nome, cpf e hobbie), *CartaoCredito* (representando informações sobre o cartão de crédito do usuário), *Periodo* (representando o mes, dia de chegada e retorno da viagem), *TipoAcomodacao* (representado o tipo de acomodação podendo ser 1, 2, 3, 4 ou 5 estrelas);
  Saídas O objeto do tipo *Confirmacao* que representa as informações sobre as passagens aéreas, hotel, locação de carro e ingressos de eventos;
  Propriedades Tempo de resposta, segurança e escalabilidade com qualidades *medio*, *medio* e *excelente*;

Adicionalmente, considere que as informações enviadas pelo agente *Manager* juntamente com problemas detectados a partir do RBR, são como descritas abaixo:

Problema Atual Problema
- Idade: 25
- Profissão: Jogador
- *Hobbie*: Futebol
- Cidade Origem: Maceió
- Cidade Destino: Lisboa
- Companhia Aérea: TAM
- Mês: Fevereiro
- Dia Partida: 10
- Dia Retorno: 15

```
– <rdf:RDF xml:base="http://139.82.24.82:8080/OWLRepository/PVE.owl">
  – <service:Service rdf:ID="Service">
    – <service:presents>
        <profile:Profile rdf:ID="Profile"/>
      </service:presents>
    – <service:describedBy>
        <process:AtomicProcess rdf:ID="Process"/>
      </service:describedBy>
    </service:Service>
  – <profile:Profile rdf:about="#Profile">
      <profile:serviceName>Pacote Viagem Evento</profile:serviceName>
    – <profile:hasInput>
      – <process:Input rdf:ID="Carro">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#Carro
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="Evento">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#Evento
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="CartaoCredito">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#CartaoCredito
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="Periodo">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Periodo
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="Pessoa">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Pessoa
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="Trecho">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Trecho
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasOutput>
    – <process:Output rdf:ID="Confirmacao">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Confirmacao
        </process:parameterType>
      </process:Output>
    </profile:hasOutput>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Tempo de Resposta</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="Viagem.owl#bom"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Escalabilidade</j.1:serviceParameterName>
            <j.1:sParameter rdf:resource="Viagem.owl#excelente"/>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Seguranca</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="Viagem.owl#excelente"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
    </profile:Profile>
  – <process:AtomicProcess rdf:about="#Process">
      <process:hasOutput rdf:resource="#Confirmacao"/>
      <process:hasInput rdf:resource="#Periodo"/>
      <process:hasInput rdf:resource="#CartaoCredito"/>
      <process:hasInput rdf:resource="#Pessoa"/>
      <process:hasInput rdf:resource="#Carro"/>
      <process:hasInput rdf:resource="#Evento"/>
      <process:hasInput rdf:resource="#Trecho"/>
      <service:describes rdf:resource="#Service"/>
    </process:AtomicProcess>
  </rdf:RDF>
```

Figura 4.11: OWL-S do Serviço provedor do Pacote Viagem Evento

```
– <rdf:RDF xml:base="http://139.82.24.82:8080/OWLRepository/PVR.owl">
  – <service:Service rdf:ID="Service">
    – <service:presents>
        <profile:Profile rdf:ID="Profile"/>
      </service:presents>
    – <service:describedBy>
        <process:AtomicProcess rdf:ID="Process"/>
      </service:describedBy>
    </service:Service>
  – <profile:Profile rdf:about="#Profile">
      <profile:serviceName>Pacote Viagem Resort</profile:serviceName>
    – <profile:hasInput>
      – <process:Input rdf:ID="Trecho">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#Trecho
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="TipoAcomodacao">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#TipoAcomodacao
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="Pessoa">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#Pessoa
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="Periodo">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Periodo
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="CartaoCredito">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#CartaoCredito
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasOutput>
    – <process:Output rdf:ID="Confirmacao">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Confirmacao
        </process:parameterType>
      </process:Output>
    </profile:hasOutput>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Escalabilidade</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="Viagem.owl#excelente"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Tempo de Resposta</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="Viagem.owl#medio"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Seguranca</j.1:serviceParameterName>
            <j.1:sParameter rdf:resource="Viagem.owl#medio"/>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      </profile:Profile>
    – <process:AtomicProcess rdf:about="#Process">
        <process:hasOutput rdf:resource="#Confirmacao"/>
        <process:hasInput rdf:resource="#CartaoCredito"/>
        <process:hasInput rdf:resource="#Pessoa"/>
        <process:hasInput rdf:resource="#Periodo"/>
        <process:hasInput rdf:resource="#TipoAcomodacao"/>
        <process:hasInput rdf:resource="#Trecho"/>
        <service:describes rdf:resource="#Service"/>
      </process:AtomicProcess>
    </rdf:RDF>
```

Figura 4.12: OWL-S do Serviço Pacote Viagem Resort

- Transporte: Ferrari
- Evento: Rock in Rio
- Problemas: "Service Failure"

Tabela 4.6: Similaridade entre o Problema Atual e o Problema do Caso (1)

1	**Atributos**	**Problema Atual**	**Problema do Caso (1)**	**Similaridade Local**
2	**Idade**	25	30	0.95
3	**Profissão**	Jogador	Jogador	1
4	**Hobbie**	Futebol	Festas	0
5	**Cidade Origem**	Maceió	Maceió	1
6	**Cidade Destino**	Lisboa	Rio de Janeiro	0
7	**Transporte**	Ferrari	Ferrari	1
8	**Companhia Aérea**	TAM	TAM	1
9	**Mes**	Fevereiro	Fevereiro	1
10	**Dia Partida**	10	10	1
11	**Dia Retorno**	15	15	1
12	**Evento**	Rock in Rio	Rock in Rio	1
13	**Problem**	"Service Failure"	"Service Failure"	1
Similaridade Global				0,83

Então, a similaridade entre o **Problema Atual** e os problemas dos casos **Caso (1)** e **(2)** é calculada como apresentado nas tabelas 4.6 e 4.7, respectivamente. Na tabela 4.6 é apresentada a similaridade entre o **ProblemaAtual** e o problema do **Caso (1)**, a coluna **Similaridade Local** apresenta nas linhas 3 ,4, 5, 6, 7, 8, 12 e 13 o resultado da comparação do tipo *Equal* (Ver Seção 2.4) aplicada nos atributos **Profissão**, **Hobbie**, **Cidade Origem**, **Cidade Destino**, **Transporte**, **Companhia Aérea**, **Evento** e **Problema**, nas linhas 2, 10 e 11 o resultado da comparação do tipo *Interval* (Ver Seção 2.4) aplicada nos atributos **Idade** (sendo intervalo entre 0 e 100), **Dia Partida** e **Dia Retorno** (sendo intervalo entre 1 e 31), e na linha 9 o resultado da comparação do tipo *EnumDistance* (Ver Seção 2.4) sobre o atributo **Mês** (Janeiro, Fevereiro, Março, Abril, Maio, Junho, Julho, Agosto, Setembro, Outubro, Novembro e Dezembro). Por fim, a linha **Similaridade Global** apresenta o valor 0,83 que é o retorno do algoritmo de similaridade global (Ver Seção 2.4) aplicado sobre as similaridades locais. A mesma lógica é aplicada na tabela 4.6 onde a similaridade global entre o **Problema Atual** e o problema do **Caso (2)** é de 0,41.

Tabela 4.7: Similaridade entre o Problema Atual e o Problema do Caso (2)

1	**Atributos**	**Problema Atual**	**Problema do Caso (2)**	**Similaridade Local**
2	**Idade**	25	28	0.97
3	**Profissão**	Jogador	Político	0
4	**Hobbie**	Futebol	Leitura	0
5	**Cidade Madri**	Maceió	Maceió	0
6	**Cidade Destino**	Lisboa	Madri	0
7	**Transporte**	Ferrari	Não existe	0
8	**Companhia Aérea**	TAM	TAM	1
9	**Mes**	Fevereiro	Fevereiro	1
10	**Dia Partida**	10	10	1
11	**Dia Retorno**	15	15	1
12	**Evento**	Rock in Rio	Não existe	0
13	**Problem**	"Service Failure"	"Service Failure"	1
Similaridade Global				0,41

Como o **Caso (1)** é o mais semelhantes ao **Problema Atual**, então o *Profile* especificado na solução do **Caso (1)** será enviada para o segundo algoritmo que verifica a similaridade entre ontologias (Ver Seção 3.3.2). Especificamente, tal algoritmo irá verificar a similaridade entre o *Profile* OWL-S descrito pela solução do *Caso 1* e a lista de ontologias OWL-S que descrevem serviços atualmente *on-line*, passados como parâmetro ao instanciarmos a classe *CaseBasedReasoning* (Figura 3.6). O objetivo deste algoritmo é encontrar os dois serviços *on-line* mais semelhantes com a solução especificada no **Caso (1)**.

A exemplo de tais serviços atualmente *on-line* temos os descritos pelo *Profile* apresentados nas figuras 4.13, 4.14 e 4.12 (Figura 4.12 apresenta *Profile* da solução do **Caso (2)**), onde:

– Figura 4.13 apresenta o *Profile* OWL-S que descreve um serviço com as seguintes características:

Nome Pacote Viagem Ouro;

Entradas Os objetos dos tipos

Trecho (representa as cidade de origem e destino), *Carro* (representa o tipo de carro), *Pessoa* (representando as informações do usuário como nome, cpf e *hobbie*), *CartaoCredito* (representando informa-

ções sobre o cartão de crédito do usuário), *Evento* (representando informações sobre o evento que o usuário deseja participar) e *Periodo* (representando o mes, dia de chegada e retorno da viagem);

Saídas O objeto do tipo *Confirmacao* que representa as informações sobre as passagens aéreas, hotel, locação de carro e ingressos de eventos;

Propriedades Tempo de resposta, segurança e escalabilidade com qualidades *bom*, *excelente* e *excelente*;

– Figura 4.14 apresenta o *Profile* OWL-S que descreve um serviço com as seguintes características:

Nome Pacote Viagem Diamante;

Entradas Os objetos dos tipos *Trecho* (representa as cidade de origem e destino), *Carro* (representa o tipo de carro), *Pessoa* (representando as informações do usuário como nome, cpf e *hobbie*), *CartaoCredito* (representando informações sobre o cartão de crédito do usuário), *Evento* (representando informações sobre o evento que o usuário deseja participar) e *Periodo* (representando o mes, dia de chegada e retorno da viagem);

Saídas O objeto do tipo *Confirmacao* que representa as informações sobre as passagens aéreas, hotel, locação de carro e ingressos de eventos;

Propriedades Tempo de resposta, segurança e escalabilidade com qualidades *excelente*, *bom* e *excelente*;

Então, aplicando o algoritmo de similaridade supracitado como apresentado nas tabelas 4.8, 4.9 e 4.10. Onde na Tabela 4.8 a coluna **Similaridade** apresenta a similaridade entre os atributos da solução do Caso (1) e o *Profile* apresentado na Figura 4.13 tendo como similaridade total 0,8. Na Tabela 4.9 a coluna **Similaridade** apresenta a similaridade entre os atributos da solução do Caso (1) e o *Profile* apresentado na Figura 4.14 tendo similaridade total 0,7. Na Tabela 4.10 a coluna **Similaridade** apresenta a similaridade entre os atributos da solução do Caso (1) e o *Profile* apresentado na Figura 4.12 tendo similaridade total 0,5.

Verifica-se que as ontologias das Figuras 4.13, com similaridade 0.8 com a solução do Caso (1), e 4.14, com similaridade 0.7 com a solução do Caso (1), são as duas mais semelhantes, então elas serão enviadas para atividade *Decision* (Ver Seção 3.3.3) que irá aplicar os mecanismos de função de utilidade

```
– <rdf:RDF xml:base="http://139.82.24.82:8080/OWLRepository/PVO.owl">
  – <service:Service rdf:ID="Service">
    – <service:presents>
        <profile:Profile rdf:ID="Profile"/>
      </service:presents>
    – <service:describedBy>
        <process:AtomicProcess rdf:ID="Process"/>
      </service:describedBy>
    </service:Service>
  – <profile:Profile rdf:about="#Profile">
      <profile:serviceName>Pacote Viagem Ouro</profile:serviceName>
    – <profile:hasInput>
      – <process:Input rdf:ID="Periodo">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#Periodo
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="CartaoCredito">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#CartaoCredito
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="Trecho">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#Trecho
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="Carro">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Carro
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="Pessoa">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Pessoa
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="Evento">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Evento
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasOutput>
    – <process:Output rdf:ID="Confirmacao">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Confirmacao
        </process:parameterType>
      </process:Output>
    </profile:hasOutput>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Tempo de Resposta</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="Viagem.owl#bom"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Escalabilidade</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="Viagem.owl#excelente"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Seguranca</j.1:serviceParameterName>
            <j.1:sParameter rdf:resource="Viagem.owl#excelente"/>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      </profile:Profile>
    – <process:AtomicProcess rdf:about="#Process">
        <process:hasOutput rdf:resource="#Confirmacao"/>
        <process:hasInput rdf:resource="#Periodo"/>
        <process:hasInput rdf:resource="#CartaoCredito"/>
        <process:hasInput rdf:resource="#Pessoa"/>
        <process:hasInput rdf:resource="#Carro"/>
        <process:hasInput rdf:resource="#Evento"/>
        <process:hasInput rdf:resource="#Trecho"/>
        <service:describes rdf:resource="#Service"/>
      </process:AtomicProcess>
    </rdf:RDF>
```

Figura 4.13: OWL-S do Serviço Pacote Viagem Ouro

```
– <rdf:RDF xml:base="http://139.82.24.82:8080/OWLRepository/PVD.owl">
  – <service:Service rdf:ID="Service">
    – <service:presents>
        <profile:Profile rdf:ID="Profile"/>
      </service:presents>
    – <service:describedBy>
        <process:AtomicProcess rdf:ID="Process"/>
      </service:describedBy>
    </service:Service>
  – <profile:Profile rdf:about="#Profile">
      <profile:serviceName>Pacote Viagem Diamante</profile:serviceName>
    – <profile:hasInput>
      – <process:Input rdf:ID="Trecho">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#Trecho
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="Evento">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#Evento
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
    – <profile:hasInput>
      – <process:Input rdf:ID="Periodo">
        – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
            http://139.82.24.82:8080/OWLRepository/Viagem.owl#Periodo
          </process:parameterType>
        </process:Input>
      </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="Pessoa">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Pessoa
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="Carro">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Carro
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasInput>
    – <process:Input rdf:ID="CartaoCredito">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#CartaoCredito
        </process:parameterType>
      </process:Input>
    </profile:hasInput>
  – <profile:hasOutput>
    – <process:Output rdf:ID="Confirmacao">
      – <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
          http://139.82.24.82:8080/OWLRepository/Viagem.owl#Confirmacao
        </process:parameterType>
      </process:Output>
    </profile:hasOutput>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Escalabilidade</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="Viagem.owl#excelente"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Tempo de Resposta</j.1:serviceParameterName>
            <j.1:sParameter rdf:resource="Viagem.owl#excelente"/>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      – <j.1:serviceParameter>
        – <j.1:ServiceParameter>
            <j.1:serviceParameterName>Seguranca</j.1:serviceParameterName>
          – <j.1:sParameter>
              <j.0:Qualidade rdf:about="Viagem.owl#bom"/>
            </j.1:sParameter>
          </j.1:ServiceParameter>
        </j.1:serviceParameter>
      </profile:Profile>
    – <process:AtomicProcess rdf:about="#Process">
        <process:hasOutput rdf:resource="#Confirmacao"/>
        <process:hasInput rdf:resource="#Periodo"/>
        <process:hasInput rdf:resource="#CartaoCredito"/>
        <process:hasInput rdf:resource="#Pessoa"/>
        <process:hasInput rdf:resource="#Carro"/>
        <process:hasInput rdf:resource="#Evento"/>
        <process:hasInput rdf:resource="#Trecho"/>
        <service:describes rdf:resource="#Service"/>
      </process:AtomicProcess>
    </rdf:RDF>
```

Figura 4.14: OWL-S do Serviço Pacote Viagem Diamante

Tabela 4.8: Similaridade a solução do Caso (1) e a OWL-S da Figura 4.13

Atributos	**solução do Caso (1)**	**OWL-S da Figura 4.13**	**Similaridade**
Name	Pacote Viagem Evento	Pacote Viagem Ouro	0
Parameter	TempoResposta:bom	TempoResposta:bom	0
	Segurança:excelente	Segurança:excelente	1
	Escalabilidade:excelente	Escalabilidade:excelente	1
Inputs	Pessoa	Pessoa	1
	Trecho	Trecho	1
	CartaoCredito	CartaoCredito	1
	Carro	Carro	1
	Evento	Evento	1
	Periodo	Periodo	1
Outputs	Confirmacao	Confirmacao	1
Similaridade Total			0,8

Tabela 4.9: Similaridade a solução do Caso (1) e a OWL-S da Figura 4.14

Atributos	**solução do Caso (1)**	**OWL-S da Figura 4.14**	**Similaridade**
Name	Pacote Viagem Evento	Pacote Viagem Diamant	0
Parameter	TempoResposta:bom	TempoResposta:excelente	0
	Segurança:excelente	Segurança:bom	0
	Escalabilidade:excelente	Escalabilidade:excelente	1
Inputs	Pessoa	Pessoa	1
	Trecho	Trecho	1
	CartaoCredito	CartaoCredito	1
	Carro	Carro	1
	Evento	Evento	1
	Periodo	Periodo	1
Outputs	Confirmacao	Confirmacao	1
Similaridade Total			0,7

(Ver Seção 3.3.3), o qual irá retornar os dois serviços com maior importância, e reputação (Ver Seção 3.3.3), o qual irá retornar o serviço com maior reputação.

Função de Utilidade: Considerando as dimensões tempo de resposta (t), segurança (s) e escalabilidade (e) estão classificadas como *ruim*, *medio*, *bom* e *excelente* com escalas 0, 0.5, 0.75 e 1, respectivamente. Então, as utilidades são como descritas abaixo:

Pacote Viagem Ouro : Possui tempo de resposta *bom*, segurança *excelente* e escalabilidade *excelente*, com utilidade $u_t(bom) = 0.75$, $u_s(excelente) = 1$ e $u_e(excelente) = 1$, respectivamente;

Pacote Viagem Diamant : Possui tempo de resposta *excelente*, segurança

Tabela 4.10: Similaridade a solução do Caso (1) e a OWL-S da Figura 4.12

Atributos	**solução do Caso (1)**	**OWL-S da Figura 4.12**	**Similaridade**
Name	Pacote Viagem Evento	Pacote Viagem Resort	0
Parameter	TempoResposta:bom	TempoResposta:medio	0
	Segurança:excelente	Segurança:medio	0
	Escalabilidade:excelente	Escalabilidade:excelente	1
Inputs	Pessoa	Pessoa	1
	Trecho	Trecho	1
	CartaoCredito	CartaoCredito	1
	Veiculo	Não existe	0
	Evento	Não existe	0
	Periodo	Periodo	1
Outputs	Confirmation	Confirmation	1
Similaridade Total			0.5

bom e escalabilidade *excelente*, com escalas $u_t(excelente) = 1$, $u_s(bom) = 0.75$ e $u_e(excelente) = 1$, respectivamente.

Desta forma, tomando que as dimensões tempo de resposta, segurança e escalabilidade, tem pesos $w_t = 0.3$, $w_s = 0.3$ e $w_e = 0.4$, podemos calcular a importância de cada um dos serviços seguindo a equação apresentada na seção 3.3.3, como descrito abaixo:

Pacote Viagem Evento Ouro : 0.3*0.75 + 0.3*1 + 0.4*1= 0.925;

Pacote Viagem Evento Diamant : 0.3*1 + 0.3*0.75 + 0.4*1= 0.925;

Definimos que os dois serviços mais semelhantes devem ser enviados para o mecanismo de reputação. Sendo assim, as OWL-S descrevendo os serviços **Pacote Viagem Ouro** e **Pacote Viagem Diamante**, serão enviadas para o mecanismo de reputação.

Para implementar o mecanismos de função de utilidade a classe *TravelUtilityFunciton* (Ver Figura 4.15) foi criada, como extensão da classe abstrata *UtilityFunction*, e o método *getUtility* foi implementado a fim de criar uma *HashMap* que tem como *Key* o nome das dimensões de qualidade supracitadas (tempo de resposta, segurança e escalabilidade) e *Value* os seus respectivos graus de utilidade. Adicionalmente, ao instanciarmos a classe *TravelUtilityFunciton* dois parâmetros são fornecidos: (i) uma *HashMap* onde a *Key* é uma das dismensões de qualidade supracitadas e *Value* o peso dela, e (ii) o número de serviços a serem enviados para o próximo mecanismos foi definido como dois.

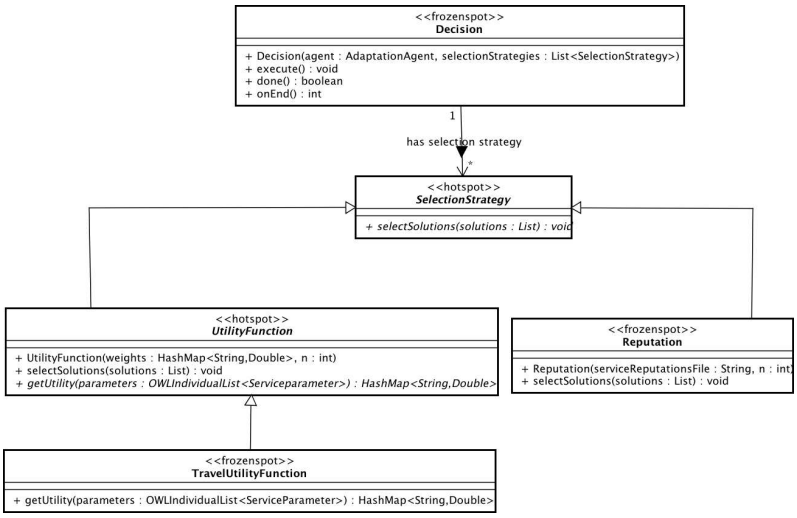

Figura 4.15: Atividade *Decision* de um APV

```
– <Services>
  – <Service>
    – <URI>
        file:/Users/baldoino/Desktop/Ontologias/OWL-S/PVO.owl
      </URI>
      <Reputation>0.9</Reputation>
    </Service>
  – <Service>
    – <URI>
        file:/Users/baldoino/Desktop/Ontologias/OWL-S/PVD.owl
      </URI>
      <Reputation>0.6</Reputation>
    </Service>
  </Services>
```

Figura 4.16: Reputação

Reputação: O mecanismos de reputação irá verificar o nome de cada serviço e verificar a reputação do mesmo. Então, considerando os serviços selecionados em questão e o arquivo XML apresentado na figura 4.16 . Como o **Pacote Viagem Ouro** possui uma maior reputação, ele será enviado para a atividade *Effector* (Ver Seção 3.3.4).

Para implementar o mecanismos de reputação a classe *Reputation* (Ver Figura 3.10), disponibilizada pelo JAAF-S, foi instanciada passando como parâmetro o endereço do arquivo XML supracitado e definindo o número de serviços a serem enviados para atividade *Effector* como um.

Na última atividade, chamada *Effector*, o APV envia uma mensagem para o agente *Manager* informando sobre o novo serviço, disponibilizá-o, configura o mecanismos de monitoramento para que o mesmo esteja apto

à monitorar o serviço selecionado e espera por novas requisições. Ou seja, os atuadores representados pelas classes *Notification* e *ServiceConfiguration* foram utilizados (Ver Seção 3.3.4).