UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Composição e Monitoração Automáticas e Contínuas de Serviços Web Não-Determinísticos

Gustavo Batoreu Valfre

Orientadores
Prof. Dr. Angelo Ernani Maia Ciarlini
Prof. Dr. Sean Wolfgand Matsui Siqueira

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2010

Composição e Monitoração Automáticas e Contínuas de Serviços Web Não-Determinísticos

Gustavo Batoreu Valfre

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓSGRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Angelo Ernani Maia Ciarlini, DSc, UNIRIO

Sean Wolfgand Matsui Siqueira, DSc, UNIRIO

Leonardo Guerreiro Azevedo, DSc, UNIRIO

Fernanda Baião, DSc, UNIRIO

Antonio Luz Furtado, PhD, PUC-Rio

José de Jesús Pérez Alcazar, DSc, USP

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2010

Valfre, Gustavo Batoreu.
V168 Composição e monitoração automáticas e contínuas de serviços web não determinísticos / Gustavo Batoreu Valfre, 2010.
139f.

Orientador: Angelo Ernani Maia Ciarlini.
Coorientador: Sean Wolfgand Matsui Siqueira.
Dissertação (Mestrado em Informática) – Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2010.

1. Serviços na Web. 2. Serviços na Web – Planejamento. 3. Composição automática de serviços Web. 4. Serviços Web – Monitoração e controle. 5. Serviços Web não determinísticos. I. Ciarlini, Angelo Ernani Maia. II. Siqueira, Sean Wolfgand Matsui . II. Universidade Federal do Estado do Rio de Janeiro (2003-). Centro de Ciências Exatas e Tecnologia. Curso de Mestrado em Informática. III. Título.

CDD – 004.678

DEDICATÓRIA

A minha esposa, por ter
permanecido ao meu lado, por
me mostrar como a vida é boa,
e por ter me dado o maior dos presentes,
a minha filha Luísa, que ainda não conheço,
mas já aprendi a amar.

AGRADECIMENTOS

A TODOS OS PROFESSORES, colegas de turma e pessoal da secretaria que participaram desta jornada, sempre solícitos. Meus sinceros agradecimentos.

AOS MEUS ORIENTADORES, que se dedicaram além do que eu poderia imaginar ser possível, principalmente, nos momentos de dificuldades e incertezas. Acreditando no meu trabalho, deram-me a confiança necessária, dividindo comigo as expectativas, conduzindo-me a maiores reflexões e grau de compreensão.

A TODOS OS COLEGAS DE TRABALHO. Sou-lhes bastante grato. Em especial, a Marcelo Costa da Rocha, meu superior direto, que me concedeu a oportunidade de tentar, ingressar, condições favoráveis para os estudos e o apoio necessário para concluir o curso e, Vera Duarte, por ter me dado a permissão e o incentivo em alcançar tal resultado. Sem a ajuda de vocês não seria possível.

A TODOS OS MEUS AMIGOS, que nesses últimos três anos, escutaram papos relacionados a mestrado e por vezes, não pudemos estar juntos.

A MINHA FAMÍLIA, pais, irmãos e parentes que, mesmo sem o churrasco, se mantiveram incansáveis em suas manifestações de apoio e carinho. Em especial, para minha mãe, Nadia Maria Batoreu, minha eterna fonte de inspiração, pela sua dedicação e pelo o que eu sou.

A MINHA QUERIDA ESPOSA Liliane Gouveia da Silva Lyra, porque soube tolerar e compreender o meu estranho mau humor e abriu mão de vários momentos importantes de nossas vidas para que eu pudesse me dedicar aos estudos, mesmo na nossa lua-de-mel. Meu eterno amor.

A Deus, por ter me guiado em mais uma jornada e, finalmente, a todos que, de uma forma ou de outra, me ajudaram a chegar até aqui. Muito obrigado.

VALFRE, Gustavo Batoreu. **Composição e Monitoração Automáticas e Contínuas de Serviços Web Não-Determinísticos**. UNIRIO, 2010. 139 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

Atualmente há um grande interesse na composição de Serviços Web principalmente por possibilitar integrações entre aplicações. A capacidade de gerar composições automaticamente permite utilizar Serviços Web existentes e gerar combinações para efetuar diferentes atividades mais rapidamente com um menor custo. As propostas encontradas na literatura utilizam diferentes técnicas para combinar Serviços Web, mas, na sua maioria, são aplicadas a ambientes (determinísticos) onde as possibilidades de composição são conhecidas e os resultados são previsíveis. Por outro lado, a necessidade de gerar e executar composições de serviços tende a ser contínua, pois novos objetivos a serem alcançados podem surgir em decorrência de novas demandas dos usuários, do resultado da execução de outros serviços ou da combinação desses dois fatores. Técnicas de Inteligência Artificial permitem que se infiram objetivos a serem atingidos e se criem planos para atingi-los por meio da composição de Serviços Web. No entanto, para se ter mais chances de atingir os objetivos de forma satisfatória, torna-se importante que o planejamento leve em conta o não-determinismo do ambiente, permitindo o tratamento adequado das contingências. O trabalho apresentado nesta Dissertação propõe a utilização de Planejamento e mecanismos de monitoração e controle, para o tratamento em tempo real de eventos não-determinísticos, utilizando um conjunto de regras de inferência para continuamente determinar as tarefas a serem realizadas. Os testes realizados demonstraram resultados satisfatórios, com pouca necessidade de configuração. A geração automática de novas composições ocorreu com bom desempenho e tamanho reduzido quando comparadas às geradas em outras abordagens.

Palavras-chave: Composição Automática, Planejamento, Serviços Web, Monitoração e Controle, Não-determinismo.

ABSTRACT

Nowadays there is great interest in Web Services composition mainly because it allows the integration between applications. The ability to automatically generate compositions enables applications to combine existing Web Services in order to perform different activities more quickly at a lower cost. The approaches found in the literature use different techniques to combine Web Services, but mostly apply to (deterministic) environments where composition possibilities are known and the results are predictable. Moreover, the need to generate and execute Web Services compositions tends to be continuous, because new goals to be achieved may arise due to new requests from the users, the result of the execution of other services or the combination of both. Artificial Intelligence techniques allow the inference of goals to be achieved and the generation of plans to achieve them by means of Web Services composition. However, to have more chances to successfully achieve the goals, it is important for the planning process to take into consideration the non-determinism of the environment, allowing an appropriate treatment of contingencies. This work proposes the use of Planning algorithms combined with monitoring and control mechanisms to handle real-time nondeterministic events using a set of policies to continually determine the tasks to be performed. Tests conducted showed satisfactory results, with little need of configuration. Automatic generation of new compositions occurred with good performance and reduced size when compared to those generated by other approaches.

Keywords: Automatic Composition, Planning, Web Services, Monitoring and Control, Nondeterminism.

Sumário

Índice de Figuras

Índice de Códigos

Introdução

Serviços são utilizados por instituições como forma de disponibilizar funcionalidades de uma aplicação para outra permitindo a sua invocação remota para concluir uma determinada atividade (SIRIN *et al.*, 2004). Porém, o uso de funcionalidades individualmente, na maioria dos casos, não é suficiente para concluir atividades, sendo necessário combiná-las através da criação de composições de serviços (MILANOVIC *et al.*, 2004).

As composições de serviços permitem a realização de atividades e processos mais complexos e são expostos como serviços compostos para aplicações consumidoras, possibilitando que invoquem um único serviço ao invés da construção de uma lógica de invocação para cada serviço individual. A redundância de código entre aplicações é reduzida, uma vez que uma composição criada pode ser compartilhada entre diversas aplicações consumidoras (SIRIN *et al.*, 2003).

A tarefa de criar uma nova composição nem sempre é simples de ser realizada, pois exige um conhecimento prévio dos objetivos desejados a serem alcançados, quais Serviços Web deverão ser invocados e a seqüência correta de invocação (GANNOD *et al.*, 2007).

As composições devem ainda estar preparadas para lidar com eventuais falhas ou imprevistos, visto que os Serviços Web invocados podem sofrer alterações no decorrer do tempo ou algum Serviço Web pode não estar mais disponível (HARNEY *et al.*, 2008). Um grande esforço pode ser exigido para garantir que composições estejam operando corretamente conforme o número de Serviços Web e a quantidade de consumidores crescem.

Dessa forma, muito estudo vem sendo realizado para encontrar formas mais automatizadas para a construção de composições, deixando que dispositivos computacionais efetuem a atividade de composição de Serviços Web para satisfazer um determinado objetivo (CHAN *et al.*, 2007).

Este trabalho expõe alguns dos desafios existentes em composições automáticas de serviços, especialmente, em ambientes não-determinísticos – quando não existe a garantia de que o estado final alcançado seja determinado simplesmente pelo estado atual e pelas ações executadas na composição. Estes desafios estão concentrados, principalmente, em capacidades para expressar objetivos, o correto sequenciamento de Serviços Web e a verificação dos resultados obtidos. É proposta uma solução para a geração de composições automáticas, de forma contínua, permitindo a monitoração de eventos que ocorram no ambiente e o acionamento de regras de inferência para a detecção de quando a geração e a execução de uma nova composição é necessária, seja para resolver conflitos ou para atender a novas demandas.

1.1 Motivação

A geração de uma composição automática deve alcançar uma solução satisfazível, através do encadeamento lógico de ações que, ao serem executadas, atendam às condições estabelecidas do objetivo informado. Porém, algumas condições podem não ser alcançadas após a execução da composição, tipicamente, pelo fato de composições serem aplicadas em ambientes não-determinísticos.

Uma ação executada pode levar a diferentes estados, podendo haver estados indesejados, que não levam ao resultado esperado. Assim, uma composição que leve em conta o não-determinismo poderá acionar uma ação alternativa para permitir o seu prosseguimento, aumentando as chances de sucesso. Entretanto, pode não haver uma ação alternativa possível, o que inviabiliza que a composição possa concluir o seu objetivo.

Se uma composição não pode prosseguir, deve desfazer atividades realizadas previamente para manter a consistência do ambiente. Entretanto, o desfazimento pode

enfrentar problemas semelhantes, não possibilitando que ações sejam desfeitas. Situações de inconsistência podem ser momentâneas, existindo a possibilidade de serem corrigidas num período de tempo posterior, dependendo do objetivo.

Por outro lado, um objetivo pode possuir uma relevância momentânea ou futura. A relevância momentânea está relacionada com objetivos que devem ser alcançados naquele instante para serem válidos, como por exemplo, o atendimento de uma emergência médica, onde é necessária a verificação de um hospital mais próximo e o acionamento de uma equipe para o pronto atendimento do paciente. Se o paciente não for atendimento do momento do acionamento, não haverá a necessidade de um atendimento posterior. Um objetivo com relevância futura pode ser exemplificado através do agendamento de uma consulta médica. Supondo que um agendamento tenha sido realizado e algum tempo depois se verifica que o médico não poderá atender o cliente na data programada, permanecendo a necessidade do cliente, uma data alternativa pode ser uma solução válida.

Quando a relevância é momentânea, deve ser alcançado num determinado instante previsto, caso contrário, havendo sucesso ou não, a relevância do objetivo é extinta, não importando mais o seu alcance. Para um objetivo que possui uma relevância futura, o objetivo permanece válido durante um período de tempo podendo haver novas tarefas relacionadas a serem executadas de modo que possa ser atingido. Se o objetivo inicial ainda se encontra relevante, uma nova composição pode ser gerada, mesmo com eventuais atrasos ou prejuízos decorrentes das execuções de composições anteriores que falharam. A nova geração pode levar em consideração falhas anteriores e a possibilidade de pequenos ajustes no objetivo para alcançar um resultado minimamente satisfatório (objetivos ajustados).

Por fim, os objetivos de relevância futura, alcançados por composições podem estar relacionados com programações de atividades futuras ou recursos que deverão ser necessários em um determinado período. Dessa forma, o resultado obtido pode tornar-se inválido à medida que uma das atividades ou recursos previstos não esteja mais disponível ao longo do tempo. Assim, torna-se necessário detectar quando o resultado obtido por serviços executados precisa ser modificado em virtude de

alterações do ambiente – quando existe a ocorrência de um evento que altera o estado do mundo.

Logo, se o estado alcançado por uma composição não satisfaz um objetivo ou está em conflito com condições do ambiente, novas questões surgem: dado um objetivo que não pode ser alcançado, como atingir um objetivo ajustado, utilizando as limitações conhecidas? Se recursos e atividades previstas tornam-se indisponíveis, como detectar e gerar composições para tratar tais situações?

Esses são os problemas que serão tratados com a geração contínua de composições automáticas de Serviços Web.

1.2 Proposta

Para a criação de uma arquitetura que facilite a geração e execução contínua de composições automáticas de Serviços Web que levem em conta o não-determinismo foram exploradas, inicialmente, outras soluções existentes de composições de serviços, como o ASTRO (MARCONI *et al.*, 2008), uma solução para integração de processos de negócio e o CASCOM (KLUSCH *et al.*, 2005), uma solução para composição de Serviços Web Semânticos. Apesar de solucionarem outros aspectos relacionados com a composição de serviços, não possuíam a capacidade de tratar a geração contínua de composições. Além disso, algumas, como o CASCOM, permitem o tratamento inclusive de descrições semânticas dos serviços, mas não tratam explicitamente o não-determinismo no processo de planejamento. Outras, como o ASTRO, levam em conta o não-determinismo no planejamento, mas não são focadas na composição em tempo real dos serviços. Além disso, foram encontradas dificuldades (como o acesso ao código e a incompatibilidade com novos padrões para especificação e execução de serviços) que inviabilizaram a extensão delas para incorporar as necessidades levantadas.

Este trabalho propõe uma nova solução para gerar tanto a composição automática, quanto as questões de monitoração e controle das composições.

O módulo de monitoração e de controle permanece verificando constantemente as composições que estão sendo executadas e notificações externas que são recebidas através de Serviços Web. Quando uma situação específica é identificada (mapeada em um conjunto de estados), uma regra de inferência associada é acionada podendo requisitar a geração automática de uma nova composição para tratar a situação encontrada.

Para a geração da composição automática, está sendo utilizada uma técnica de planejamento com não-determinismo da Inteligência Artificial, permitindo a geração de planos sofisticados, visando maximizar as chances de se alcançar o objetivo informado (MARCONI *et al.*, 2008). O processo de planejamento faz uso de uma rede hierárquica de tarefas, que usa conhecimento específico do domínio para reduzir o espaço de busca de uma solução. Os planos gerados são árvores onde cada nó corresponde a uma tarefa básica, diretamente associada para a invocação de uma operação de um Serviço Web básico, e cada aresta corresponde a uma condição do mundo modelado, que habilita ou não a tarefa básica a que está ligada. No mapeamento entre tarefas básicas e invocações de operações Serviços Web, são usados parâmetros retornados pelo serviço para indicar o efeito gerado pelo serviço e a reflexão na modelagem do mundo. Após a execução de uma tarefa básica, com base no modelo do mundo já atualizado, testam-se as condições de modo a determinar a próxima tarefa a executar.

Com base na alternância entre inferência de objetivos a serem atingidos, de planejamento não-determinístico para a composição dos serviços em tempo real e da execução controlada das composições, procura-se ter flexibilidade para aumentar o nível de automação na utilização de Serviços Web.

1.3 Estrutura

O Capítulo 1 introduziu as necessidades de se utilizar uma abordagem para Composição e Monitoração Automáticas e Contínuas de Serviços Web Não-Determinísticos e uma solução proposta.

O Capítulo 2 descreve toda a fundamentação teórica relacionada com o assunto de Serviços Web, conceitos de Composição de Serviços Web e planejamento, uma técnica de Inteligência Artificial para auxiliar na atividade de composição automática.

O Capítulo 3 apresenta a arquitetura proposta para a solução, descrevendo o funcionamento da solução em tempo de Geração de Composições, em tempo de Execução de Composições e, em tempo de Monitoração e Controle.

O Capítulo 4 descreve a implementação da solução e seus módulos, bem como, o nível de interação entre os módulos.

O Capítulo 5 descreve a utilização da solução em um cenário específico de marcação de viagens e discute as características de sua utilização.

O Capítulo 6 efetua a conclusão do trabalho e propõe novas pesquisas.

2. Fundamentação Teórica

No início da década de 90, comunidades de desenvolvimento de aplicações distribuídas buscavam formas mais eficientes para a comunicação entre aplicações. Os avanços providos pela programação orientada a objetos associados com as capacidades de comunicação dos protocolos de redes possibilitaram que fornecedores desenvolvessem protocolos mais específicos, como o CORBA (YANG *et al.*, 1996) e o DCOM (DCOM, 2010).

O objetivo do CORBA (*Common Object Request Broker Architecture*) (YANG *et al.*, 1996) é definir uma arquitetura que disponibilize comunicação entre clientes e servidores em ambientes heterogêneos. O DCOM (DCOM, 2010) é uma extensão distribuída do COM (*Component Object Model*) da Microsoft permitindo que uma aplicação COM interaja com um objeto COM remoto como se o objeto estivesse residindo localmente.

Apesar da semelhança entre o funcionamento do DCOM e CORBA, detalhes no seu desenvolvimento impedem a interoperabilidade entre os dois padrões. Além disso, apesar de terem sido desenvolvidos em diferentes plataformas, qualquer solução desenvolvida sobre esses protocolos permanecerá dependente de um fornecedor. Uma aplicação desenvolvida em DCOM restringirá que todos os nós participantes estejam rodando sobre o Sistema Operacional *Windows*. No caso de CORBA, cada nó necessitará executar o mesmo produto ORB.

Com o advento da Internet, a necessidade de comunicação entre aplicações tornou-se mais crítica, empresas desejavam integrar seus sistemas além dos seus domínios, utilizando os diferentes meios de telecomunicações disponíveis e protocolos de rede. Pessoas queriam ter acesso a suas informações a partir de qualquer lugar, utilizando diferentes dispositivos, com capacidades limitadas de processamento.

A busca por desenvolvedores especializados em padrões proprietários também era um fator que dificultava as soluções. As soluções exigiam ser agnósticas a fornecedor, plataforma e linguagem.

Todos esses desafios motivaram o desenvolvimento de um novo modelo de computação distribuída em padrões abertos da Internet. Em Janeiro de 2002, o *World Wide Web Consortium* (W3C) criou um grupo de trabalho, contendo pessoas de diversos fornecedores e universidades, para estabelecer uma arquitetura de Serviços Web.

Segundo o W3C, Serviços Web fornecem uma forma padrão para inter-operação entre diferentes aplicações, executadas em variadas plataformas e frameworks (BOOTH *et al.*, 2004).

A forma com que os termos serviços e Serviços Web são empregados pode gerar confusão. No texto, o termo serviço refere-se à capacidade de aplicações exporem funcionalidades, enquanto que, o termo Serviços Web refere-se à forma com que um serviço é desenvolvido, baseando-se no padrão *Web Services*, publicado pelo W3C (CHINNICI *et al.*, 2007).

2.1. Serviços

O termo serviço cobre um domínio de produtos e atividades intangíveis distintas que é difícil descrever em uma simples definição. Do ponto de vista de aplicações, um serviço descreve uma função específica de uma aplicação que é disponibilizada para outra. Um serviço deve abstrair complexidades da aplicação que o está provendo. Deve também fornecer uma interface padronizada para que o consumidor possa utilizá-lo de forma concisa para alcançar o resultado esperado. Pode ser classificado como um serviço simples ou composto.

Um serviço é dito simples (LU *et al.*, 2006), quando possibilita acesso a atividades atômicas de um único sistema, que não podem ser decompostas em outras atividades, sendo assim, funcionalmente indivisível (WU *et al.*, 2003).

Um serviço composto (LU *et al.*, 2006) é a combinação de serviços simples ou de outros serviços compostos, visando disponibilizar uma nova funcionalidade através de um encadeamento lógico de invocação dos serviços e troca de informações. A Arquitetura Orientada a Serviços descreve como deve ser o relacionamento entre serviços e aplicações.

2.1.1. Arquitetura Orientada a Serviços

A Arquitetura Orientada a Serviços (SOA) padroniza a forma com que as aplicações devem prover e consumir Serviços, permitindo o reuso de ativos (PAPAZOGLOU *et al.*, 2007) e a criação de novos Serviços a partir de uma infra-estrutura de sistemas existentes.

Para a indústria em geral, essa característica foi essencial, pois permitiu que aplicações heterogêneas, antes isoladas, pudessem expor suas funcionalidades, de forma padronizada, para outros sistemas consumidores (GANNOD *et al.*, 2007). Aplicações legadas, que deveriam ser reescritas para acompanhar a evolução tecnológica, ganharam sobrevida e nova importância à medida que novos consumidores utilizavam suas funcionalidades.

Um importante aspecto da abordagem SOA é a separação da interface do serviço do seu desenvolvimento propriamente dito, não importando a tecnologia que foi utilizada para o desenvolvimento do serviço. Isso permite que um serviço existente possa ser consumido por diferentes consumidores sem a necessidade de adequações, diminuindo consideravelmente o custo, tempo e complexidade de novas soluções.

SOA prevê que os serviços sejam disponibilizados em um repositório para permitir que sejam encontrados e utilizados por outras aplicações. A figura 2-1 apresenta o repositório de serviços e suas operações básicas. O consumidor de serviço utiliza a operação de pesquisar para encontrar serviços relativos a uma condição. O provedor de serviço utiliza a operação registrar para publicar um serviço existente.

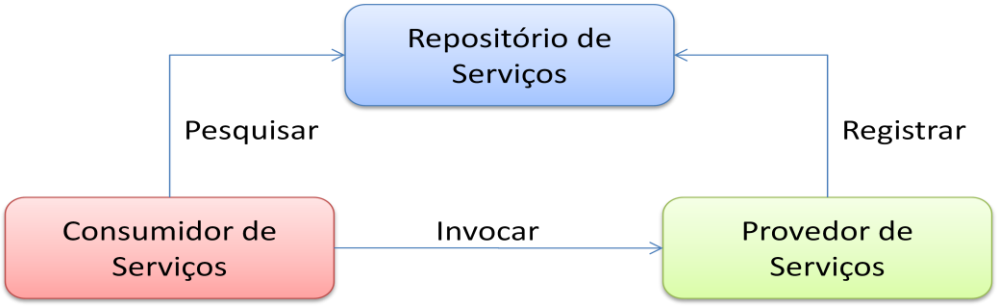

Figura 2-1 – Operações para utilização de serviços

O repositório de serviços é uma espécie de páginas amarelas, onde o serviço é catalogado e são inseridas informações sobre atividades desempenhadas pelo serviço, como deve ser utilizado e a própria interface do serviço (JOSUTTIS, 2007). Serviços podem ser descobertos em dois momentos distintos: em tempo de desenvolvimento ou em tempo de execução.

Em tempo de desenvolvimento, o desenvolvedor deverá identificar, no repositório de serviços, qual serviço será utilizado e importar, na aplicação cliente, as definições da interface do serviço, gerando os mapeamentos necessários. Em tempo de execução, a aplicação cliente já conhece as operações que serão invocadas do serviço.

Descobertas de serviços realizadas em tempo de execução exigem que a aplicação cliente tenha a capacidade de identificar o serviço mais adequado para a função que deseja executar, buscar a descrição da interface do serviço no repositório e efetuar o mapeamento entre a sua estrutura de dados interna com a estrutura esperada pelo serviço. Assim, poderá executar a invocação propriamente dita do serviço selecionado.

Enquanto, na primeira forma de descobrimento, o desenvolvedor identifica o serviço mais adequado, no descobrimento em tempo de execução, a aplicação consumidora deve possuir a capacidade de efetuar essas tarefas num curto período de tempo e exige mecanismos sofisticados de inferência. Por outro lado, o descobrimento em tempo de execução permite que sejam utilizados os serviços mais relevantes no momento da sua utilização, ou ainda, a última versão do serviço registrado no repositório. Como os serviços podem sofrer alterações no decorrer do

tempo, todos os consumidores que possuem a interface de um serviço previamente mapeado deverão também ser atualizados. Presume-se que um serviço seja reutilizado por muitos consumidores. Dessa forma, a atividade de atualização pode demandar bastante esforço.

Para que a reutilização de um serviço seja possível, devem ser oferecidos serviços relevantes para os consumidores. Porém, como qualquer funcionalidade pode ser transformada em serviço simples, definir a interface do serviço no nível de abstração correto é um desafio (FEUERLICHT, 2006).

Serviços compostos também são previstos pelo SOA conforme demonstra a Figura 2-2.

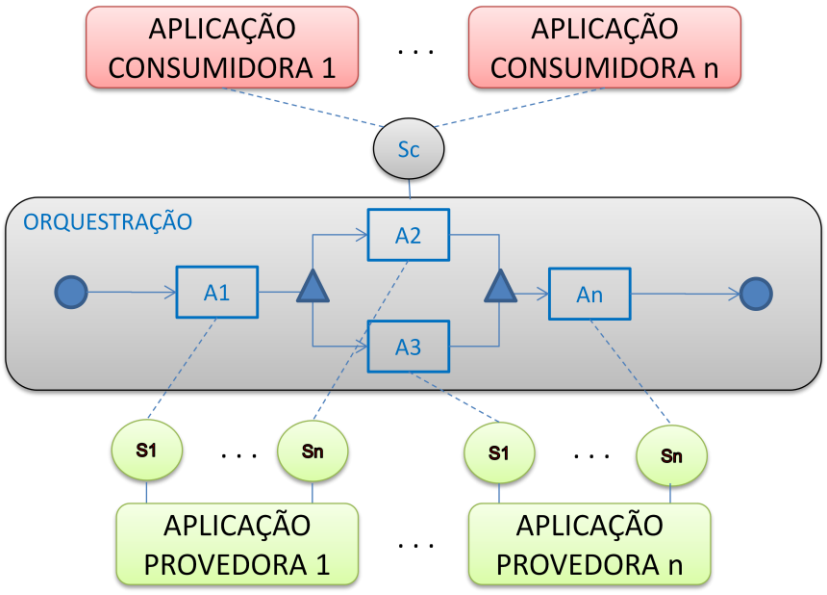

Figura 2-2 – Composição de Serviços

A Figura 2-2 exemplifica um conjunto de aplicações expondo serviços S para um serviço composto Sc. A orquestração possui um fluxo lógico de ações A, que estão associadas com serviços S. A orquestração faz parte do serviço composto Sc que está sendo consumido por um conjunto de aplicações consumidoras.

O Serviço composto ao ser invocado por um dos consumidores instancia o processo de orquestração que decide quais serviços chamar na ordem do fluxo e de acordo com os processos decisórios existentes (representados pelos triângulos).

Uma ação A pode também estar relacionada com serviços compostos, permitindo a combinação desses e gerando assim serviços compostos que realizam mais atividades para solucionar problemas maiores.

SOA não é exatamente uma arquitetura nova. O seu conceito aproxima-se muito das características de programação orientada a objetos, a qual já podia ser utilizada com qualquer tecnologia para o desenvolvimento de aplicações distribuídas (JOSUTTIS, 2007). Entretanto, SOA ganhou muita força recentemente com a popularização de Serviços Web, que alavancou e disseminou a construção de aplicações distribuídas.

2.2. Serviços Web

Um Serviço Web é uma aplicação que expõe uma função acessível através de tecnologias Web utilizando padrões abertos para descrevê-lo (CHINNICI *et al.*, 2007). Apesar de tecnologicamente não trazer nenhuma grande inovação e ser relativamente simples, essa tecnologia conquistou um grande número de desenvolvedores e a grande maioria dos fornecedores atualmente incorpora capacidade de integração em suas aplicações e ferramentas utilizando Serviços Web.

Os Serviços Web fornecem várias características positivas. Entre elas, pode ser destacada a independência de fornecedores. Diferentemente do DCOM e do CORBA, o Serviço Web foi criado para ser utilizado sobre qualquer tecnologia e plataforma, utilizando um padrão aberto, bem difundido, o XML (*eXtensible Markup Language*) (BRAY *et al.*, 2008).

O XML é uma linguagem que permite a criação de diferentes tipos de estruturas de dados e a definição de relações entre essas estruturas (BRAY *et al.*, 2008). Essa característica permite que dados sejam serializados em um formato de mensagem que é facilmente descodificável por qualquer plataforma. Essa foi uma vantagem considerável sobre outras tecnologias, pois torna mais fácil definir novos formatos de

dados com base em formatos existentes. Além disso, é uma linguagem virtualmente suportada por qualquer plataforma de computação.

Vários fornecedores aderiram ao padrão de Serviços Web, o que permitiu a sua interoperabilidade entre diferentes plataformas (IRANI, 2001), não existindo limitações entre Sistemas Operacionais, Linguagens de Programação ou exigências específicas de rede e hardware.

A sua facilidade de uso rapidamente conquistou um grande número de adeptos (GOTTSCHALK *et al.,* 2000), o que ajudou a difundir o padrão tanto a nível acadêmico, pois foram vislumbradas novas possibilidades de soluções, quanto a nível comercial, permitindo ganhos de produtividade e diminuição de custos com integrações de sistemas.

O fato de ser um padrão aberto facilitou ainda a proposição e a adoção de extensões sobre o próprio padrão de Serviços Web, as quais foram projetadas para tratar questões importantes para os sistemas baseados nesses serviços, tais como a segurança, o controle transacional, a semântica e a composição de serviços. Esses padrões formam a pilha de tecnologias de Serviços Web (ERL, 2005).

2.2.1. Tecnologias de Serviços Web

As tecnologias de Serviços Web podem ser dispostas em um modelo de camadas que é conhecido como a pilha da Arquitetura de Serviços Web (BOOTH *et al.*, 2004).

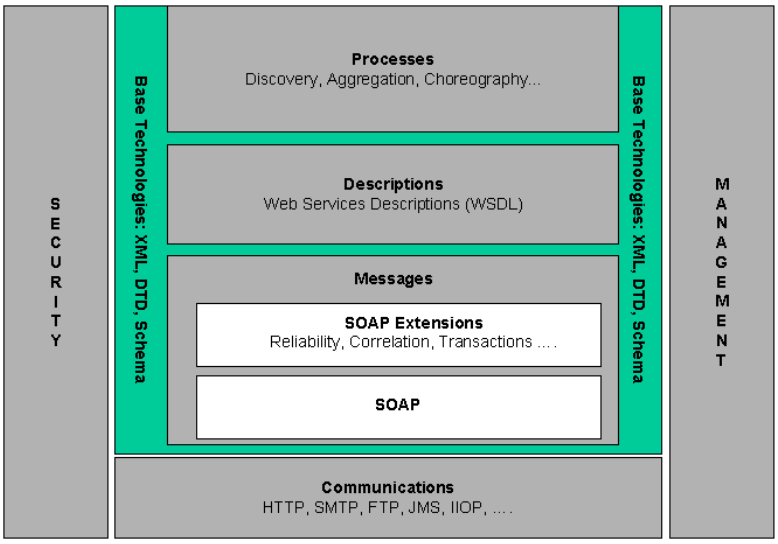

Figura 2-3- A pilha de arquitetura de Serviços Web (BOOTH *et al.*, 2004)

A pilha de arquitetura de Serviços Web (Figura 2-3) é composta por um conjunto de especificações abertas oriundas de padrões já existentes da Internet ou especificações que ganharam grande aceitação e se tornaram padrões (BOOTH *et al.*, 2004). A pilha básica é composta dos padrões: HTTP, XML, SOAP, WSDL e UDDI.

Na base da pilha, encontra-se o HTTP (*Hypertext Transfer Protocol*), um protocolo baseado em RPC (*Remote Procedure Call*), largamente utilizado pela Internet. Em seguida, o SOAP (*Simple Object Access Protocol*) (GUDGIN *et al.*, 2007), um protocolo de mensagem baseado em XML e independente de plataforma. O WSDL (*Web Services Description Language*) é uma linguagem para a definição de interfaces baseada em XML e que descreve Serviços Web, em termos de suas operações e das estruturas de dados utilizadas pelas mensagens enviadas e recebidas. Estas especificações são apresentadas em detalhes, a seguir.

Apesar de a tecnologia de Serviços Web prever o conceito de independência de protocolos de transporte de dados, o protocolo predominantemente utilizado atualmente é o HTTP. Por ser o principal protocolo utilizado para a Web e possuir facilidades para navegar sobre firewalls, tornou-se o protocolo de transporte mais comum no uso de Serviços Web. Outros protocolos adicionais podem ser utilizados, incluindo o SMTP (*Simple Mail Transfer Protocol*), o FTP (*File Transfer Protocol*) e até mesmo o TCP (*Transmission Control Protocol*), mas na prática existem poucas soluções que utilizam esses protocolos (WEERAWARANA, 2005).

Quanto à utilização de Serviços Web, um ponto crítico a ser levado em consideração é que HTTP não é um protocolo de transporte com garantia. Por ser um protocolo originalmente concebido para a apresentação de páginas web, não lhe foram conferidas características de garantia de entrega dos dados enviados, controle de duplicidade e mecanismos transacionais. Dessa forma, é possível que uma mensagem seja perdida em decorrência de indisponibilidades na rede ou qualquer outro problema. Nesse caso, quando existe a necessidade de utilização de Serviços Web sobre HTTP para aplicações de missão crítica, então lógica adicional ou outras ferramentas devem ser consideradas para garantir a troca de mensagens.

SOAP (*Simple Object Access Protocol*) (GUDGIN *et al.,* 2007) fornece um protocolo de empacotamento para a troca de estruturas de dados entre Serviços Web sobre um protocolo de transporte, tal como HTTP. SOAP define um formato baseado em XML para a especificação de mensagens que podem ser trocadas por Serviços Web e ações que devem ser efetuadas quando uma mensagem SOAP é recebida.

Serviços Web utilizam o WSDL para especificar um contrato de serviço. WSDL possui três aspectos: operações do serviço disponíveis, mensagens que o serviço irá aceitar e o protocolo através do qual o consumidor deve acessar o serviço. O contrato do serviço consiste da descrição de um conjunto de endereços de destinatários (*endpoints*), que operam com mensagens e especificações de como um documento XML deve ser formatado quando é enviado para os *endpoints* (CHINNICI *et al.*, 2005). WSDL usa o termo *port* para descrever um *service endpoint* para uma mensagem. WSDL descreve o contrato para um serviço como uma coleção de portas que o serviço possui disponível.

Serviços Web suportam a funcionalidade de descobrimento de serviços. O UDDI (*Universal Description, Discovery and Integration*) (CLEMENT *et al.*, 2004) é um registro de Serviços Web que apóia o descobrimento dinâmico e provê descrições para os Serviços Web registrados. O provedor e o consumidor de Serviços Web podem usar SOAP e HTTP para publicar e recuperar informações sobre serviços no registro.

2.3. Serviços Web Semânticos

Atividades de seleção de Serviços Web, geração de composições e integração automática realizadas por computador, sem auxílio de seres humanos, necessitam de mais artifícios do que os padrões tradicionais de Serviços Web oferecem. São necessárias descrições semânticas dos serviços e capacidades de inferência sofisticadas para realizar essas atividades (SIRIN, 2003).

Para endereçar essas necessidades, a comunidade de Web Semântica iniciou o trabalho de padronização de Serviços Web Semânticos. A proposta é que através de descrições das capacidades e características dos Serviços Web, sem ambigüidade e de forma que a máquina possa interpretar, seja possível a realização automática das atividades.

2.3.1. A Influência da Web Semântica

O objetivo da Web Semântica é fornecer capacidades de inferência para a máquina interpretar o conteúdo disponibilizado na web, utilizando um vocabulário associado a este conteúdo (BERNERS-LEE *et al.,* 2001). Dessa forma, agentes de software poderiam analisar o conteúdo e alcançar conclusões que iriam bem além das análises sintáticas realizadas por mecanismos de busca atualmente.

A idéia geral é que ao publicar um conteúdo, o responsável incluísse uma série de anotações acerca do conteúdo, permitindo que a máquina pudesse ler e efetuar associações. Entretanto, esse assunto ainda gera controvérsias, existindo uma corrente de céticos que não acredita ser possível colocar em prática tal atribuição. De fato, até o momento, ainda não está claro como o assunto irá evoluir. Já existem, porém, muitas linguagens propostas pela comunidade de web semântica que são aplicáveis a Serviços Web.

O RDF (*Resource Description Framework*) (LASSILA, 1999) e a OWL (*Web Ontology Language*) (MCGUINNESS *et al.*, 2004), são duas linguagens que estão sendo desenvolvidas com a participação de pesquisadores, empresas e mantidas pelo W3C, com a finalidade de estruturar e associar dados de forma efetiva para descoberta,

composição e reuso por diferentes aplicações. Essa estruturação permite a representação de significado e a estruturação de conteúdos através de relacionamentos entre conceitos.

Em RDF são utilizados conjuntos de triplas (sujeito, verbo e objeto) para expressar significado. Um documento RDF efetua afirmações sobre coisas em particular (pessoas, objetos,...) com propriedades (tais como "é uma irmã de", "é o autor de") com certos valores (outra pessoa, outro objeto) (BERNERS-LEE *et al.*, 2001). Cada sujeito, verbo ou objeto recebe um identificador único (URI) para que se possa identificá-lo corretamente. Entretanto, como é possível a utilização de identificadores distintos (por exemplo, sinônimos) para o mesmo conceito, é necessário um mecanismo que possa correlacioná-los, chamado de ontologias.

As ontologias são documentos que definem taxonomias (isto é, classes de objetos e seus relacionamentos diretos) e um conjunto de regras de inferência, que permitem a dedução de novos relacionamentos e dependências. As ontologias podem ser escritas em OWL e permitem aumentar a descrição dos Serviços Web (MCILRAITH *et al.*, 2001).

2.3.2. Serviços Web e a Web Semântica

Serviços Web são puramente sintáticos por natureza. Dessa forma, para permitir um maior nível de compreensão pela máquina, a Web semântica permite estendê-los para um novo paradigma, chamado de Serviços Web Semânticos.

Os Serviços Web Semânticos prevêem funcionalidades para a modelagem conceitual de serviços, incluindo a descrição de estados e comportamentos específicos, para a dedução e inferência de relacionamentos entre objetos, para a descoberta de serviços por nível de relevância, e para a mediação de processos e a composição de serviços. Além disso, prevêem uma arquitetura extensível a outros padrões. Existem diferentes propostas para Serviços Web Semânticos, incluindo as linguagens: OWL-S, WSMO, entre outras.

OWL-S (*Semantic Markup for Web Services*) é uma Ontologia específica em OWL para Serviços Web que permite usuários e agentes de software automaticamente descobrirem, comporem e monitorarem recursos web oferecidos pelos serviços. Interfaces de Serviços Web podem ser descritas incluindo *input*, *outputs*, *preconditions* e *effects* (IOPE) para auxiliar atividades de composição automática. O OWL-S permite construtores, tais como: *sequence, if-then-else, fork repeat-until* e etc. para definir um processo composto.

O WSMO (*Web Service Modeling Ontology*) propõe o uso de ontologias para descrever: Serviços Web, aspectos funcionais e comportamentais de um Serviço Web, objetivos que representam desejos do usuário e problemas relacionados com interoperabilidade (mediações) entre diferentes elementos do WSMO.

Embora, OWL-S e WSMO compartilhem da visão sobre ontologias serem essenciais para funcionalidades de automação do uso de Serviços Web, elas possuem grandes diferenças na abordagem que utilizam para alcançar esse resultado. Essas diferenças vão desde a forma com que as ontologias são definidas até a forma com que é realizada a interação entre Serviços Web para a geração de Composições.

2.4. Composição de Serviços Web

Composições permitem descrever relações de dependência e comportamento entre Serviços Web (Coreografia) e, fluxos lógicos de invocação de Serviços Web (Orquestração). Assim, é possível realizar novas atividades através da combinação de funcionalidades existentes. As composições devem ser preparadas para lidar com eventuais falhas ou imprevistos, uma vez que, os Serviços Web podem sofrer alterações no decorrer do tempo ou podem não estar mais disponíveis.

A geração de composições manuais de Serviços Web pode demandar um esforço considerável, de acordo com a quantidade de Serviços Web envolvidos e a lógica necessária para manipulá-los. Muitos trabalhos vêm sendo realizados para encontrar formas mais automatizadas para a construção de composições. Com base na sua forma

de geração, uma composição pode ser classificada como estática, semi-automática ou automática (RAO *et al.*, 2004).

Na modelagem de fluxos de composição estática, realizada por seres humanos, os Serviços Web a serem utilizados são pré-selecionados e a lógica de manipulação é construída manualmente. Em cenários com números elevados de Serviços Web, selecionar um serviço pode demandar tempo (SIRIN *et al.*, 2003). Assim, composições semi-automáticas auxiliam o desenvolvedor com mecanismos que fornecem facilidades de indicação de qual serviço selecionar e sugestões para a geração do fluxo.

As composições automáticas são geradas por recursos computacionais e, dessa forma, necessitam de pouca ou nenhuma intervenção manual. Essa característica permite ainda a geração sobre demanda, onde composições são geradas à medida que surge a necessidade de se solucionar um problema em tempo de execução. Composições estáticas e semi-automáticas são desenvolvidas previamente, para, posteriormente, serem disponibilizadas em ambiente de execução.

2.4.1. Orquestração

Em composições, a orquestração é o ponto central de execução dos Serviços Web. Define seqüências de invocação dos Serviços Web e alternativas possíveis de acordo com o resultado obtido da sua invocação.

Existem muitas linguagens para a descrição de orquestrações de Serviços Web, sendo WS-BPEL (*Web Services Business Process Execution Language*) a mais utilizada (WEERAWARANA *et al.*, 2005) no mercado atualmente. A linguagem surgiu em julho de 2002 com a especificação do BPEL4WS 1.0 a partir dos esforços de junção das especificações XLANG (THATTE, 2001) da Microsoft e WSFL (LEYMANN, 2001) da IBM, com o objetivo de unir as melhores características da cada uma. Posteriormente, outras contribuições vieram da BEA, SAP e Siebel e, em Maio de 2003, foi publicada a versão 1.1. A especificação do BPEL4WS (atualmente conhecida por WS-BPEL) foi submetida para um comitê técnico da OASIS para que pudesse ser desenvolvida oficialmente como um padrão aberto (ALVES *et al.*, 2007).

O WS-BPEL possui total aderência à linguagem WSDL, permitindo um modelo de interação para suportar transações distribuídas utilizando Serviços Web. Suas capacidades para trabalhar com processos habilitam declarações sobre relacionamentos com parceiros, processamentos de dados, manipulações de regras de negócio e atividades a serem cumpridas.

Podem ser definidos dois tipos de processos: abstratos e executáveis. Os processos abstratos descrevem o comportamento do processo sem cobrir os detalhes de execução. São utilizados para fornecer ao parceiro externo detalhes das interações possíveis com o processo, sem expor a sua lógica de negócio interna. Processos executáveis definem tanto o comportamento do processo de negócio quanto as sua lógica interna, sendo esse o processo efetivamente executado.

A linguagem WS-BPEL permite construções distintas (*sequence, switch, while, pick, if-then-else*) e a utilização de regras dinâmicas para representar processos decisórios e, assim, atribuir comportamentos diferenciados de acordo com os resultados obtidos na execução dos Serviços Web. A invocação de Serviços Web pode ocorrer de forma serializada, quando existe uma dependência, ou paralelizada, quando não existe uma ordem de precedência entre os serviços. Existe suporte para tratar exceções, que eventualmente ocorram na chamada dos Serviços Web, desfazimento (ou compensação) de atividades para manter a integridade do processo executado e, estruturas de repetição, que podem ser utilizadas, por exemplo, para tratar novas tentativas para a conclusão de uma atividade.

Processos modelados em WS-BPEL podem ser construídos para não manter estados, e, nesse caso, são ditos *stateless*, sendo geralmente utilizados para modelar processos de curta duração onde não há interações com parceiros externos.

Processos que exigem interações, existindo troca de mensagens com parceiros externos, geralmente necessitam manter o seu estado para garantir a consistência transacional e, assim, são ditos processos de longa duração ou *statefull*. O processo de longa duração, ao enviar mensagens para um parceiro externo, armazena todas as suas variáveis em meio magnético, garantindo assim que o seu estado corrente será mantido até o parceiro externo retornar a resposta do diálogo. Para esse tipo de

processo, o WS-BPEL possui a capacidade de correlacionar mensagens com os seus respectivos processos em andamento e manter a sua integridade, função fundamental para esse tipo de fluxo (JENNINGS *et al.*, 2000).

Originalmente, o WS-BPEL foi concebido para a descrição de orquestrações estáticas, onde um desenvolvedor previamente constrói os processos executáveis e, em seguida, os disponibiliza em ambiente de execução. O trabalho (COURBIS *et al.*, 2004) propõe extensões ao WS-BPEL para permitir algum nível de dinamismo. Para a criação de orquestrações dinâmicas, são necessárias linguagens com alto poder de expressividade, como é o caso do OWL-S.

De forma equivalente ao WS-BPEL, o *process model* no OWL-S *ServiceModel*, possui a capacidade de descrever processos utilizando controles capazes de representar diferentes comportamentos de uma orquestração, mas associando as características semânticas do OWL, para fornecer descrições mais ricas dos Serviços Web, incluindo pré-condições e efeitos. Isso permite a criação de orquestrações dinâmicas e a seleção automática de Serviços Web.

2.4.2. Coreografia

Coreografias descrevem as interações e formas de conversação possíveis entre dois ou mais Serviços Web a partir de uma perspectiva global ou local (LONG *et al.*, 1995). O modelo global de coreografia especifica as trocas de mensagens de um ponto de vista geral (entre todos os participantes). O modelo local de coreografia especifica interações possíveis na perspectiva de um participante (Serviço Web) e também referencia as interfaces dos processos envolvidos. Assim, a coreografia global corresponde a múltiplas coreografias locais (para cada participante).

Diferentemente da orquestração, a coreografia não trata de um ponto central de controle para a execução de Serviços Web. O seu papel está em descrever o comportamento que um Serviço Web deverá possuir dentro de um domínio, descrevendo os estados que poderão ser alcançados e a seqüência de interações relativas a cada estado com outros Serviços Web.

A vantagem principal da coreografia é definir um comportamento geral de um domínio de Serviços Web, sem a influência de um processo centralizado, que pode gerar distorções quanto ao funcionamento do todo. Assim, é possível estabelecer que um Serviço Web será utilizado dentro de uma seqüência lógica que faça sentido do ponto de vista do domínio, e não da interpretação de utilização de um processo particular que o está invocando.

O W3C publicou o padrão WS-CDL (*Web Services Choreography Description Language*) (ROSS-TALBOT *et al.*, 2006), que propõe uma especificação para a descrição de contratos para a colaboração entre participantes de qualquer tipo, independente da plataforma ou modelo de programação utilizados para a implementação do provedor. O contrato possui uma definição global das condições e restrições para a troca de mensagens, que descreve o comportamento de todos os participantes envolvidos. Cada participante pode utilizar a definição global para construir e testar soluções que estejam em conformidade. Assim, um participante pode interagir com outro participante através de uma ou mais mensagens ordenadas. As interações e os papeis exercidos pelos participantes são descritos no contrato, assim como a forma de se trocar mensagens e o tratamento de exceções. Quando estas regras não são seguidas, pode ser visto como um erro de não conformidade com a descrição do WS-CDL.

Nota-se que o WS-CDL não se propõe a estabelecer formas de se executar uma composição. Coreografias norteiam a criação de uma composição, devendo haver mecanismos que efetuem a tradução para um modelo de execução escolhido. ROSS-TALBOT *et al*. (2006) propõem uma forma de se efetuar orquestrações em WS-BPEL tomando como base coreografias descritas em WS-CDL.

2.4.3. Composição Manual

Na modelagem de fluxos de composição manual (ou estática), utiliza-se geralmente o padrão WS-BPEL, por ser largamente difundido por diferentes fornecedores de software. Em SRIVASTAVA *et al*. (2003) é descrito em detalhes um cenário de composição de Serviços Web para uma agência de viagens utilizando o

WS-BPEL. A composição da agência de viagens é invocada quando o cliente deseja efetuar uma reserva e, durante a sua execução, são invocados os Serviços Web dos parceiros para efetuar a reserva do hotel e do vôo.

O cenário da agência de viagens apresenta as diversas questões que devem ser tratadas por composições de Serviços Web em problemas do mundo real, tais como, paralelismo nas atividades de reserva, cancelamentos de reservas, problemas de comunicação com os parceiros e falta de recursos disponíveis. Durante a etapa de desenvolvimento de uma composição, o desenvolvedor deverá analisar as situações possíveis que possam ocorrer durante a execução da composição e construir fluxos capazes de lidar com tais eventualidades.

2.4.4. Composição Assistida

A Composição assistida ou semi-automática é muito similar à composição manual no aspecto de construção da lógica de invocação dos Serviços Web. Porém, fornece ao desenvolvedor certas facilidades como, por exemplo, a indicação de quais Serviços Web utilizar para realizar uma determinada atividade. Em cenários com diversos Serviços Web similares disponíveis, a correta identificação de qual Serviço Web utilizar pode não ser uma atividade trivial (SIRIN *et al.*, 2004).

Em SRIVASTAVA *et al*. (2003), o exemplo da agência de viagens é explorado através de um processo semi-automático de composição para facilitar a identificação de Serviços Web para o usuário em cada passo da composição, usando as descrições semânticas dos Serviços Web em DAML-S (versão anterior do OWL-S) com as descrições de invocação do WSDL. Porém, prevê a necessidade de incorporação de técnicas de planejamento (GHALLAB *et al.*, 2004) para facilitar a automação na atividade de inferência do sistema, necessitando de menos apoio do usuário na construção de composições.

Em (SHENG *et al.*, 2002) é descrita uma plataforma chamada SELF-SERV, que permite que um desenvolvedor descreva uma composição através de um diagrama de estado, utilizando um módulo gráfico fornecido pela plataforma. Em tempo de execução, um usuário poderá trocar Serviços Web que são invocados pela

orquestração para corrigir ou aprimorar as atividades executadas. A plataforma possui a capacidade de recuperar intervenções realizadas anteriormente e levá-las em consideração.

O abordagem descrita em (CHEN *et al.*, 2003) guia o desenvolvimento de composições, descoberta e seleção de Serviços Web Semânticos a partir de um domínio de conhecimento. O domínio de conhecimento é modelado utilizando ontologias e permite o acoplamento de mecanismos de geração automática de composições web.

2.4.5. Composição Automática

A geração automática de composições traz um benefício direto que é a diminuição de mão-de-obra para se construir composições de Serviços Web. Além disso, se o processo de geração da composição ocorrer em tempo real, os Serviços Web utilizados tendem a ser os mais atualizados, aumentando teoricamente as chances de se alcançarem os objetivos de forma satisfatória. Entretanto, um processo de geração automática deve tratar uma série de desafios. Entre eles, podemos citar:

- Busca por Serviços Web para solucionar um problema;
- Interpretação dos Serviços Web para identificar a forma de utilização;
- Questões de QoS dos Serviços Web;
- Como sequenciar os Serviços Web de forma adequada para alcançar um objetivo;
- Como expressar um objetivo e determinar o que foi alcançado;
- Como capturar o estado de um Serviço Web executado; e
- Como descrever o fluxo de dados e controles para os Serviços Web.

Existem muitas técnicas e frameworks propostos para a geração automática de composições de Serviços Web (AGARWAL *et al.*, 2008). Dentre as diversas propostas, podem-se verificar basicamente duas linhas de pensamento (MENDLING,

2005): Algumas ignoram a grande aceitação das linguagens estáticas de composição e descrevem uma metodologia própria para composição automática utilizando os padrões emergentes, mais alinhados com as especificações da Web Semântica. Outras propostas buscam uma aproximação com linguagens que já estão sendo largamente utilizadas para composições manuais e que derivam de conceitos da disciplina de Gestão de Processos de Negócio. Essas propostas procuram, por exemplo, adaptar as linguagens para gerar fluxos dinâmicos, mas evitando ruptura com os padrões já estabelecidos.

No primeiro caso, a grande maioria dos trabalhos parte da premissa de que existirão Serviços Web Semânticos padronizados e disponíveis, focando basicamente seu esforço na atividade de confecção do melhor fluxo para atingir um objetivo informado. Procura-se então explorar os melhores algoritmos para resolver os problemas específicos do domínio do problema abordado.

No segundo caso, os trabalhos reforçam a necessidade de utilização dos Serviços Web existentes e dos padrões já estabelecidos. Procura-se criar soluções que aproveitem ao máximo os artefatos disponíveis, simulando basicamente o processo de criação de composições estáticas. Para tanto, existe a necessidade de criar mecanismos de tradução entre os serviços publicados e algum tipo de representação semântica que permita a geração de composições automaticamente.

A existência dessa divisão de abordagens decorre do fato de que alguns pesquisadores acreditam ser mais viável criar composições automáticas alinhadas com técnicas da Web Semântica, partindo do pressuposto de que a sua infraestrutura de Serviços Web estará disponível para a criação de composições. Outros, no entanto, acreditam ser necessário criar mecanismos para que a máquina possa compor, porém utilizando os padrões já estabelecidos. Aparentemente essa discussão é mais abrangente, sendo mais relacionada à própria evolução da Web Semântica em si do que à atividade de composição automática. Ela é motivada essencialmente pelo fato de que, até o momento, são raras as utilizações dos padrões da Web Semântica, pelo menos no âmbito comercial.

Abstraindo diferenças entre notações específicas de cada cenário, chega-se aos problemas comuns das duas linhas de pensamento relacionadas a composições automáticas: como fazer com que a máquina selecione serviços necessários para a composição e a confecção de um fluxo lógico que gere o resultado esperado (orquestração).

Para a construção dos fluxos de composições automáticas de serviços geralmente são utilizadas técnicas de planejamento de IA (BARRETO *et al.*, 2007). Existem muitas técnicas já descritas, cada qual com seus pontos positivos e negativos.

Uma vez que Serviços Web podem ser representados como ações, é possível tratar a composição de Serviços Web como um problema de planejamento (ALVES *et al.*, 2007). A partir de um objetivo de usuário fornecido, o planejador irá encontrar uma coleção de Serviços Web que alcancem o objetivo. Para tanto, é necessário entender a estruturação de um Serviço Web e suas limitações.

Os Serviços Web utilizam o WSDL, que fornece um conjunto de operações que poderão ser invocadas por uma aplicação. Uma operação de um Serviço Web é descrita através de um nome e parâmetros de entrada e saída (IO) que permitem que ela seja invocada. Não há, no entanto, informações suficientes para que um agente de software possa determinar a sequência de operações para alcançar um objetivo do usuário.

Linguagens Web Semânticas, como o OWL-S, permitem representar processos atômicos (o equivalente de operações WSDL) em termos de entradas, saídas, pré-condições e efeitos (IOPEs), estendendo as descrições WSDL. No que diz respeito à composição automática de Serviços Web, podemos descrever basicamente três tipos de tarefas que podem ser automatizadas com a utilização do OWL-S (WU *et al.*, 2003): descoberta automática de Serviços Web, invocação automática de Serviços Web e composição de Serviços Web.

A descoberta automática de Serviços Web permite a localização de Serviços Web com base em um conjunto específico de informações e restrições fornecidas acerca dos serviços. A localização dos serviços ocorre então por meio de registros ou ferramentas de busca baseadas em ontologias.

Para a invocação, a composição deve identificar os atributos que são utilizados pelo Serviço Web e efetuar o mapeamento dos dados que possui na composição com esses atributos. Como os atributos podem ter nomes diferentes para o mesmo significado, a composição deve possuir capacidades semânticas para invocar os serviços corretamente.

Para a composição automática, podem ser utilizadas muitas técnicas, sendo o planejamento uma das mais empregadas.

2.4.6. Frameworks para Composição Automática

O projeto de pesquisa ASTRO (MARCONI *et al.*, 2008) possibilita a geração automática de composições em WS-BPEL a partir de um objetivo informado pelo usuário. Em caso de falhas, utiliza uma composição gerada a partir de um objetivo secundário para desfazer o que havia sido feito até o momento da falha. Fornece uma plataforma completa para desenvolvimento de funcionalidades para composição automática de Serviços Web, apoiando o analista desde a captação de requisitos de usuários, até a verificação da composição gerada com relação aos requisitos. Disponibiliza ainda um ambiente de monitoração durante a execução da composição.

O CASCOM (KLUSCH *et al.*, 2005) é uma plataforma que abrange diversos aspectos relacionados à mobilidade. O seu objetivo é prover uma solução de agentes inteligentes para dispositivos móveis que possam lidar com atividades levando em consideração as condições correntes encontradas no ambiente, adaptando-se a elas. A sua implementação foi realizada a partir do framework para agentes inteligentes JADE/LEAP, que é baseada no padrão FIPA e prevê quatro componentes principais (BERGENTI *et al.*, 2006): a camada de rede, a camada de coordenação de serviços, o Subsistema de Contexto e o Subsistema de Segurança. A camada de rede possui capacidades para lidar com os diferentes tipos de rede provendo eficiência, estabilidade e garantia. Provê um diretório, centralizado ou distribuído, de Serviços Web Semânticos, baseados em OWL-S. A camada de coordenação de serviços fornece capacidade de descobrimento, composição e execução de Serviços Web Semânticos. A composição gerada irá orquestrar um ou mais Serviços Web para

alcançar um objetivo. O Subsistema de Contexto é encarregado de adquirir, armazenar e fornecer informações sobre contexto para as demais camadas. O Subsistema de Segurança e Privacidade é responsável por garantir a segurança e a privacidade das informações para os componentes da solução.

WU *et al.* (2007) propõem uma solução para tratar problemas de composição automática de Serviços Web levando em consideração as complexidades de se manipularem processos e dados heterogêneos de diferentes Serviços Web. Partem do princípio de que os Serviços Web estarão com anotações semânticas feitas com SAWSDL (FARRELL *et al.*, 2007), um padrão para anotação recomendado ao W3C, que é baseado em WSDL-S (AKKIRAJU *et al.*, 2005). A composição gerada não especifica detalhes de como invocar um Serviço Web, delega as atividades para mediadores de dados que devem decidir qual é o Serviço Web mais apropriado para efetuar as atividades e formatar adequadamente a mensagem de entrada para a sua invocação.

Para a geração, propriamente dita, de composições automáticas de Serviços Web, os frameworks apresentados utilizam técnicas de planejamento.

2.5. Planejamento

Planejamento é uma técnica de Inteligência Artificial (IA) responsável pela geração de um plano, que é uma combinação de ações para resolver um problema específico (GHALLAB *et al.*, 2004). Um problema é caracterizado por um estado inicial e um objetivo a ser alcançado. O estado inicial informa ao Planejador como o mundo está naquele momento, enquanto que o objetivo descreve como o mundo deve se parecer quando o plano for executado (TATE *et al.*, 1990). Em planejamento, o "mundo" refere-se ao conjunto de estados possíveis do domínio da aplicação em cima do qual o planejamento está sendo realizado.

Um plano é um conjunto de ações ordenadas em conformidade com suas pré-condições e efeitos. As pré-condições descrevem um conjunto de estados requeridos para que a ação possa ser executada. Efeitos descrevem os estados alcançados após a

execução da ação. O planejador leva em consideração as pré-condições e efeitos das ações disponíveis para ordená-las e gerar o plano.

Ações são chamadas determinísticas quando apenas um estado pode ser alcançado após a sua execução. Se todas as ações são determinísticas, o planejador usualmente gera planos que se restringe a sequências de ações. Quando uma ação permite que mais de um estado seja alcançado, não sendo possível saber antes da sua execução o estado que será alcançado, a ação é dita não-determinística e o plano precisa ser representado por estruturas de dados mais complexas, capazes de indicar a próxima ação a ser executada dependendo do estado que é atingido.

Existem diversos métodos de planejamento disponíveis na literatura. A adequação de cada método depende das propriedades do domínio e do problema a ser resolvido. Para a composição automática de Serviços Web, os métodos mais comumente adotados nas diferentes propostas são baseados em redes hierárquicas de tarefas (*Hierarchical Task Networks* - HTNs) (NAU *et al.*, 2003) ou em verificação de modelos (*Model Checking*) (BERTOLI *et al.*, 2003).

No planejamento baseado em HTNs, criam-se planos pela decomposição de tarefas. Os objetivos correspondem a um conjunto de tarefas a serem executadas. As tarefas podem ser abstratas ou primitivas. No primeiro caso, elas estão associadas a diversos métodos alternativos para a sua decomposição em sub-tarefas. No segundo caso, elas estão associadas a ações que podem ser diretamente executadas no domínio. O planejador decompõe tarefas abstratas em tarefas menores até alcançar tarefas primitivas que possam ser realizadas diretamente. Esse processo assemelha-se muito com a decomposição de processos de negócio que norteiam o desenvolvimento de composições manuais de Serviços Web (WU *et al.*, 2003). Esse tipo de planejamento tende a ser eficiente por reduzir a busca de alternativas de planos a combinações de métodos previamente definidos, de acordo com conhecimento específico do domínio.

O planejamento baseado em verificação de modelos é uma técnica utilizada em domínios não-determinísticos. A partir da indicação dos estados que podem ser atingidos por cada ação quando aplicada a cada estado, executam-se métodos de verificação automática de modelos (BERTOLI *et al.*, 2003) para gerar os planos. Os

planos, nesse caso, são regras de inferência que indicam a ação a ser executada em cada estado de modo a se atingir o objetivo. A abordagem é flexível para tratar inclusive objetivos estendidos, que especificam propriedades do caminho de execução e não apenas dos estados finais. O tratamento do não-determinismo torna a abordagem particularmente atraente para a composição de Serviços Web, mas implica em complexidade alta para o planejamento em tempo real.

2.5.1. Planejamento e Composição Automática

Muitos trabalhos utilizam algoritmos de planejamento para a geração de composições de Serviços Web. Em composições automáticas de Serviços Web com planejamento, Serviços Web são mapeados em ações. O planejador, durante a atividade de planejamento, leva em consideração as ações disponíveis para a geração do plano, que em seguida é convertido para uma linguagem de execução apropriada, mapeando novamente as ações em Serviços Web que serão utilizados.

O projeto de pesquisa ASTRO utiliza o algoritmo MBP (BERTOLI *et al.*, 2003), que é baseado na técnica de planejamento não-determinístico com verificação de modelos. O ASTRO aborda diversos aspectos relativos à Engenharia de Software que facilitam o mapeamento dos requisitos levantados sobre o domínio em grafos de estados associados a cada Serviço Web. Consegue compor serviços levando em conta o não-determinismo e as diversas possibilidades de interação entre serviços. No entanto, o ASTRO é voltado para a geração de composições sofisticadas a serem planejadas uma única vez para serem executadas muitas vezes. A complexidade do tipo de planejamento adotado tende a tornar o planejamento em tempo real inviável. Dessa forma, o ASTRO não prevê o planejamento em tempo real, seja para atingir novos objetivos ou para tratar situações que não tenham sido previamente levadas em consideração.

Em (SIRIN *et al.*, 2004) é proposta uma solução com a utilização do planejador SHOP2 (NAU *et al.*, 2003), com Serviços Web descritos em OWL-S. A solução provê um algoritmo que traduz Serviços Web Semânticos para a linguagem de especificação adotada pelo SHOP2 e depois gera o plano. O uso do SHOP2 para a

geração do plano possibilita que as tarefas estejam na mesma ordem que serão executadas posteriormente. Isso permite a identificação correta do estado corrente do mundo em cada etapa do processo de planejamento, aperfeiçoando processos de inferências e raciocínio e a habilidade de se chamar programas externos. A abordagem restrita ao uso de um planejador determinístico como o SHOP2, no entanto, tem aplicação limitada em domínios intrinsecamente não-determinísticos como aqueles em que Serviços Web são executados.

O OWLS-Xplan (KLUSCH *et al.*, 2005) é uma solução para composição automática de Serviços Web Semânticos descritos em OWL-S. Ela propõe o uso de um algoritmo híbrido desenvolvido especificamente para a solução, que combina a eficiência do planejador *FastFoward* (HOFFMANN *et al.*, 2001) baseado em *graph-plan* e o planejamento HTN. Diferentemente do HTN, XPLAN sempre encontra uma solução se existem ações/estados que combinados levam a um plano possível, característica herdada de planejadores baseados em ação. O HTN se concentra na busca de planos formados por soluções para subproblemas expressas sob a forma de métodos, o que não garante a obtenção de uma solução se ela for muito diferente do que foi previamente modelado. Por outro lado, a existência de uma rede de tarefas que alcance um objetivo traz ganhos consideráveis de eficiência, motivo pelo qual o XPLAN combina as duas abordagens. O CASCOM, na camada de coordenação de Serviços Web, utiliza o OWLS-Xplan para as atividades de geração automáticas de composições. O OWLS-Xplan é um planejador determinístico, mas trabalha com a idéia de replanejamento automático quando ocorrem falhas na execução do plano. Tal abordagem já dá alguma flexibilidade para tratar resultados diferentes do esperado quando serviços são executados. No entanto, isso difere de um planejamento não-determinístico que leve em conta as possibilidades de resultado de cada ação para aumentar as chances de alcance dos objetivos.

Serviços Web são intrinsecamente não-determinísticos, pois não é possível determinar o estado que será alcançado por um Serviço Web após a sua conclusão. Mesmo o Serviço Web mais simples, que realiza uma simples ação de consulta, poderá retornar um resultado inesperado, por exemplo, o Serviço Web pode estar indisponível. Dessa forma, o planejador deverá idealmente levar em consideração os

possíveis resultados de cada ação para definir as próximas ações. Para tal, é necessária a utilização de planejadores não-determinísticos. O não-determinismo dos serviços e o surgimento de novas situações, em decorrência da interação em paralelo de um sistema com o usuário ou com outros sistemas, demandam também um mecanismo de monitoração que permita a identificação da necessidade de se planejar composições de serviços para atingirem novos objetivos. Diante disso, a pesquisa desenvolvida neste trabalho investigou uma arquitetura que possibilitasse tratar tanto o não-determinismo quanto a continuidade no processo de composição e execução de Serviços Web.

Inicialmente foi verificada a possibilidade de estender soluções existentes para construir a arquitetura desejada. A primeira alternativa considerada foi a extensão do ASTRO para permitir a capacidade de controle de uma orquestração, prevendo capacidades de replanejamento. Contudo, alguns componentes que deveriam sofrer alterações não possuíam códigos fontes disponíveis ou foram descontinuados pela equipe do projeto, não permitindo assim sua extensão. Além disso, verificou-se o problema de eficiência para o planejamento baseado em verificação de modelos em tempo de execução.

Outra possível solução poderia ser estender o CASCOM no sentido de tratar melhor o não-determinismo e a continuidade do processo de composição e execução. Contudo, ao se tentar montar o ambiente, várias dificuldades foram encontradas, tais como manuais desatualizados, códigos com problemas de compilação etc. Além disso, a extensão já seria difícil pelo fato do planejador OWLS-Xplan ser determinístico.

Diante das dificuldades encontradas com as soluções existentes disponíveis, e observando-se a disponibilidade de um planejador não-determinístico que poderia ser adaptado à composição automática de Serviços Web, decidiu-se por investir na solução descrita nesta dissertação. Foi utilizado um planejador não-determinístico baseado em HTN, que foi desenvolvido inicialmente para atender à *Geração de Enredos em Storytelling para TV Interativa*. A utilização do planejador, conhecido por ND-HTN (SILVA, 2010), foi de grande importância para a solução proposta, uma vez que todo o código fonte estava disponível em Prolog, o que permitiu ajustes

necessários para o seu uso com Serviços Web e um bom nível de integração. Focou-se a pesquisa na composição de serviços não-determinísticos em tempo real, levando-se em conta o controle da execução das composições e a monitoração para a verificação da necessidade de gerar e executar novas composições. Questões relativas a pontos importantes da composição de serviços (tais como a descrição semântica dos serviços e a descoberta de serviços) foram abstraídas para permitir o tratamento do problema dentro do escopo de uma dissertação de mestrado.

3. Uma Arquitetura para Composição e Monitoração Automáticas e Contínuas de Serviços Web Não-Determinísticos

O capítulo 2 fez um apanhado geral sobre os assuntos relacionados a composições de Serviços Web. Foram apresentados os principais conceitos, tecnologias, desafios e a forma como o tema vem sendo abordado. Áreas de conhecimento como a Inteligência Artificial e Web Semântica foram mencionadas, pois tiveram influência direta nas soluções que estão sendo empregadas em Composições de Serviços Web. Para finalizar, foram apresentadas as principais soluções propostas por outros autores, descrevendo algumas diferenças para a solução proposta.

A partir deste capítulo, será apresentada a solução proposta, a qual permite a geração automática e o controle da execução de composições. A solução está focada no tratamento não-determinístico dos serviços e na criação de um processo contínuo, onde ocorre uma monitoração dos estados para identificar a necessidade de geração e execução de novas composições.

3.1. Modelagem Conceitual

A solução parte do pressuposto que determinados aspectos do mundo ou ambiente real serão representados internamente permitindo que a solução possa interagir corretamente. A interação ocorre através do recebimento de eventos externos, que são traduzidos e interpretados em ações que ao serem executadas no ambiente real produzem efeitos que determinarão qual será a próxima ação a ser executada. A figura 3-1 ilustra a modelagem conceitual da solução.

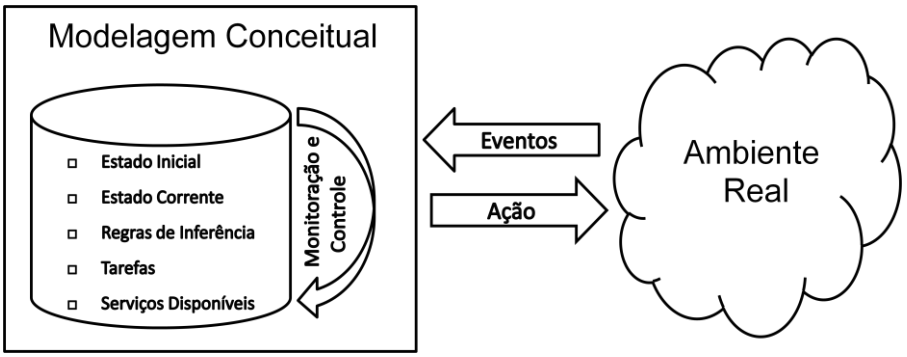

Figura 3-1- Modelagem Conceitual do Solução

A figura 3-1 demonstra o recebimento de eventos do ambiente real para um determinado cenário mapeado e ações que são exercidas sobre o ambiente. Para tanto, a solução deve conhecer previamente o cenário que estará interagindo, usando um conjunto de artefatos armazenados e que façam uma representação do ambiente.

O estado inicial descreve para a solução um conjunto de características válidas do ambiente real quando a solução é iniciada. Conforme a solução é executada, novas informações são acrescidas ou retiradas, permitindo a solução conhecer e acompanhar a evolução do estado corrente no ambiente real. As informações podem descrever objetos, pessoas, tipos de relação ou alguma característica que seja observada no ambiente real. Assim, por exemplo, um carro pode ser descrito através do estado inicial: modelo Uno, localização Rio de Janeiro e sem movimento. Ao se movimentar, o estado inicial deve ser atualizado para descrever a velocidade corrente do veículo e a sua direção atual, sendo esse o novo estado corrente.

As regras de inferência descrevem quando uma tarefa deve ser acionada a partir de um estado específico. Por exemplo, supondo que o veículo ao se movimentar chegue a uma cidade onde existem radares presentes, uma regra de inferência poderia acionar uma tarefa de verificação de velocidade limite e notificar o condutor. As tarefas são previamente descritas e permitem a execução de um conjunto de ações no ambiente, sendo essas ações materializadas com a invocação de Serviços Web correspondentes. Quando uma ação é executada sobre o ambiente o estado corrente é atualizado com os efeitos correspondentes gerados no ambiente real. Dessa forma,

seria possível armazenar a informação de que o condutor já teria sido notificado quanto à existência de radares naquela cidade, não sendo notificado novamente.

Eventos externos podem indicar mudanças no estado corrente permitindo que a funcionalidade de controle verifique as regras de inferência existentes para a realização de novas tarefas. As tarefas são configuradas previamente e descrevem possibilidades de comportamentos distintos através da combinação de outras tarefas. Por exemplo, suponha a necessidade de movimentação financeira entre duas contas, pode ser criada uma tarefa que descreve como a transferência deve ser realizada, havendo como pré-condição, a existência das duas contas para que a transferência seja realizada. Entretanto pode haver diferentes tipos de transferência, para o mesmo banco ou bancos distintos. Assim, uma tarefa mais genérica de transferência pode estar relacionada com duas outras tarefas de transferência mais específicas, uma para transferência para o mesmo banco e outra para transferência entre bancos distintos. Por sua vez, a transferência para o mesmo banco pode estar relacionado com uma sequência de tarefas mais específicas, onde existe uma tarefa de realizar o débito do dinheiro de uma conta e outra tarefa para efetuar o crédito na conta de destino.

As tarefas são organizadas de forma hierárquica e permitem que sejam criadas diferentes possibilidades de solução para um objetivo específico, onde cada ramificação gerada é uma sequência de ações que levam em consideração as informações conhecidas para aumentar as chances de se alcançar o objetivo e eliminar ramificações que são consideradas inadequadas.

3.2. Arquitetura Geral

A solução proposta, chamada SACCAS (Solução para Automação e Controle de Composições Automáticas de Serviços Web não-determinísticos), utiliza técnicas de planejamento não-determinístico combinadas com planejamento baseado em HTNs.

A figura 3-2 apresenta um diagrama que exemplifica a organização dos módulos da solução e a dependência que os módulos possuem entre si.

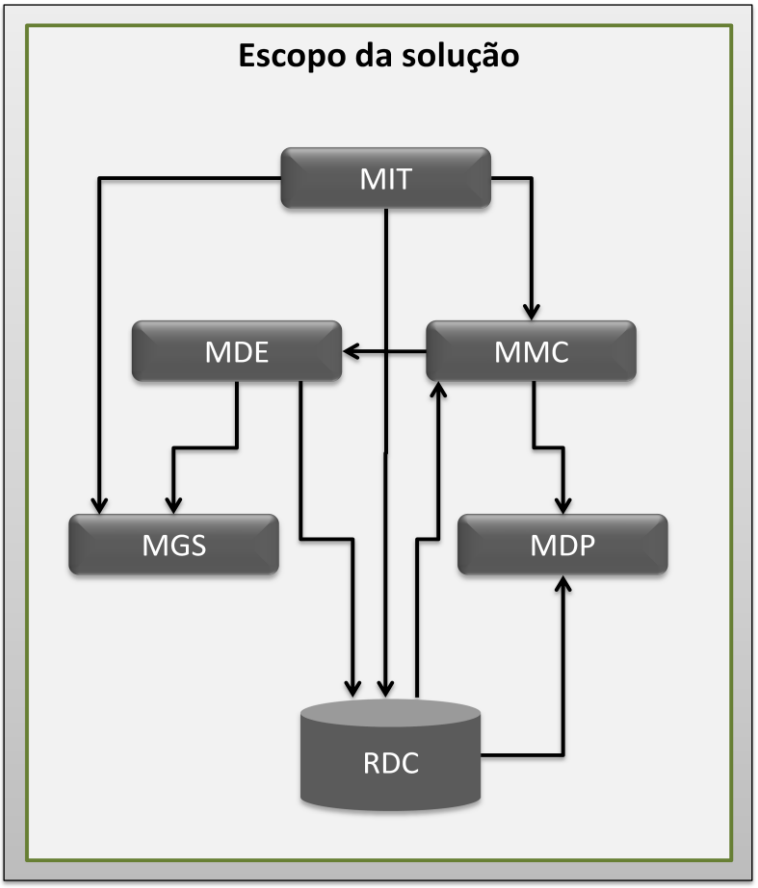

Figura 3-2- Arquitetura SACCAS

A Arquitetura do SACCAS (Figura 3-2) é dividida em cinco módulos independentes e um repositório, que necessitam interagir entre si para o correto funcionamento da solução. Cada módulo possui o seu papel bem definido conforme as descrições a seguir:

- Módulo de Planejamento (MDP) – É responsável por gerar o plano para realizar uma tarefa;
- Módulo de Execução (MDE) – Efetua a execução da composição de Serviços Web a partir de um plano previamente gerado pelo MDP e enviado pelo MMC;
- Módulo de Monitoração e Controle (MMC) – Permanece verificando os estados correntes do ambiente e aciona regras de inferência quando

necessário, invocando os módulos de planejamento e execução quando necessário;

- Repositório de Conhecimento (RDC) – Armazena informações gerais sobre o estado do ambiente, regras de inferência, tarefas e serviços disponíveis;
- Módulo de Gestão de Serviços (MGS) – Efetua o mapeamento de serviços abstratos nos serviços concretos, realiza a invocação dos serviços e realiza a comunicação com o MDE;
- Módulo de Interação com Terceiros (MIT) – Responsável pelas interfaces gráficas do ambiente para facilitar o uso por usuários finais.

Pelo fato do SACCAS ser uma solução de natureza extremamente dinâmica, todos os módulos são utilizados em tempo de execução, característica fundamental para se adaptarem às diferentes situações encontradas em razão do não-determinismo.

Muitas soluções necessitam gerar boa parte dos seus artefatos em tempo de desenvolvimento, mesmo quando os processos são automatizados. Por exemplo, o ASTRO possui o processo automático de geração de composições, entretanto é realizado em ambiente de desenvolvimento. Quando a composição é finalizada, é passada para o ambiente de produção onde é permanentemente reutilizada.

O SACCAS, uma vez configurado, permanece em execução constante, realizando as atividades previstas. O processo de execução possui basicamente três sub-processos bem definidos: Monitoração e Controle de Composições, Geração de Composições e Execução de Composições.

O restante do capítulo foi organizado de forma a explorar o objetivo de cada sub-processo, suas funcionalidades e relacionamentos com os módulos da solução. O capítulo 4 detalhará o funcionamento de cada módulo.

3.3. Geração de Composições

Um planejador determinístico possui a capacidade de determinar, com precisão, qual será o efeito gerado após a execução de uma ação. Entretanto, em ambientes de Serviços Web, não é possível assumir que após a invocação de um Serviço Web um conjunto específico de efeitos será alcançado.

O Serviço Web invocado pode estar momentaneamente indisponível ou gerar diferentes efeitos após a sua conclusão. Uma composição deve possuir a capacidade de tratar os diferentes efeitos causados e determinar o próximo Serviço Web a ser invocado.

Um planejador não-determinístico possui a capacidade de gerar planos com diferentes possibilidades de caminhos, levando em consideração efeitos causados pela execução de uma ação, sendo assim, mais adequado para a geração de composições de Serviços Web.

Algoritmos de planejamento não-determinístico necessitam de formas eficientes de analisar todas as possibilidades retornadas por uma ação e gerar planos que possuem comportamento condicional e incorporem estratégias de tentativa e erro (GHALLAB *et al.*, 2004).

Para que o planejamento possa ser aplicável em composições de Serviços Web, estes devem ser previamente mapeados em ações descrevendo suas pré-condições e efeitos. Tipicamente, um Serviço Web possui apenas parâmetros de entrada e parâmetros de retorno. Numa composição manual, um ser humano é responsável por descrever o sequenciamento lógico de invocação dos Serviços Web, identificando as informações necessárias que serão passadas e as informações retornadas por cada Serviço Web. As pré-condições e efeitos de um Serviço Web estão implícitos e o ser humano deve inferir qual será o comportamento do Serviço Web dentro do contexto da sua solução. Para a geração de composições automáticas com planejamento, a máquina deve realizar esse trabalho de inferência, sendo necessário o mapeamento prévio do Serviço Web em ações, contendo além de parâmetros de entrada e saída, suas pré-condições e efeitos explicitados. Na solução do SACCAS, operadores possuem a finalidade de representar o comportamento dos Serviços Web e são utilizados pelo Módulo de Planejamento para a geração do plano.

O planejador, a partir de um objetivo informado sob a forma de uma tarefa abstrata a ser executada, gera um plano que tem a forma de uma árvore, onde os nós correspondem a ações (operações dos Serviços Web a serem invocadas ou ações internas de controle do fluxo) e as arestas correspondem a condições a serem testadas para definir a próxima a ação a ser executada. Tal árvore é criada com base nas pré-condições e efeitos de cada ação, de modo a se garantir o cumprimento da tarefa em todos os caminhos possíveis. O plano gerado no SACCAS representa a própria composição que será utilizada pelo Módulo de Execução. O código 3-1 exemplifica o trecho de um plano gerado:

```
 -> consultar_voos(1,gilberto,campinas,vitoria,28/09/2010,_G1879,_G1880)
 [?][alternativa_trajeto(1,gilberto,_G5696,_G5697)]
   -> consultar_hoteis(1,gilberto,vitoria,28/09/2010,_G5709,_G5710)
   [?][alternativa_hotel(1,gilberto,vitoria,28/09/2010,_G5725,_G5726)]
     -> consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G5755,_G5756)
     [?][alternativa_trajeto(1,alberto,_G5769,_G5770)]
       -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5782,_G5783)
       [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5798,_G5799)]
       [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5845,_G5846)]
     [?][alternativa_trajeto_inexistente(1,alberto,_G5890,_G5891)]
       -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5903,_G5904)
       [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5919,_G5920)]
       [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5966,_G5967)]
(...)
```

Código 3-1 Trecho simplificado de um plano gerado

O símbolo -> é utilizado para indicar uma ação e [?] para indicar as alternativas de uma ação não-determinística, como pode ser observado na ação *consultar_voos*. A figura 3-3 exibe graficamente o trecho simplificado do plano gerado.

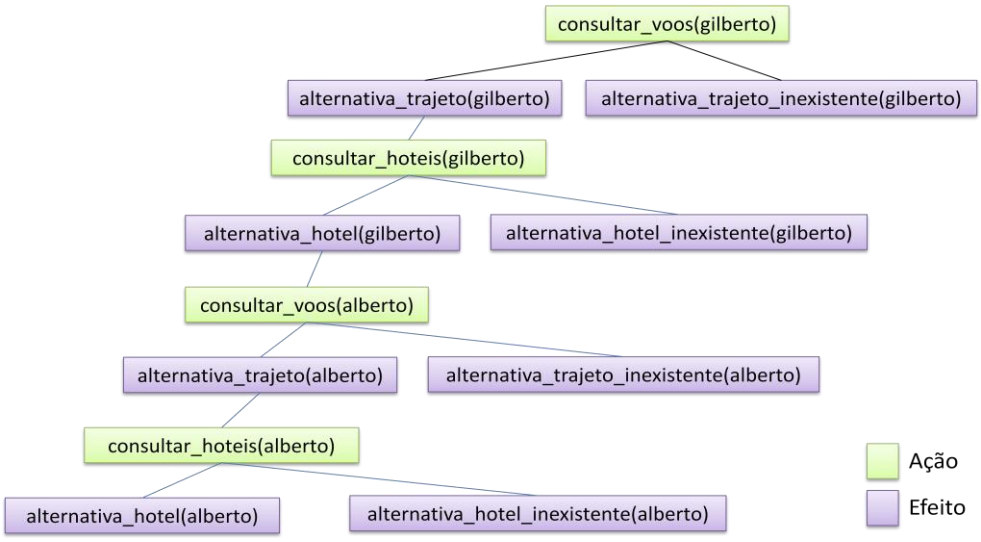

Figura 3-3 Representação gráfica do plano simplificado

A figura 3-3 exemplifica as ramificações existentes geradas pelo plano que são alternativas de caminhos possíveis para se alcançar um objetivo.

3.4. Execução de Composições

Existem muitas linguagens que permitem definições para a execução de composições de Serviços Web, tais como: WS-BPEL, OWL-S e WSMO. Entretanto, em ambientes de geração automática de composições de Serviços Web, existem ainda incertezas sobre qual é o padrão mais adequado a ser utilizado.

O WS-BPEL vem demonstrando limitações, onde se destacam a falta de mecanismos semânticos e flexibilidade para a geração de fluxos. Outras linguagens que surgiram para suprir as necessidades de composições automáticas de Serviços Web, como é o caso do OWL-S, ainda se mostram pouco eficientes. Os problemas concentram-se na dificuldade de se expressar a semântica dos serviços, em questões de estruturação e controle dos fluxos e na complexidade que é inserida com a sua utilização. O WSMO é outra linguagem que vem ganhando espaço sobre o OWL-S, mas ainda não existe consenso dos autores sobre qual é a linguagem mais adequada para esse tipo de abordagem.

A falta de direcionamento claro sobre essas linguagens que viriam a padronizar a definição de composições prejudica o que seria de fato o seu ponto forte, a portabilidade. Uma linguagem padrão e robusta deve permitir que sejam construídos motores de execução de diferentes fornecedores, sendo que a solução não estaria obrigatoriamente ligada a um fornecedor em específico. Porém, isso ainda não aconteceu, as linguagens apesar de possuírem uma especificação bem avançada, ainda não contam com motores de execução de fácil acesso.

Dessa forma, devido à falta de uma linguagem padronizada e de uso consensual, e de um motor que a suporte adequadamente, foi construído um módulo específico para a execução de composições de Serviços Web, chamado Módulo de Execução (MDE), o qual é capaz de tratar em tempo de execução os planos gerados (sob a forma de árvores) pelo MDP. O MDE recebe o plano gerado pelo MDP, sendo capaz

de interpretar a lógica de invocação dos Serviços Web e de verificar o resultado obtido após a invocação de cada operação. Este módulo é responsável também por atualizar o estado do mundo modelado com base nos resultados das ações e por decidir, de acordo com esse estado, a próxima ação a ser executada.

O MDE inicia o processamento do plano, identificando a primeira tarefa a executar e verifica o operador associado a ela para determinar quais são as suas pré-condições. Depois que são validadas, todas as pré-condições, de acordo com o estado inicial definido, a tarefa é executada. As tarefas básicas podem corresponder a ações internas (determinísticas) de controle ou a invocações de operações (não-determinísticas) de Serviços Web. No primeiro caso, os efeitos da ação são imediatamente usados para atualizar o estado corrente do mundo. No segundo caso, a operação do Serviço Web precisa ser invocada e verificado o resultado obtido.

Durante a execução, o MDE processa operadores associados à operação de Serviço Web que deverá ser invocada, validando suas pré-condições e carregando informações necessárias para a invocação do Serviço Web. A validação das pré-condições ocorre quando o conjunto de estados corrente satisfaz todas as pré-condições exigidas para a invocação do Serviço Web. As operações representam visões abstratas das capacidades do Serviço Web, não descrevendo questões inerentes a estruturas de dados nem tão pouco sobre protocolos utilizados. Dessa forma, o MDE solicita a chamada de uma operação do Serviço Web ao MGS, que possui o conhecimento concreto sobre como invocar o Serviço Web, qual protocolo utilizar, qual endereço físico e como especificar a estrutura de dados utilizada pelo Serviço Web. O MGS invoca o serviço e repassa o retorno obtido ao MDE. Este, por sua vez, traduz o retorno em termos de efeitos que indicam como o modelo do mundo deve ser modificado para refletir o resultado da invocação. Para tal, os operadores possuem uma lista de listas de efeitos não determinísticos. Associado a cada lista de efeitos, existe uma condição de teste (em cima dos valores de retorno) que permite identificar quais os efeitos que devem ser considerados na atualização do estado do mundo.

Os estados são modelados como conjuntos de fatos Prolog positivos. Os efeitos poderão resultar em atividades de adição ou remoção de fatos no estado corrente. Quando um efeito é dito positivo, é realizada a adição de um novo fato ao estado (se

este fato ainda não pertencia ao estado). Quando o efeito é dito negativo, um fato é retirado do estado corrente (se o fato pertencia ao estado). Assim, a cada execução de um Serviço Web ou de uma ação interna, o estado corrente da composição é atualizado, permitindo que o motor possa verificar no plano as condições de teste que ligam a ação que acabou de ser executada às próximas possíveis ações, determinando qual será a próxima ação a ser executada ou finalizando o processo quando não existirem mais ações a serem realizadas.

O MDE deve ainda estar preparado para lidar com eventuais falhas, como, por exemplo, o Serviço Web não retornar nenhuma resposta dentro do tempo definido. Essas falhas deverão ser mapeadas para condições que o MDE possa tratar e definir ações de correção e/ou seguir com um fluxo de execução alternativo.

Podem existir situações onde a ação que resultou em um estado de erro era fundamental para a continuidade da execução da composição de Serviços Web, nesses casos, o MDE pode optar por interromper a execução e definir um conjunto de atividades de compensação, visando alcançar um estado de integridade.

Entretanto, pode não ser possível desfazer atividades previamente realizadas, havendo o risco de a composição permanecer em um estado não previsto. Nesse caso, o MDE deve registrar o estado de falha e dar a execução do plano corrente por terminada. O MMC deverá então tratar essas situações.

3.5. Monitoração e Controle de Composições

O MMC possui a responsabilidade de capturar informações do contexto de execução dos Serviços Web a serem invocados. A definição do termo contexto é necessária para o correto entendimento do seu comportamento. Muitos pesquisadores já publicaram trabalhos descrevendo definições sobre o termo, mas uma das mais aceitas atualmente é:

"Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade (pessoa, lugar ou objeto) que é considerada relevante para a interação entre um usuário e uma aplicação, incluindo-os.". (DEY, 2001)

Apesar de ser uma definição um pouco abrangente, essa definição, quando olhada sob a perspectiva da composição de Serviços Web, nos permite observar o extenso leque de possibilidades que podem ser exploradas no que tange à automação do processo de geração e execução de composições de Serviços Web.

Dentro da Computação Ubíqua, uma das suas principais áreas de pesquisa, a Computação Sensível ao Contexto (*Context-Aware Computing*), vem trabalhando muito próximo desse tema e propõe a padronização de uma maneira de capturar entradas capazes de refletir a situação atual do usuário, do ambiente no qual o mesmo se encontra e do aparato computacional utilizado. Tais entradas são os chamados contextos. A Computação Sensível ao Contexto possui muitas pesquisas sendo realizadas para tratar assuntos ainda em aberto, destacando-se a automação de tarefas, a identificação e escolha de fontes de contextos, o tratamento de falhas e a utilização de algoritmos de aprendizado. Esses assuntos assemelham-se diretamente com as questões que serão vistas no MMC.

Problemas de composição de Serviços Web tendem a possuir uma quantidade significativa de Serviços Web envolvidos e efeitos diversificados. Normalmente, uma solução de composição de Serviços Web está inserida em um contexto onde os Serviços Web participantes estão distribuídos entre aplicações remotas. Essas aplicações podem estar dentro de uma mesma rede local, em redes distribuídas ou mesmo sobre a Internet. Dessa forma, muitos fatores podem influenciar negativamente a execução de uma composição (SCHROEDER *et al.*, 2006, LONG *et al.*, 1995). Um plano gerado por um planejador não-determinístico pode possuir um número considerável de ações necessárias para se alcançar o objetivo definido. Como cada ação pode conter efeitos não-determinísticos, isso pode provocar uma explosão combinatória no número de caminhos alternativos a serem tratados e, mesmo que o planejador não-determinístico leve em consideração todas as alternativas conhecidas para a geração de um plano, ainda não há como garantir que, durante a execução, não apareçam situações inesperadas, resultando em falhas. Dessa forma, é necessário ter mecanismos para limitar o número de alternativas consideradas pelo planejamento não-determinístico e para permitir que falhas sejam tratadas quando a execução de uma composição de serviços leva a um estado indesejável. Suponha, por exemplo,

que exista uma aplicação para efetuar agendamentos automáticos de consultas médicas. Uma vez que ocorra uma solicitação, a aplicação deve gerar e executar uma composição utilizando os Serviços Web de consulta à agenda do cliente e do médico para verificar uma data adequada para a consulta e em seguida, efetuar a chamada ao Serviço Web de agendamento de consultas. Um planejador não-determinístico pode levar em conta o fato de um médico já estar ocupado e prever o agendamento da consulta com outro médico. No entanto, para se tentar considerar todas as eventualidades, pode ser necessário criar planos muito grandes (prevendo por exemplo o atendimento alternativo por diversos médicos em diversos locais, no caso de impossibilidades sucessivas de atendimento) ou até infinitos, o que poderia inviabilizar a geração e execução do plano.

Ainda que o objetivo seja alcançado com sucesso, após um período de tempo, o resultado pode não ser mais relevante. Certamente, não há como alterar o resultado obtido, mas pode perder o seu valor ou se tornar inconsistente dentro de uma nova realidade, exigindo que sejam executadas ações para se retornar a um estado consistente. Usando o mesmo exemplo do agendamento de consultas médicas, assuma, por exemplo que, pouco tempo depois a marcação de uma consulta, o médico registrou na sua agenda, uma atividade inalterável que coincide com o agendamento da consulta do cliente. Nota-se que apesar do objetivo da composição do Serviço Web ter sido alcançado inicialmente, o contexto nesse momento encontra-se em estado inconsistente.

Por fim, alterações no contexto decorrentes da interação com o usuário e com outros agentes pode levar à necessidade de se executar novas tarefas para se atingir novos objetivos.

Dessa forma, surge a necessidade de monitoração contínua do contexto tanto para reduzir a complexidade do planejamento quanto para tratar situações inesperadas e novas necessidades de alcançar objetivos.

O Módulo de Monitoração e Controle tem a função de monitorar continuamente o contexto para detectar situações que demandam a geração e execução de composições de Serviços Web. No exemplo do agendamento de consultas médicas, o

Módulo de Monitoração e Controle poderia detectar, por exemplo, a indisponibilidade do médico para a consulta em questão e, a partir da base de regras de inferência, presente no Repositório de Conhecimento, determinar que a consulta tenha que ser desmarcada. Assim, o Módulo de Monitoração e Controle deverá solicitar um plano ao Módulo de Planejamento para o cancelamento da consulta e, em seguida, requisitar que o Módulo de Execução execute a composição gerada, objetivando solucionar o conflito de agendas. Ao cancelar o agendamento, o Módulo de Monitoração e Controle identifica uma nova situação, a existência de um pedido de cliente para consulta médica não atendida. Assim sendo, o MMC dá início a um novo processo de agendamento.

Enquanto uma composição possui um ciclo de vida controlado, pois é gerada, executada e finalizada para um objetivo, os processos de monitoração e controle permanecem ativos constantemente aguardando situações que necessitem de intervenção ou novas atividades.

Novas atividades podem ser realizadas quando o estado do mundo é alterado ou quando objetivos são alcançados com sucesso. Regras de inferência associadas a estados específicos alcançados por meio da execução com sucesso de uma composição podem levar o Módulo de Monitoração e Controle a iniciar um novo processo de geração e execução de composição. No exemplo de agendamentos automáticos de consultas médicas, o sucesso da marcação da consulta médica para um paciente pode gerar uma nova atividade a ser realizada, por exemplo, a pré-aprovação da consulta médica pelo plano de saúde. Assim, é gerada uma nova composição para efetuar a pré-aprovação da consulta médica, sendo necessários outros Serviços Web, como o de solicitação de pré-aprovação de consulta médica com o respectivo plano de saúde e o serviço de notificação de confirmação de pré-aprovação para o médico.

4. Detalhamento da Solução Proposta

O capítulo 3 detalhou os principais sub-processos pertencentes ao processo de execução e o relacionamento com os módulos da solução. O capítulo 4 está organizado de forma a apresentar o funcionamento interno de cada módulo e como foi realizado o seu desenvolvimento.

Inicialmente, as ferramentas e linguagens de desenvolvimento que foram utilizadas na solução são descritas, bem como o motivo das escolhas. Então, explicam-se como as tarefas são encadeadas hierarquicamente utilizando o planejador ND-HTN e se apresenta a estrutura do plano que é gerado.

Posteriormente, descreve-se como o plano é recebido e executado, avaliando-se resultados obtidos das ações executadas e determinando o que fazer em seguida. O uso de regras de inferência que são utilizadas para o acionamento de novas tarefas e o relacionamento com os módulos de planejamento e execução são, então, descritos.

O Repositório de Conhecimento, que é o repositório central para todos os demais módulos, também é apresentado e se descreve como é realizada a comunicação com o Módulo de Execução e como efetivamente os Serviços Web são invocados. As ações a serem realizadas são traduzidas para a invocação de Serviços Web que recebem como entrada os parâmetros necessários para a execução de uma operação. Como resultado, o Serviço Web retorna um conjunto de parâmetros, que são usados para atualizar a representação interna do estado do mundo.

Finalmente, se descreve como seria a interface gráfica da solução, que, entretanto, não foi efetivamente desenvolvida, permanecendo como trabalho futuro.

4.1. Implementação da Solução

Para a implementação da solução, foi utilizado um conjunto de ferramentas e linguagens para compor os módulos e os componentes implementados para testar toda a solução, conforme é apresentado na figura 4-1.

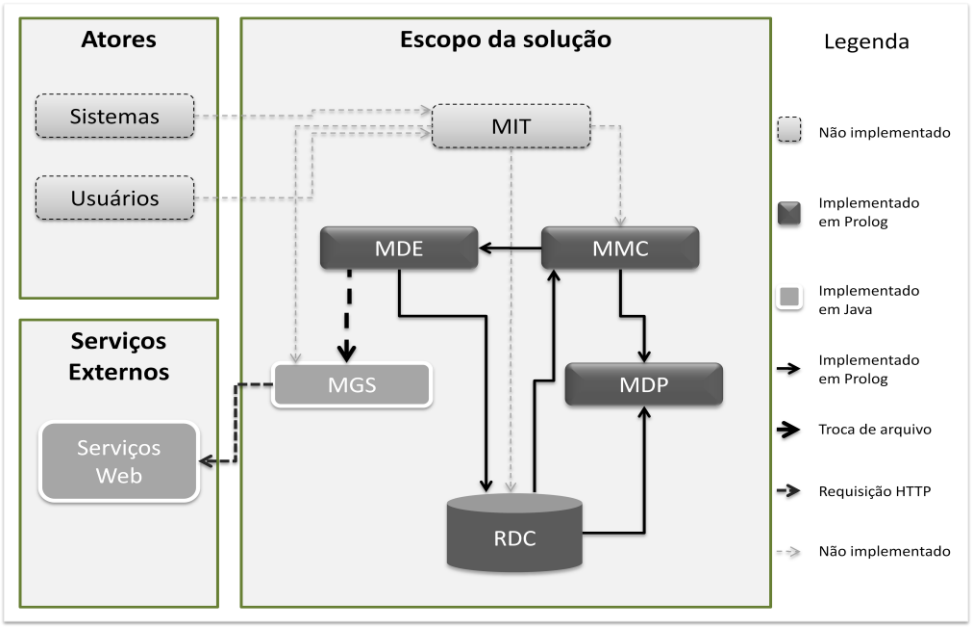

Figura 4-1- Implementação da Solução

Na figura 4-1 é possível observar como a solução foi desenvolvida e as interfaces necessárias para o funcionamento correto da solução.

O Prolog é uma linguagem utilizada em Inteligência Artificial que permite a declaração de fatos e regras sobre objetos (BRATKO, 1986), o que facilita a definição de processos de inferência. Todos os módulos foram desenvolvidos em Prolog, exceto o MIT, que não foi efetivamente desenvolvido e o MGS. Para a execução da lógica desenvolvida em Prolog foi utilizado o *SWI Prolog* 5.10.0[1].

A linguagem Java foi utilizada para a construção do MGS, pois é o módulo que efetivamente faz comunicação com Serviços Web. A linguagem Java foi escolhida, pois possui APIs nativas para a comunicação com Serviços Web, o que facilita o processo de integração.

[1] http://www.swi-prolog.org/download/stable

No ambiente de desenvolvimento foi utilizada a Plataforma *Eclipse* 3.5.2[1] como interface principal para o desenvolvimento dos códigos Java e Prolog. Para o desenvolvimento de todos os códigos Java foi utilizado o *Java Development Kit* 1.5.0_21[2].

Também foram construídos Serviços Web para simular os serviços de uma empresa de Hotéis e uma empresa de Companhia Aérea, conforme descrito no capítulo 5. A geração dos Serviços Web e *Proxys* utilizaram os componentes do Eclipse *Web Tool Platform* 3.1.1[3]. Os Serviços Web foram gerados utilizando o padrão *Document/literal*[4]. O servidor de aplicação para executar os Serviços Web da Companhia Aérea e do Hotel foi o *Apache Tomcat* v6.0[5].

4.2. Módulo de Planejamento

O Módulo de Planejamento é responsável por gerar um plano para satisfazer um objetivo. O plano gerado pelo ND-HTN (SILVA, 2010) possui uma estrutura simples e sequencial, baseada em listas Prolog, o que facilita a sua execução. Os caracteres "[" e "]" indicam em Prolog o início e fim de estruturas do tipo lista. Dentro da estrutura de um plano, os caracteres "(" e ")" são utilizados para limitar sub-planos. Essa sintaxe é utiliza tanto pelo planejador do MDP quanto pelo *parse* que o MDE executa. A estrutura do plano pode ser descrita por:

plan = task,[[cond 1, (sub_plan 1)],...,[cond N, (sub_plan N)]]

- *Task* - É a tarefa que deverá ser realizada, podendo ser uma tarefa interna ou externa, sendo obrigatória que seja mencionada uma tarefa. Uma tarefa interna é definida por um operador básico que não resulta em uma chamada a

[1] http://www.eclipse.org/downloads/
[2] https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jdk-1.5.0_21-oth-JPR@CDS-CDS_Developer
[3] http://download.eclipse.org/webtools/downloads/
[4] Padrão indicado para garantir interoperabilidade entre plataformas conforme publicado em http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html
[5] http://tomcat.apache.org/download-60.cgi

Serviços Web. Uma tarefa externa é definida por um operador básico que efetua chamada a um Serviço Web.

- *Cond* - Uma condição que deve refletir um estado alcançado após a execução de uma ação. Tarefas internas são determinísticas e geram apenas uma "true" (avaliada sempre como verdade). Tarefas externas são tratadas como não-determinísticas e podem possuir *n* condições.

- *Sub_plan* - O sub-plano possui a mesma estrutura de um plano, sendo executado quando sua respectiva condição for satisfeita. Um sub-plano vazio indica que o fim do plano foi alcançado.

Por se tratar de um planejador HTN, para que um plano possa ser gerado, é necessário que sejam descritas tarefas abstratas contendo um encadeamento de tarefas primitivas. Tarefas abstratas também podem conter outras tarefas abstratas, ou ainda, uma combinação entre tarefas primitivas e abstratas. Tarefas abstratas são combinadas para permitir a representação de grandes atividades para situações de diferentes complexidades. Entretanto, um plano gerado possui apenas tarefas primitivas, ou seja, tarefas que exercem diretamente uma ação sobre o ambiente causando algum tipo de efeito, não contendo sub-tarefas. Assim, um plano gerado consiste das tarefas primitivas resultantes da decomposição das tarefas abstratas de modo a se cumprir um objetivo.

Para que o planejador ND-HTN possa gerar um plano, é necessário que todas as tarefas sejam definidas através de operadores, sejam elas primitivas ou abstratas. Um operador descreve todas as características necessárias de uma tarefa e possui a seguinte notação:

operator(id, task, input, output, preconds, effects, ndet_effects, cost, main_effects, sub_tasks, order).

- *Id* – Identificador do operador, número inteiro que identifica um único operador.

- *Task* – Nome da tarefa. A tarefa contém ainda os parâmetros que serão utilizados pelo plano;

- *Input* – Lista Prolog com os parâmetros de Entrada da operação de Serviço Web associada ao operador (incluídos para adequação com a solução SACCAS);

- *Output* – Lista Prolog com os parâmetros de Saída da operação de Serviço Web associada ao operador (incluídos para adequação com a solução SACCAS);

- *Preconds* – Lista Prolog com as pré-condições (literais Prolog-fatos ou negação de fatos) para que uma tarefa possa ser executada;

- *Effects* – Lista Prolog com os efeitos determinísticos (literais Prolog) que uma tarefa irá gerar em qualquer situação;

- *Ndet_effects* – Define os efeitos não-determinísticos que uma tarefa irá gerar dependendo da condição alcançada pela tarefa (alterado para adequação com a solução SACCAS), representada por uma lista, onde cada elemento é uma lista de efeitos, correspondendo a um dos possíveis resultados da execução do operador;

- *Cost* – Custo do operador. Não utilizado na solução SACCAS, pois não é levado em consideração custo na escolha de um melhor caminho.

- *Main_effects* – Efeitos gerais do operador. Está presente no operador, mas não é utilizado em tempo de planejamento e não foi utilizado na solução SACCAS;

- *Sub_tasks* – Sub-tarefas que deverão ser invocadas por essa tarefa. Uma tarefa que possui sub-tarefas é considerada uma tarefa abstrata.

- *Order* – Ordem que as sub-tarefas deverão ser invocadas. Sem a inclusão dessa informação as sub-tarefas serão executadas de forma aleatória.

A estrutura original do operador no planejador foi modificada para permitir a definição do *input* e do *output*. Essas definições não são utilizadas em tempo de planejamento. Entretanto, durante a execução, os operadores são verificados antes de

uma chamada de Serviço Web para identificar os parâmetros que serão passados e retornados.

Operadores relacionados com tarefas abstratas possibilitam a hierarquia de herança entre operadores dependentes - relacionados às sub-tarefas pertencentes a tarefa abstrata. A herança de operadores permite a propagação de características dos operadores mais abstratos para os operadores subseqüentes, possibilitando que diferentes ramificações sejam criadas até alcançar operadores respectivos as tarefas primitivas. Cada tarefa primitiva pode alcançar um resultado diferente que será consolidado nas tarefas mais abstratas, progressivamente, até alcançar a tarefa abstrata mais geral, permitindo assim que a tarefa abstrata possua diferentes soluções.

Como os operadores relacionados com tarefas abstratas permitem que sejam definidas inúmeras ramificações para resolver problemas mais específicos, a tarefa abstrata pode ser utilizada na resolução de problemas bem complexos (com muitas possibilidades de finalização) como é o caso de composições de Serviços Web.

No contexto de composição de Serviços Web, os operadores que representam tarefas primitivas possuem comportamentos distintos para representar tarefas internas e tarefas externas.

As tarefas internas são utilizadas na composição de Serviços Web para realizar atividades dentro do ambiente da solução e não invocam Serviços Web, assim, as definições de *input* e *output* do operador não são relevantes. Essas atividades podem ser desde a busca de um dado existente dentro do ambiente, representação de regras de negócio embutidas no operador ou para a consistência de uma informação. As tarefas internas são definidas como tarefas determinísticas e, conseqüentemente, não geram efeitos *ndet_effects.*

As tarefas externas são utilizadas exclusivamente para a chamada de Serviços Web, dessa forma, as definições de *input* e *output* do operador serão utilizadas em tempo de execução. A solução de composição de Serviços Web trata toda chamada a um Serviço Web como sendo uma tarefa não-determinística, assim, o operador de uma tarefa externa deverá possuir obrigatoriamente efeitos não-determinísticos.

Originalmente, o planejador ND-HTN não previa condições nos efeitos não-determinísticos descritos em *ndet_effects*. Conforme descreve código 4-1 abaixo:

```
[hotel_reservado(ARE,FUNC,DATA, DESTINO, ALT_HOTEL_RESERVADA,HOSPEDAGEM)],
[hotel_nao_reservado(ARE,FUNC,DATA, DESTINO, ALTS_HOTEIS)]
```

Código 4-1 Exemplo de Efeitos Não-determinísticos

Os termos *hotel_reservado* e *hotel_nao_reservado* descrevem dois estados possíveis para efeitos não-determinísticos. Porém, o retorno de uma invocação de um Serviço Web tradicional não informa o efeito que foi ocasionado pela sua execução. Sendo assim, através dos parâmetros retornados pelo Serviço Web, é realizado o mapeamento para um efeito não-determinístico através de condições previamente configuradas. Conforme descreve código 4-2 abaixo:

```
(
        (RESERVA_HOTEL_OK = true),
                [hotel_reservado(ARE,FUNC,DATA, DESTINO,
        ALT_HOTEL_RESERVADA,HOSPEDAGEM)]
),
(
        (RESERVA_HOTEL_OK = false),
                [hotel_nao_reservado(ARE,FUNC,DATA, DESTINO, ALTS_HOTEIS)]
)
```

Código 4-2 Exemplo de Efeitos Não-determinísticos com Condições

No exemplo acima, *RESERVA_HOTEL_OK* é um parâmetro retornado por um Serviço Web e baseado no seu valor, é mapeado um estado correspondente, podendo ser *true*, para indicar que a reserva foi efetuada ou *false*, para indicar que a reserva não foi efetuada. É interpretado o estado alcançado pela execução do Serviço Web e, assim, mapeado o estado correspondente.

Dessa forma, a estrutura original do operador foi modificada para permitir a inclusão de uma lista de condições em *ndet_effects*, conforme notação abaixo:

((lst_cond 1), [ndet_effect 1]),…,((lst_cond n), [ndet_effect n])

- *lst_cond* – Lista de condições (literais Prolog) que determinará qual é o efeito não-determinístico que foi gerado a partir da invocação do Serviço Web;

- *ndet_effect* – Efeito não determinístico presente no operador.

O planejador ND-HTN foi modificado para ignorar a lista de condições (*lst_cond*), uma vez que será utilizada em tempo de execução, e considera apenas os efeitos não-determinísticos, que em tempo de planejamento, gerarão tantas alternativas quantos forem o número de listas alternativas de efeitos não-determinísticos existentes.

Ainda com relação às tarefas, os seus parâmetros, definidos em *Task*, são instanciados em tempo de planejamento, a partir das informações fornecidas para ND-HTN. Por exemplo, sejam *t1* e *t2* operadores que definem duas tarefas distintas e *t2* uma sub-tarefa primitiva de *t1*. Se ambos possuem o parâmetro *A* em comum e o planejador identifica que o parâmetro *A* possui o valor *'v'*, o plano gerado irá conter a tarefa primitiva *t2* com o valor *'v'* no lugar do parâmetro *A*.

A geração de planos com os valores dos parâmetros instanciados de uma tarefa potencializa tanto a geração do plano quanto a execução do plano em si. A geração do plano é facilitada, pois conhecer os valores das variáveis das suas tarefas permite efetuar otimizações e controles no plano específicas para uma execução.

Suponha a geração de um plano onde exista a necessidade de se repetir a mesma tarefa *n* vezes. O fato de não se conhecer o valor de *n* dificulta ou impossibilita a geração do plano, pois os planejadores não permitem instruções de repetição. Como tarefas abstratas podem ter definições recursivas, o conhecimento de *n* pode ser usado para criar no plano *n* execuções de uma tarefa básica.

Efeitos positivos acrescentam novos fatos ao estado corrente, enquanto os efeitos negativos retiram fatos do estado corrente. Essa propagação de estados é fundamental para o planejador validar as pré-condições das tarefas subsequentes e auxilia diretamente no carregamento das variáveis do sistema.

Durante a geração do plano, o planejador ND-HTN trabalha com todas as possibilidades de transição de estados causadas por operadores não-determinísticos. A partir do estado inicial corrente cada execução de um operador não-determinístico gera um conjunto de estados alternativos que se diferenciam pelos efeitos não-

determinísticos. Para cada alternativa, o novo estado corrente é calculado e é feita uma chamada recursiva para a geração do restante do plano. Os sub-planos gerados por cada chamada recursiva são colocados no plano, usando-se os respectivos efeitos não-determinísticos para diferenciar uma alternativa de outra em tempo de execução.

A utilização de variáveis Prolog para representar os parâmetros da tarefa e argumentos das precondições, com a respectiva instanciação em tempo de execução, facilitam o mapeamento das informações utilizadas para a execução de cada tarefa. Por exemplo, uma tarefa primitiva que necessite invocar um Serviço Web utilizando os valores definidos em *input* (também através de variáveis Prolog) já possui seus valores preenchidos automaticamente através da simples instanciação de variáveis.

A geração dinâmica de composições de Serviços Web proposta pela Arquitetura do SACCAS tende a gerar planos melhores do que o de outras propostas onde as composições são geradas estaticamente, pois possibilita que sejam geradas composições mais específicas, utilizando informações conhecidas. Consegue-se assim eliminar ramificações que tenham sido identificadas como impossíveis na situação corrente. A geração de planos menores e, consequentemente, composições, permite um ganho direto com relação ao desempenho de outras soluções que envolvem a geração de composições de Serviços Web, por ser tratarem muitas vezes de problemas de ordem exponencial.

O plano gerado pelo ND-HTN representa, na íntegra, a composição do Serviço Web que será executada.

4.3. Módulo de Execução

O Módulo de Execução é responsável por executar sequencialmente uma composição de Serviços Web, verificando o melhor fluxo a ser seguido e baseando-se nas condições descritas no plano.

Inicialmente, deve identificar se a tarefa a ser executada é uma tarefa classificada como interna ou externa. A identificação ocorre a partir de uma consulta

ao Repositório de Conhecimento que possui uma classificação de todos os tipos de tarefas registradas. Caso seja uma tarefa interna, a execução ocorre localmente, não existindo invocação de Serviços Web externos. Para as tarefas externas, o Repositório de Conhecimento efetua um mapeamento para qual Serviço Web deverá ser realizada a invocação.

A invocação ao Serviço Web não é realizada diretamente pelo Módulo de Execução. É enviada uma solicitação de invocação, de forma assíncrona, para o Módulo de Gestão de Serviços que possui o papel de invocar o Serviço Web propriamente dito e retornar o resultado obtido para o Módulo de Execução. Este por sua vez, verifica as condições de efeitos não-determinísticos definidas no operador da tarefa em *ndet_effects*, para identificar qual foi o efeito gerado e em seguida, atualiza o estado corrente da representação interna do mundo com os efeitos positivos e negativos da tarefa, incluindo os efeitos definidos em *effects*. Como mencionado anteriormente, efeitos positivos acrescentam novos fatos ao estado corrente, enquanto os efeitos negativos retiram fatos do estado corrente.

O Módulo de Execução, nesse momento, deve determinar por qual sub-plano deve seguir o seu processamento. Para tanto, verifica dentre as possibilidades de tarefas existentes, a primeira que tenha a sua lista de condições satisfeita a partir do estado corrente da representação do mundo. O processo se repete até que seja encontrado um sub-plano vazio.

4.4. Módulo de Monitoração e Controle

O Módulo de Monitoração e Controle é um processo contínuo que permanece interagindo com os usuários, Serviços Web registrados e o Repositório de Conhecimento, permitindo que sejam recebidas notificações e novas solicitações de atividades, bem como realizado o controle do ambiente.

A descrição do mundo (o contexto) armazenada no Repositório de Conhecimento é alterada em decorrência da execução de composições geradas e executadas pelo SACCAS e das interações com os usuários e com Serviços Web

especificamente disponibilizados pelo SACCAS para que sistemas externos notifiquem eventos. Nesse último caso, um Serviço Web de notificação pode ser acionado por um sistema externo para que o SACCAS tome conhecimento, por exemplo, de que um recurso necessário não está disponível, o que pode demandar a execução de tarefas (por exemplo, em um contexto de viagens, há uma viagem programada com hotéis reservados, mas o vôo foi cancelado). Uma vez que a descrição do mundo é alterada no Repositório de Conhecimento, o MMC pode analisar a necessidade de realizar tarefas. As tarefas a serem realizadas podem ser decorrentes da necessidade de:

- tratar solicitações diretas de usuários;
- tratar notificações de eventos de usuários ou Serviços Web;
- tratar conflitos internos; ou
- dar continuidade a um processo longo dividido em etapas, onde cada etapa envolve a geração e execução de uma composição de serviços.

A base de regras de inferência, armazenada no Repositório de Conhecimento, reúne todas as regras de inferência da solução e define o comportamento do SACCAS. Uma regra de inferência informa o que fazer quando uma determinada situação se verifica, sendo um importante mecanismo para acionar tarefas para todo tipo de ação necessária, sejam elas corretivas, quando estados indesejados são detectados, ou aditivas, informando que novas tarefas devem ser realizadas quando certos estados são alcançados. O MMC identifica que tarefas precisam ser executadas através do exame da base de regras de inferência registradas. A condição de ativação de cada regra de inferência registrada é avaliada de acordo com o estado corrente do mundo.

A base de regras de inferência é constituída por regras de inferência registradas no Repositório de Conhecimento, possuindo a seguinte notação:

regra_inferência((lst_cond),tarefa)

- *Lst_cond* – Lista de literais correspondentes à condição de ativação da regra de inferência;

- *Tarefa* – Tarefa a ser realizada quando uma regra de inferência é satisfeita.

Quando a condição de ativação de uma regra de inferência é satisfeita, inicia-se o processo de realização da tarefa presente na regra de inferência. A realização ocorre com o MMC requisitando, ao Módulo de Planejamento, a geração de um plano específico para solucionar a tarefa. O plano gerado é repassado ao Módulo de Execução que inicia o processo de execução da composição.

Vale ressaltar que, dependendo da regra de inferência acionada, a composição gerada pode não levar à invocação de Serviços Web, sendo chamada de composição determinística. Nesse caso, a composição executa apenas tarefas primitivas internas, também conhecidas como tarefas determinísticas. Em geral, esse tipo de composição ocorre quando existe a necessidade de se efetuar algum tipo de validação no Repositório de Conhecimento ou solucionar inconsistências encontradas.

Assim como nas solicitações de atividades, as regras de inferência também são informadas pelo usuário. Dessa forma, o usuário possui a capacidade de configurar no SACCAS, através de regras de inferência, o comportamento esperado de uma solicitação em cada estado possível de ser alcançado. É possível ainda descrever dependências entre estados de uma mesma solicitação e regras de relacionamento entre duas ou mais solicitações, desde que previamente tenham sido criadas tarefas com tais finalidades.

Regras de inferência que descrevem a dependência entre uma situação final alcançada por uma composição para acionar uma nova tarefa, permitem que o SACCAS seja capaz de efetuar processos maiores e mais complexos. Por exemplo, suponha que um cliente deseja comprar um produto pela Internet. Se a compra for realizada com sucesso e o preço for acima de certo valor, a regra de inferência pode prever que no dia seguinte se ofereça a compra de outro produto pela metade do preço. A oferta mencionada pode ser configurada como sendo uma regra de inferência que é acionada apenas quando a venda ocorrer com sucesso e for acima de um limite. Além disso, tratando este comportamento através de uma regra de inferência, torna-se simples modificá-lo quando não há mais oferta disponíveis, bastando para isso desativar a regra de inferência.

As regras de inferência de dependência podem ainda efetuar o papel de desfazimento de tarefas já realizadas. Por exemplo, para o mesmo exemplo acima, onde o cliente efetuou uma compra pela Internet, suponha, que próximo da data de entrega, a empresa responsável detectou que o produto não existe mais em estoque. Nesse caso, pode ser acionada uma regra de inferência para gerar uma composição que irá cancelar a compra do cliente com a empresa e, em seguida, estornar o pagamento realizado pelo cliente.

As regras de inferência também podem descrever relacionamentos entre duas ou mais solicitações criando restrições ou agrupamentos. Por exemplo, um funcionário possui uma atividade em determinado local e, um determinado período de tempo depois, outra atividade é marcada para o mesmo horário. Uma regra de inferência poderia definir qual atividade deve permanecer e qual atividade deve ser cancelada baseada, por exemplo, em uma prioridade. Podem ainda ser criadas regras de inferência para tratar agrupamentos. Por exemplo, se tantas pessoas pediram um produto que não se encontra mais em estoque, a regra de inferência pode prever que se requisite automaticamente a compra desse produto para reposição do estoque.

4.5. Repositório de Conhecimento

O Repositório de Conhecimento (RDC) é responsável por armazenar regras de inferência, operadores, mapeamento para Serviços Web, informações sobre as tarefas (ações executadas, em execução e planejadas) e informações sobre o contexto do ambiente SACCAS. É acessado por todos os módulos, que o utilizam como fonte de informação para o seu processamento e como repositório para os resultados obtidos. Durante a execução, informações sobre as tarefas e sobre o contexto são continuamente atualizadas. É possível também adicionar novos operadores (e respectivos mapeamentos para Serviços Web) e regras de inferência.

Por simplicidade, no protótipo inicial implementado, as informações contidas no SACCAS são representadas no banco de dados interno do Prolog (que funciona apenas em memória principal), o que facilitou o desenvolvimento, já que a maior

parte do código do sistema é em Prolog. O conteúdo do repositório é carregado inicialmente através de um arquivo texto, onde os fatos e regras são declarados explicitamente. Note-se que, em tempo de execução, o conteúdo do repositório é modificado. Para permitir a persistência dos dados, é possível salvar o contexto novamente em arquivo para recuperação posterior. No entanto, para uma solução que garanta a persistência e segurança do processo, as informações deverão no futuro ser armazenadas em um banco de dados externo integrado ao Prolog.

4.6. Módulo de Gestão de Serviços

O Módulo de Gestão de Serviços é responsável por registrar os Serviços Web que estarão disponíveis para o SACCAS realizar o mapeamento dos parâmetros necessários para a invocação e retorno do Serviço Web e realizar a chamada dos Serviços Web propriamente ditos.

As informações sobre os Serviços Web são armazenadas no Repositório de Conhecimento e descrevem as operações disponibilizadas pelos Serviços Web, dados de entrada, saídas e também, informações sobre pré-condições e efeitos, necessárias para a etapa de geração de composição automática.

O SACCAS assume que as operações de cada Serviço Web registrado podem ser modeladas como processos atômicos independentes, onde é feita uma invocação que retorna uma resposta. Dessa forma, ao ser registrada, cada operação do Serviço Web pode ser representada internamente por um operador com pré-condições e efeitos, cuja correspondência com a entrada e a saída da operação é devidamente indicada. Quando a situação da interação com um Serviço Web com várias operações é relevante para a geração de composições, isso é devidamente representado no contexto por meio de fatos. Esses fatos podem então ser verificados nas pré-condições e alterados nos efeitos dos operadores.

Os Serviços Web são invocados quando o MDE requisita ao MGS, de forma assíncrona, a execução de uma tarefa externa. O MGS possui os *proxys* dos Serviços Web registrados. O *proxy* é um cliente de um Serviço Web que foi gerado

previamente, durante o registro do Serviço Web, e que conhece como efetuar a chamada remota ao Serviço Web, possuindo informações relativas a protocolos a serem utilizados e como estruturar dados enviados e retornados.

4.7. Módulo de Interação com Terceiros

O Módulo de Interação com Terceiros (MIT) foi previsto, mas não foi desenvolvido para esta versão do protótipo. A proposta é que seja responsável por receber as requisições oriundas dos atores externos. As requisições recebidas serão traduzidas e armazenadas no Repositório de Conhecimento e posteriormente serão utilizadas pelos demais módulos para que possam cumprir suas atividades.

Através do MIT, o ator externo poderá manipular Serviços Web, regras de inferência ou objetivos, bem como acompanhar o andamento do ambiente: as ações executadas, ações que estão em andamento e as ações que estão planejadas. O ator poderá criar novos objetivos, alterar ou cancelar um objetivo existente.

O objetivo informado pelo ator é a etapa inicial para o funcionamento do ambiente, pois é através dos objetivos que o ambiente deve iniciar as atividades de planejamento de tarefas a serem cumpridas.

Um objetivo já planejado pode ser alterado ou excluído. Entretanto, o MIT nessa etapa não leva em consideração os efeitos que essas mudanças terão sobre o ambiente, sendo essa uma responsabilidade do Módulo de Monitoração e Controle.

As regras de inferência são aplicadas sobre o ambiente para permitir a criação de regras ou restrições que serão levadas em conta na etapa de planejamento. O ator poderá incluir novas regras de inferência, alterar ou cancelar regras de inferência existentes, mas de forma similar aos objetivos, os efeitos causados por tais mudanças serão tratados pelo Módulo de Controle.

Com relação à manipulação de Serviços Web, os atores poderão registrar, alterar ou excluir. Posteriormente, os Serviços Web serão invocados pelo Módulo de

Monitoração e Controle para realizar ações previamente planejadas para completar uma tarefa.

O MIT permitirá que os atores possam acompanhar a execução das atividades dentro do SACCAS, verificando ações que já foram executadas, ações que estão planejadas e ações que estão em andamento. Assim, a partir de um objetivo previamente cadastrado, é possível verificar o que o SACCAS executou para cumprir o objetivo. Situações não previstas podem ocorrer e o SACCAS pode não ser capaz de solucioná-las, nesses casos, o usuário poderá verificar e informar o que deve ser feito para solucionar a situação.

O MIT irá interagir com os atores através de uma interface gráfica própria, quando o ator é um usuário, ou através de Serviços Web específicos, quando os consumidores são sistemas externos.

5. Estudo de Caso: Marcação de Viagens

O capítulo 4 descreveu o funcionamento de cada módulo pertencente à arquitetura proposta pelo SACCAS e como foram desenvolvidos para funcionar de forma desacoplada, permitindo que possam evoluir de forma independente.

Este capítulo tem como objetivo descrever um cenário hipotético para se exemplificar a utilização do SACCAS. Para este estudo de caso, será descrito um cenário de marcações de viagens para executivos de uma companhia, através da agência de viagens prestadora de serviços de agendamentos inteligentes. Ambas as empresas são fictícias, descritas apenas para contextualizar o cenário.

O capítulo está organizado em cinco seções. A primeira descreve o cenário e as principais características funcionais da solução. A segunda seção descreve os serviços web construídos para possibilitar a simulação de serviços web reais respectivos a uma companhia aérea e a uma agência de hotéis. A terceira seção descreve como foi realizada a configuração da solução para operar no cenário considerado e a quarta seção descreve como foram realizados os testes no ambiente. Finalmente, a quinta seção apresenta os resultados obtidos.

5.1. O Cenário de Agendamento de Viagens

Uma agência de viagens possui um sistema capaz de gerenciar compromissos de seus clientes e efetuar reservas de recursos necessários para as suas atividades, tais como marcações de voos, hotéis e etc. Havendo conflitos na agenda dos clientes, compromissos desmarcados ou necessidades como viagens de última hora, o sistema

possui a capacidade de organizar os compromissos do cliente e, consequentemente, os recursos associados necessários, otimizando tempo de deslocamento e custos.

Através deste sistema, a agência de viagens oferece um serviço exclusivo chamado Agendamentos de Reuniões Executivas (ARE). Esse serviço permite que, a partir de uma reunião agendada, em um determinado local e data, os recursos necessários para o grupo de participantes sejam reservados, desde que não existam conflitos de agenda com outras reuniões para nenhum dos participantes. Caso contrário, a reunião não poderá ser agendada.

5.2. Descrição dos Serviços Web Utilizados

Para que o sistema da agência de viagens possa funcionar adequadamente, Serviços web de parceiros devem ser disponibilizados para a realização de operações de consulta, reserva e liberação de recursos. Para o cenário proposto, foram construídos Serviços Web fictícios para a simulação de uma Companhia Aérea e Reservas de Hotel.

5.2.1. Serviços Web Companhia Aérea

O Serviço Web da Companhia Aérea possui as operações apresentadas no esquema da figura 5-1.

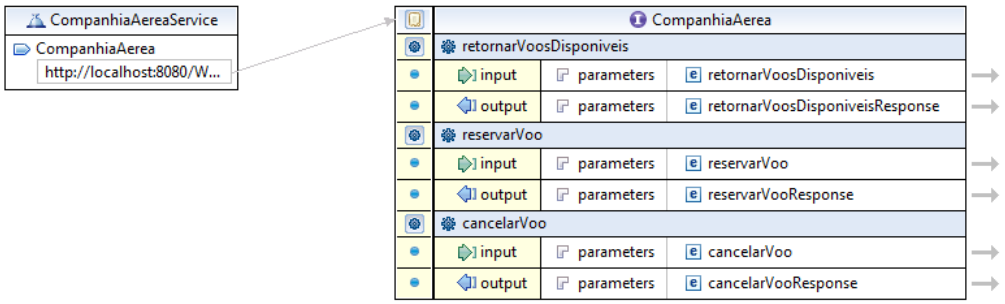

Figura 5-1 - Operações do Serviço Web da Companhia Aérea

A figura 5-1 exibe que o Serviço Web *CompanhiaAereaService* está relacionado com as operações: *retornarVoosDisponíveis, reservarVoo e cancelarVoo.* Cada operação está relacionada a uma mensagem de entrada (*input*) e uma mensagem de saída (*output*) finalizada com a palavra *Response*.

Para a operação *retonarVoosDisponíveis* (Figura 5-2), é enviada a mensagem *retornarVoosDisponíveis* com os parâmetros de entrada (*cidade_ori, cidade_des, data_ini*), bem como o tipo de cada um, que, no exemplo, é *string* para os três parâmetros. O sistema da empresa aérea usa esses parâmetros para pesquisar em sua base de dados os voos disponíveis entre duas cidades a partir de uma data inicial informada. A grande maioria dos parâmetros do exemplo está definida com o tipo *string* para facilitar a codificação dos Serviços Web e em nada impactam a solução de composição.

Figura 5-2 - Descrição da operação retornarVoosdisponiveis

A operação *retonarVoosDisponíveis* (Figura 5-3) retorna a mensagem *retornarVoosDisponíveisResponse* com o parâmetro *retornarVoosDisponíveisReturn* do tipo *string*, contendo uma lista de voos disponíveis.

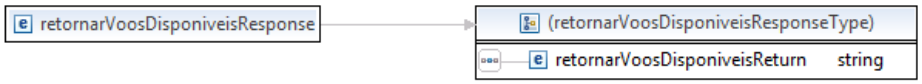

Figura 5-3 - Descrição da operação retornarVoosDisponiveisResponse

Para a operação *reservarVoo* (Figura 5-4), é enviada a mensagem *reservarVoo* com os parâmetros que são enviados para o Serviço Web, o qual deve efetuar a reserva de uma passagem para o passageiro especificado (*passageiro*), entre a cidade de origem (*cidade_ori*) e a cidade de destino (*cidade_des*), podendo haver escalas (*trajeto*) na data informada (*data_ini*).

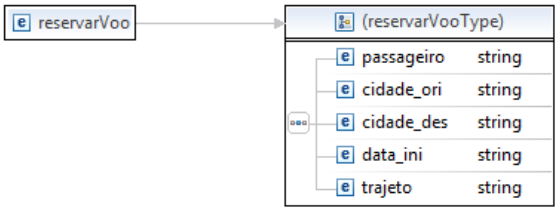

Figura 5-4 - Descrição da operação reservarVoo

A operação *reservarVoo* (Figura 5-5) retorna a mensagem *reservarVooResponse* com o parâmetro *reservarVooReturn* contendo a passagem reservada ou "-1" para descrever que ocorreu uma falha na reserva.

Figura 5-5 - Descrição da operação reservarVooResponse

Para a operação *cancelarVoo* (Figura 5-6), é enviada a mensagem *cancelarVoo* com o parâmetro ticket do Serviço Web que deve efetuar o cancelamento de uma passagem a partir de um ticket informado.

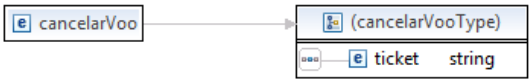

Figura 5-6 - Descrição da operação cancelarVoo

A operação *cancelarVoo* (Figura 5-7) retorna a mensagem *cancelarVooResponse* com o parâmetro *cancelarVooReturn* contendo um número positivo para indicar sucesso ou um número negativo para indicar que não foi possível efetuar o cancelamento.

Figura 5-7 - Descrição da operação cancelarVooResponse

5.2.2. Serviços Web Hotel

O segundo Serviço Web desenvolvido foi o hotel com as seguintes operações:

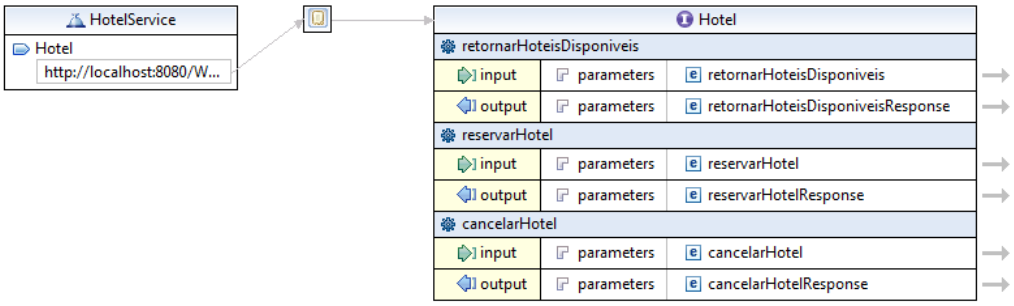

Figura 5-8 - Operações do Serviço Web do Hotel

A figura 5-8 exibe que o Serviço Web *HotelService* está relacionado com as operações: *retornarHoteisDisponíveis, reservarHotel e cancelarHotel*. Cada operação está relacionada com uma mensagem de entrada (input) e uma mensagem de saída (output) finalizada com a palavra *Response*.

Para a operação *retonarHoteisDisponíveis* (Figura 5-9), é enviada a mensagem *retornarHoteisDisponíveis* com os parâmetros cidade de destino (cidade_des) e a data de chegada (data_ini) que são enviados para que o Serviço Web possa consultar a existência ou não de hotéis disponíveis para o local e data informados.

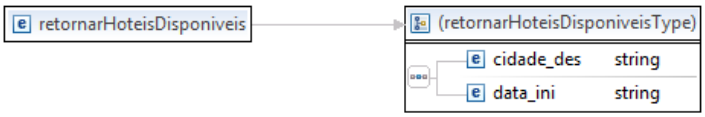

Figura 5-9 - Descrição da operação retornarHoteisDisponiveis

A operação *retonarHoteisDisponíveis* (Figura 5-10) retorna a mensagem *retonarHoteisDisponíveisResponse* com o parâmetro *retornarHoteisDisponiveisReturn* com a lista de hotéis disponíveis.

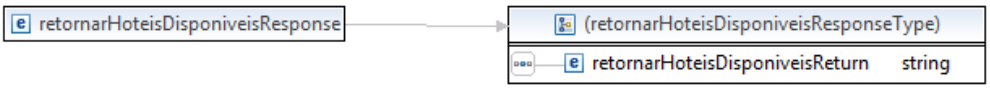

Figura 5-10 - Descrição da operação retornarHoteisDisponiveisResponse

Para a operação *reservarHotel* (Figura 5-11), é enviada a mensagem *reservarHotel* com os parâmetros nome do hospede (*hospede*), cidade da reserva (*cidade_des*), data da reserva (*data_ini*) e o nome do hotel (*hotel*), para que o Serviço Web possa efetuar a reserva do hotel solicitado.

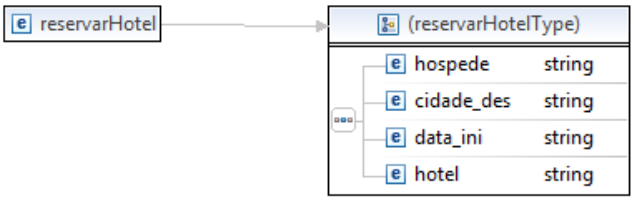

Figura 5-11 - Descrição da operação reservarHotel

A operação *reservarHotel* (Figura 5-12) retorna a mensagem *reservarHotelResponse* com o parâmetro *reservarHotelReturn* que deve conter o código da reserva efetuada, se ocorreu com sucesso, ou "-1" se ocorreu algum problema durante a atividade de reserva.

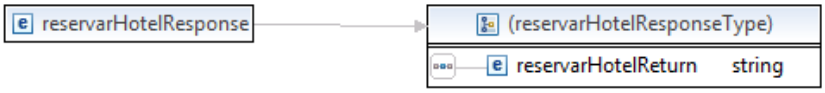

Figura 5-12 - Descrição da operação reservarHotelResponse

Para a operação cancelarHotel (Figura 5-13), é enviada a mensagem cancelarHotel com o parâmetro cod_reserva que deve conter o código da reserva que deverá ser cancelada.

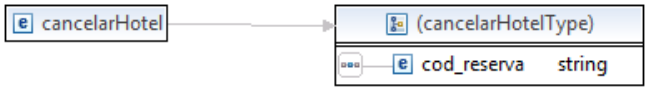

Figura 5-13 - Descrição da operação cancelarHotel

A operação *cancelarHotel* (Figura 5-14) retorna a mensagem *cancelarHotelResponse* com o parâmetro *cancelarHotelReturn* que deve conter um

número positivo para indicar sucesso no cancelamento, caso contrário, retornará um número negativo indicando que não foi possível efetuar o cancelamento da reserva do hotel.

Figura 5-14 - Descrição da operação cancelarHotelResponse

Os Serviços Web construídos são chamados pelo Módulo de Execução, através do Módulo de Gestão de Serviços, durante a execução de uma composição de Serviço Web gerada automaticamente para tarefas relacionadas com uma ARE.

5.3. Configuração do Ambiente

A configuração do ambiente é realizada definindo-se:

- os predicados a serem usados para descrever os objetos que deverão possuir o seu estado monitorado e controlado a partir de regras de inferência associadas;
- os operadores primitivos não-determinísticos a serem associados às tarefas desempenhadas por Serviços Web não-determinísticos;
- os operadores primitivos determinísticos a serem associados às tarefas primitivas internas de controle;
- os operadores abstratos, organizados segundo hierarquias de herança e composição, que descrevem tarefas complexas a serem executadas e as alternativas de execução delas; e
- as regras de inferência que levam à execução de tarefas.

Para o cenário abordado, a ARE é o principal objeto que é manipulado, sendo representado por inúmeras instâncias de ARE, tantas quanto forem os agendamentos tratados. Para efetuar uma ARE (ato de efetuar a marcação de uma reunião com data, local e participantes específicos), a empresa deve notificar com antecedência para a

agência de viagens os detalhes da reunião. Quanto maior for o tempo até a realização da reunião, maior será a possibilidade de reservas de recursos necessários para que a reunião possa ocorrer com todos os seus participantes.

A notificação de uma ARE ocorre a partir da inclusão, no Repositório de Conhecimento, da seguinte definição:

are(codigo, data, local, prioridade, qtd_func, [lista_func])

- *Codigo* – Identificador único de uma ARE;
- *Data* – Data da reunião;
- *Local* – Local da reunião;
- *Prioridade* - A prioridade da reunião deve ser informada utilizando um número inteiro, quanto menor for o número maior será a prioridade da reunião;
- *Qtd_func* – Quantidade de funcionários da reunião;
- *[lista_func]*- Lista de funcionários que deverão participar da reunião.

Para o cenário proposto, a criação de uma ARE leva à inserção no repositório de conhecimento de fatos como os descritos no código 5-1 a seguir:

```
are(1,'28/09/2010','vitoria',2,2,['gilberto','alberto']).
estado_are(1,'criada').
```

Código 5-1 Fatos para criação de uma ARE

A primeira linha descreve uma nova ARE com código *1*, data *28/09/2010*, a ser realizada em *Vitória*, com prioridade *2* e *2* funcionários: *Gilberto* e *Alberto*. A segunda linha do exemplo descreve que a ARE com código igual a *1* encontra-se com o estado atual *'criada'*. O serviço de agendamento detectará, em tempo de execução, a existência de uma nova ARE e deverá iniciar o processo de verificação de regras de inferência. Assim sendo, foram configuradas quatro regras de inferência distintas para demonstrar as capacidades do ambiente SACCAS dentro do exemplo descrito:

- Regra de inferência 1 – Verificação de uma ARE;
- Regra de inferência 2 – Reserva de uma ARE;

- Regra de inferência 3 – Tratamento de Conflitos entre AREs;
- Regra de inferência 4 – Cancelamento de uma ARE;

5.3.1. Descrição da Regra de Inferência 1 – Verificação de uma ARE

A regra de inferência 1 possui o objetivo de acionar a tarefa abstrata de analisar a estrutura da ARE e recolher informações sobre os recursos necessários para satisfazê-la. Para a geração de uma composição de serviços que faça a análise, o planejador utiliza operadores definidos na hierarquia de tarefas. É preciso verificar se o local da ARE é válido e, como uma ARE envolve vários funcionários, é preciso criar mecanismos dentro da hierarquia de operadores que verifiquem o número de funcionários e incluam no plano o tratamento de cada um. Isso é feito aproveitando-se o fato de a composição de operadores poder ser recursiva, ou seja, um operador que trata uma lista de funcionários pode ser especificado para tratar um funcionário e chamar recursivamente o tratamento da lista de funcionários sem esse primeiro funcionário. O código 5-2 apresenta o operador que exemplifica esse método, gerando as consultas para *n* funcionários quaisquer:

```
operator(1260, /*ID*/
  gerar_consultas_are_n(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
  [],[],
  [ local(DESTINO),consultas_are_a_gerar(ARE,NUM_FUNC) /* PRECONDS */],
  [],[] ,10, [],
  [
    (f1,gerar_consultas_are_1(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,[HFUNC])),
    (f2,gerar_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC2,TFUNCS))
  ], /*SUBTASKS*/
  [ (f1,f2)]):- /*ORDER*/
  NUM_FUNC > 1, NUM_FUNC2 is NUM_FUNC -1,FUNCS = [HFUNC|TFUNCS].
```

Código 5-2 Operador gerar_consultas_are_n

O operador *gerar_consultas_are_n* é decomposto em um operador que gera consultas para um funcionário e outro que gera consultas para uma lista de funcionários, o qual pode ser especializado em um operador que gera a consulta para um funcionário ou para *n* funcionários, dependendo do tamanho da lista.

Cada funcionário presente na lista de participantes não pode possuir outro compromisso na mesma data (outras AREs). Não havendo impedimentos, é preciso consultar, através do Serviço Web da companhia aérea, as possibilidades de voos existentes entre o local de trabalho do funcionário e o local da reunião, retornando uma lista de possibilidades, de acordo com a disponibilidade de voos. Em seguida, é preciso realizar consultas à agência de hotéis através do Serviço Web, passando, como parâmetro, local e data, para verificar disponibilidade de hotel retornando uma lista de hotéis se houver disponibilidade. O código 5-3 apresenta o operador abstrato que gera consultas para um funcionário:

```
operator(1250, /*ID*/
  gerar_consultas_are_1(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,[FUNC]), /*TASK*/
  [], [], [ local(DESTINO) ], /*PRECONDS*/
  [], [], 10, [],
  [  (f1,gerar_consultas_are_status(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNC)),
     (f2,consultar_voos(ARE,FUNC,ORIGEM, DESTINO, DATA, NUM_ALTS_VOOS, ALTS_VOOS)),
     (f3,consultar_hoteis(ARE,FUNC, DESTINO, DATA, NUM_ALT_HOT, ALTS_HOT))
  ],
  [ (f1,f2),(f2,f3)]  /* ORDER */)
```

Código 5-3 Operador gerar_consultas_are_1

O operador abstrato *gerar_consultas_are_1* possui outros operadores, cada um responsável por uma tarefa específica. *Gerar_consultas_are_status* atualiza o estado da tarefa de consulta de um funcionário e os operadores *consultar_voos* e *consultar_hoteis* são respectivos as operações dos Serviços Web da Companhia Aérea e do Hotel, respectivamente. Os operadores são executados seguindo a ordem definida em *ORDER* conforme apresenta o código 5-3.

Para qualquer uma das chamadas de Serviços Web pode haver problemas distintos, como por exemplo, retornar a indisponibilidade de voos ou hotéis. A composição que será gerada para tratar a tarefa acionada pela regra de inferência estará apta para o tratamento dessas situações. Note que o tratamento precisa ser feito para qualquer problema que impeça a participação de algum funcionário na ARE.

Como as regras de inferência necessitam que sejam definidas condições de ativação para serem acionadas, a regra de inferência 1 será executada se, e somente se,

a ARE possuir o estado inicial de *criada* e não houver outras AREs a serem canceladas. Dessa forma, é possível ainda criar prioridades entre as regras de inferência através da dependência de estados. A seguir é apresentado o código 5-4 onde esta regra de inferência é definida:

```
politica_db(

        [
                estado_are(ARE,'criada'),
                not(are_a_ser_cancelada(_)),
                not(func_nao_disponivel_are(ARE,_, _, _, _))
        ],
        verificar_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)
):-are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),nl,writeln('VERIFICA REGRA 1 ').
```

Código 5-4 Regra de inferência 1 - Verificação de uma ARE

Conforme é descrito no código 5-4, a tarefa *verificar_are*, somente é acionada se as três condições apresentadas forem satisfeitas.

5.3.2. Descrição da Regra de Inferência 2 – Reserva de uma ARE

A regra de inferência 2 possui a condição de a ARE ter sido previamente verificada e não haver outras a serem canceladas. Quando acionada, executa a tarefa abstrata de reserva de uma ARE, que deve efetuar, para cada funcionário da lista de participantes, a reserva de um trajeto (com todos os voos) e de um hotel. Note que, entre a verificação feita anteriormente e o momento em que reservas de voo e de hotel são feitas, pode haver mudanças (por exemplo, pode-se não conseguir reservar um hotel que estava disponível). O planejamento, nessa etapa, pode considerar as alternativas de trajeto e de hotel levantadas previamente. Consegue-se, desse modo, aumentar as chances de sucesso.

Para a reserva de voo, o SACCAS identifica a lista de voos consultados anteriormente e tenta reservar um deles chamando o Serviço Web da companhia aérea. Cada voo da lista pode ser direto, entre a origem e destino, ou indireto, quando existem escalas. Independente de ser direto ou existirem escalas, a reserva de todo o trajeto é realizada de uma só vez, sendo responsabilidade do Serviço Web da

companhia aérea garantir que todos os trechos serão marcados para voos indiretos. A marcação de uma passagem é realizada em um único sentido. Dessa forma, a marcação de um voo para a volta do funcionário deveria ser requisitada separadamente. Por questões de simplificação do cenário, não está sendo tratada a marcação de voo do funcionário para a sua cidade de origem (sentido de volta).

Uma vez que a reserva de um voo tenha sido realizada, tenta-se reservar uma opção de hotel, invocando os Serviços Web da agência de hotel. Se o voo ou hotel de algum funcionário não for concretizado, toda a ARE deve ser cancelada. O código 5-5 apresenta a regra de inferência.

```
politica_db(
        [
                are_verificada_com_sucesso(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                not(are_a_ser_cancelada(_))
        ],
        reservar_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)
):-are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),nl,writeln('VERIFICA REGRA 2 ').
```

Código 5-5 Regra de inferência 2 – Reserva de uma ARE

A tarefa *reservar_are* somente é acionada se a ARE já tiver sido verificada com sucesso e não haver nenhuma ARE a ser cancelada.

5.3.3. Descrição da Regra de Inferência 3 – Tratamento de Conflitos entre AREs

A regra de inferência 3 está relacionada com a tarefa abstrata de tratamento de conflitos entre AREs. Quando acionada, a tarefa possui a função de definir, entre duas AREs, qual deve ser confirmada e qual deve ser cancelada, utilizando a informação de *prioridade* da ARE. Essa tarefa abstrata não gera nenhuma composição de Serviço Web e sim, uma composição interna, devendo a sua existência à necessidade de garantir a consistência do ambiente. Essa regra de inferência possui como condição de ativação que duas ou mais AREs estejam em conflito. O código 5-6 apresenta a regra de inferência.

```
politica_db(
        [
                problemas_verificacao_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)
        ],
        tratar_conflitos_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)
):-are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),nl,writeln('VERIFICA REGRA 3 ').
```

Código 5-6 Regra de inferência 3 – Tratamento de Conflitos entre AREs

A tarefa *tratar_conflitos_are* somente é acionada se existir alguma ARE no estado *problemas_verificacao_are*. Ao ser acionada verifica a ARE com menor prioridade e atribui o estado *are_a_ser_cancelada.* A outra ARE permanece com o estado corrente.

5.3.4. Descrição da Regra de Inferência 4 – Cancelamento de uma ARE

A regra de inferência 4 está relacionada com a tarefa abstrata de cancelamento de uma ARE, que efetua o papel contrário da tarefa de reserva, ou seja, desfaz tudo o que a tarefa de reserva efetuou. A regra de inferência 4 possui, como condição de ativação, a marcação de uma ARE a ser cancelada. O código 5-7 apresenta a regra de inferência.

```
politica_db(
        [
                are_a_ser_cancelada(ARE)
        ],
        cancelar_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)
):-are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),nl,writeln('VERIFICA REGRA 4 ').
```

Código 5-7 Regra de inferência 4 – Cancelamento de uma ARE

A tarefa *cancelar_are* somente é acionada se a ARE estiver com o estado definido para cancelamento *are_a_ser_cancelada.*

A tarefa abstrata *cancelar_are* é constituída por diversos operadores que possuem a capacidade de identificar os cancelamentos a serem realizados. Por exemplo, se a ARE possui dois funcionários com voos e hotéis reservados, para cada

um será gerado um plano para liberar todos os recursos de cada funcionário. Por outro lado, se a ARE possui dois funcionários e apenas um com recursos reservados, o plano gerado deverá possuir tarefas apenas de cancelamento de recursos para o funcionário que possui reservas. Portanto, a tarefa abstrata *cancelar_are* possui a capacidade de gerar planos específicos para qualquer combinação de estados possíveis que se encontrarem as reservas de um grupo de funcionários.

5.3.5. Funcionamento das Regras de inferência

Para cada tarefa abstrata, existem outras sub-tarefas abstratas ou primitivas, representadas pelos operadores, organizados em uma hierarquia de abstração e de composição, os quais efetuam o papel de representar e expressar as possibilidades e métodos para se atingir partes de um objetivo.

A tarefa de verificação de uma ARE (regra de inferência 1) leva, com sua decomposição em subtarefas, a duas tarefas primitivas externas, representadas pelos operadores: *consultar_voos e consultar_hoteis.* O operador *consultar_voos* está associado à operação *retonarVoosDisponíveis* do Serviço Web da companhia aérea e o operador *consultar_hoteis* está associado à operação *retonarHoteisDisponíveis* do Serviço Web da agência de hotel. Ambos os operadores preveem o tratamento de duas possibilidades de efeitos não-determinísticos: a existência de uma lista de opções ou nenhuma opção disponível. Para efeitos de exemplificação, o código 5-8 apresenta o operador *consultar_voos*.

```
operator(1280, /*ID*/
        consultar_voos(ARE,FUNC,ORIGEM, DESTINO, DATA, NUM_ALTS_VOOS, ALTS_VOOS),
        [ORIGEM,DESTINO,DATA], /*INPUT*/
        [NUM_ALTS_VOOS,ALTS_VOOS], /*OUTPUT*/
        [ /*PRECONDS*/
         local(ORIGEM), local(DESTINO), funcionario(FUNC)
        ],
        [ /*EFFECTS*/
        trajetos_planejados(ARE,FUNC,ORIGEM, DESTINO, DATA, NUM_ALTS_VOOS)
        ],
        [ /*NDET_EFFECTS*/
         ((NUM_ALTS_VOOS >= 1),
                [alternativa_trajeto(ARE, FUNC, NUM_ALTS_VOOS, ALTS_VOOS)     ]),
         ((NUM_ALTS_VOOS < 1),
                [alternativa_trajeto_inexistente(ARE, FUNC, NUM_ALTS_VOOS, ALTS_VOOS)  ])          ],
        0, /*COST*/
        [], /*MAIN_EFFECTS*/
        [], /*SUBTASKS*/
        []):-db(local_trabalho(FUNC,ORIGEM)),write(' 1280'). /*ORDER*/
```

Código 5-8 Operador consultar_voos

O operador *consultar_voos* é responsável por descrever a tarefa primitiva externa não-determinística, consultar vôos disponíveis. Quando o operador *consultar_voos* é executado pelo Módulo de Execução, uma requisição é enviada ao Módulo de Gestão de Serviços com os parâmetros definidos em *input (ORIGEM, DESTINO e DATA)*, que é traduzida na invocação da operação *retornarVoosDisponíveis* pertencente ao Serviço Web *CompanhiaAereaService*.

O MGS deve retornar o resultado da execução da operação *retornarVoosDisponíveis*, mapeando os atributos retornados nos parâmetros definidos em *output (NUM_ALTS_VOOS e ALTS_VOOS). NUM_ALTS_VOOS* descreve o número de alternativas de voos possíveis entre as duas cidades informadas e *ALTS_VOOS* retorna as alternativas de voos. Baseado no conteúdo de *NUM_ALTS_VOOS,* os efeitos não-determinísticos são avaliados. No exemplo, se o número de alternativas de voos for maior do que um, o fato *alternativa_trajeto* é adicionado aos estado corrente. Caso contrário, o fato *alternativa_trajeto_inexistente* é que é inserido.

Dessa forma, enquanto o Módulo de Planejamento leva em consideração as possibilidades descritas pelos operadores no momento de geração do plano, o Módulo de Execução verifica condições correspondentes aos diferentes efeitos não-

determinísticos para definir se a execução deve seguir em um ramo ou outro do plano ou parar, quando nenhuma condição para a continuidade da execução é satisfeita.

O mesmo raciocínio se aplica às tarefas abstratas de reserva de uma ARE e de cancelamento de uma ARE, uma vez que a sua decomposição leva a operadores associados a operações[1] de Serviços Web da agência de hotéis e de companhias aéreas. Esses operadores possuem efeitos não-determinísticos e os planos que os incluem têm ramificações para tratar diferentes situações. Apenas a tarefa abstrata de tratamento de conflitos entre AREs termina gerando planos sem ramificações, pois implica na execução apenas de tarefas primitivas internas que são determinísticas.

No exemplo apresentado neste capítulo, o conjunto de operadores especificados é suficiente para gerar pelo menos um plano para a execução de cada tarefa abstrata associada a uma das 4 regras de inferência listadas. O ambiente, no entanto, pode ser ainda bastante enriquecido se incluídos operadores que fornecem mais alternativas para a execução de partes das tarefas abstratas.

5.4. Testes no Ambiente

Para se testar o protótipo desenvolvido, considerando o cenário descrito, foram efetuados testes diversificados que visavam explorar diferentes aspectos, entre eles: o comportamento do ambiente, a facilidade de extensão e a baixa necessidade de intervenção.

Os testes foram realizados em um notebook *HP Pavilion* dv4-1435dx com 4GB de memória RAM, Sistema Operacional *Windows Vista Home Premium 64bits* com *Service Pack* 2, processador *Intel Core 2* Duo CPU T6500 2x2.10GHz e disco rígido de 300GB. No momento dos testes, a CPU tinha cerca de 10% de utilização e a memória RAM cerca de 50% de ocupação. A mesma máquina foi utilizada para o desenvolvimento de toda a solução.

[1] Serviços Web podem conter um ou mais operações, cada uma ligada a um método que deverá efetuar uma função específica. Assim, os operadores estão associados a uma operação de um Serviço Web.

O servidor de aplicação *Apache-Tomcat* foi instalado na mesma máquina. Dessa forma, os Serviços Web do hotel e da companhia área eram invocados localmente pela composição gerada através do SACCAS, não havendo, assim, latência de rede.

No início dos testes, havia apenas as configurações iniciais descritas na seção anterior, não havendo qualquer tipo de dado armazenado que pudesse influenciar negativamente no desempenho dos testes. Para a aplicação deste estudo de caso, havia 37 estados descritos, que poderiam ser combinados entre si, e 66 operadores subdivididos em 4 tarefas abstratas.

O teste iniciou-se com o agendamento de uma reunião para uma data futura com dois funcionários através da inserção no Repositório de Conhecimento dos fatos (em Prolog) conforme demonstra o código 5-9.

```
1. assert(are(1,'28/09/2010','vitoria',2,2,['gilberto','alberto'])),
2. assert(estado_are(1,'criada')).
```

Código 5-9 Inclusão de fatos relacionados a ARE 1 no RDC

Na primeira linha, é incluída uma nova ARE com o identificador *1*, para a data do dia *28/09/2010*, a ser realizada em *Vitória*, para dois funcionários, *Gilberto* e *Alberto*, com prioridade *2*. A segunda linha descreve que a ARE com o identificador *1*, encontra-se no estado *criada.* O estado inicial continha, além desses fatos, um conjunto de fatos que descreviam informações sobre o domínio (funcionário, local de trabalho e etc.). Como esses fatos são constantes e não são modificados pelos operadores, eles são apresentados no ANEXO III e omitidos das descrições de estado que parecem neste capítulo. A partir da inclusão desses fatos o estado do ambiente é alterado conforme descrito no código 5-10.

```
---------- ESTADO ATUAL DO AMBIENTE SACCAS ----------

are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
estado_are(1,criada)

-----------------------------------------------------------------------------------------------------
```

Código 5-10 Estado após inclusão da ARE 1

O Módulo de Monitoração e Controle identificou o novo estado e iniciou o processo de verificação das regras de inferência, acionando a regra de inferência 1 - verificação de uma ARE, conforme exemplificado no código 5-11.

```
VERIFICAR REGRA 1
CONDICAO.: [estado_are(1,criada),not(are_a_ser_cancelada(_G359)),
                not(func_nao_disponivel_are(1,_G367,_G368,_G369,_G370))]
ACAO.....:      verificar_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])

CONDICAO - OK - EFETUAR PLANEJAMENTO

planner(([],[]), ([ (t2,verificar_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto]))],[]),_G396,_G397,50000)
```

Código 5-11 Acionamento da regra de inferência 1 pelo MMC

Em seguida solicitou a geração de um plano ao Módulo de Planejamento. O plano gerado é apresentado a seguir no código 5-12.

```
-> inicia_dependencias_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
 -> gerar_dependencias_are_status(1,28/09/2010,vitoria,2,2,gilberto)
 -> func_disponivel(1,vitoria,28/09/2010,gilberto,1)
 -> gerar_dependencias_are_status(1,28/09/2010,vitoria,2,1,alberto)
 -> func_disponivel(1,vitoria,28/09/2010,alberto,1)
 -> finaliza_dependencias_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
 -> inicia_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
 -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,2,gilberto)
 -> consultar_voos(1,gilberto,campinas,vitoria,28/09/2010,_G1879,_G1880)
[?][alternativa_trajeto(1,gilberto,_G5696,_G5697)]
  -> consultar_hoteis(1,gilberto,vitoria,28/09/2010,_G5709,_G5710)
  [?][alternativa_hotel(1,gilberto,vitoria,28/09/2010,_G5725,_G5726)]
    -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,1,alberto)
    -> consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G5755,_G5756)
    [?][alternativa_trajeto(1,alberto,_G5769,_G5770)]
      -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5782,_G5783)
      [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5798,_G5799)]
      [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5845,_G5846)]
    [?][alternativa_trajeto_inexistente(1,alberto,_G5890,_G5891)]
      -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5903,_G5904)
      [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5919,_G5920)]
      [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5966,_G5967)]
  [?][alternativa_hotel_inexistente(1,gilberto,vitoria,28/09/2010,_G6013,_G6014)]
    -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,1,alberto)
    -> consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G6043,_G6044)
    [?][alternativa_trajeto(1,alberto,_G6057,_G6058)]
      -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G6070,_G6071)
      [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G6086,_G6087)]
      [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G6133,_G6134)]
    [?][alternativa_trajeto_inexistente(1,alberto,_G6178,_G6179)]
      -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G6191,_G6192)
      [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G6207,_G6208)]
      [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G6254,_G6255)]
[?][alternativa_trajeto_inexistente(1,gilberto,_G5093,_G5094)]
  -> consultar_hoteis(1,gilberto,vitoria,28/09/2010,_G5106,_G5107)
  [?][alternativa_hotel(1,gilberto,vitoria,28/09/2010,_G5122,_G5123)]
    -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,1,alberto)
    -> consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G5152,_G5153)
    [?][alternativa_trajeto(1,alberto,_G5166,_G5167)]
      -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5179,_G5180)
      [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5195,_G5196)]
      [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5242,_G5243)]
    [?][alternativa_trajeto_inexistente(1,alberto,_G5287,_G5288)]
      -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5300,_G5301)
      [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5316,_G5317)]
      [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5363,_G5364)]
 [?][alternativa_hotel_inexistente(1,gilberto,vitoria,28/09/2010,_G5410,_G5411)]
    -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,1,alberto)
    -> consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G5440,_G5441)
    [?][alternativa_trajeto(1,alberto,_G5454,_G5455)]
      -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5467,_G5468)
      [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5483,_G5484)]
      [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5530,_G5531)]
    [?][alternativa_trajeto_inexistente(1,alberto,_G5575,_G5576)]
      -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5588,_G5589)
      [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5604,_G5605)]
      [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5651,_G5652)]
```

Código 5-12 Plano gerado para verificação de uma ARE

O código 5-12 apresenta o plano gerado para a verificação da ARE. O plano está sendo apresentado numa formatação especial, levemente comprimido, onde são suprimidos os "[" "]" referentes as listas em Prolog, sem perder a sua legibilidade. O símbolo -> é utilizado para indicar uma ação e [?] para indicar as alternativas de uma ação não-determinística, como pode ser observado na ação *consultar_voos*. O plano apresenta para as ações não-determinísticas um conjunto de possibilidades, gerando novas ramificações (representado pela endentação), à medida que, novas ações não-determinísticas são encontradas. Os demais planos não serão apresentados nesse capítulo, mas encontram-se disponíveis no ANEXO IV.

Com o plano gerado, o MMC solicitou ao Módulo de Execução a execução do plano, que por sua vez solicitou a invocação dos Serviços web ao Módulo de Gestão de Serviços, conforme é apresentado no código 5-13.

```
EXECUTAR PLANO:

-> ACAO A SER EXECUT: inicia_dependencias_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])

-> ACAO A SER EXECUT: gerar_dependencias_are_status(1,28/09/2010,vitoria,2,2,gilberto)

-> ACAO A SER EXECUT: func_disponivel(1,vitoria,28/09/2010,gilberto,1)

-> ACAO A SER EXECUT: gerar_dependencias_are_status(1,28/09/2010,vitoria,2,1,alberto)

-> ACAO A SER EXECUT: func_disponivel(1,vitoria,28/09/2010,alberto,1)

-> ACAO A SER EXECUT: finaliza_dependencias_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])

-> ACAO A SER EXECUT: inicia_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])

-> ACAO A SER EXECUT: gerar_consultas_are_status(1,28/09/2010,vitoria,2,2,gilberto)

-> ACAO A SER EXECUT: consultar_voos(1,gilberto,campinas,vitoria,28/09/2010,_G1879,_G1880)
<< RETORNO OPERACAO.: [1,[voo(campinas,vitoria,28/9/2010)]]
?? COND.............: [alternativa_trajeto(1,gilberto,_G5696,_G5697)] - PASSOU

-> ACAO A SER EXECUT: consultar_hoteis(1,gilberto,vitoria,28/09/2010,_G5709,_G5710)
<< RETORNO OPERACAO.: [1,[plaza]]
?? COND.............: [alternativa_hotel(1,gilberto,vitoria,28/09/2010,_G5725,_G5726)] - PASSOU

-> ACAO A SER EXECUT: gerar_consultas_are_status(1,28/09/2010,vitoria,2,1,alberto)

-> ACAO A SER EXECUT: consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G46,_G47)
<< RETORNO OPERACAO.: [1,[voo(rio_de_janeiro,vitoria,28/9/2010)]]
?? COND.............: [alternativa_trajeto(1,alberto,_G14,_G15)] - PASSOU

-> ACAO A SER EXECUT: consultar_hoteis(1,alberto,vitoria,28/09/2010,_G27,_G28)
<< RETORNO OPERACAO.: [1,[plaza]]
?? COND.............: [alternativa_hotel(1,alberto,vitoria,28/09/2010,_G16,_G17)] - PASSOU

-> ACAO A SER EXECUT: finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])

-> ACAO A SER EXECUT: finaliza_exec(1,vitoria)
```

Código 5-13 Execução do plano para verificação de uma ARE

O código 5-13 apresenta o caminho do plano seguido de acordo com os resultados alcançados por cada ação. *COND* descreve qual foi a condição satisfeita do plano. Assim, o SACCAS efetuou o planejamento e execução para a verificação da ARE alcançando um estado que continha, além dos fatos listados no estado inicial, o conjunto de fatos descritos no código 5-14.

```
---------- ESTADO ATUAL DO AMBIENTE SACCAS ----------

1.  are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
2.  are_verificada_com_sucesso(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
3.  trajetos_planejados(1,gilberto,campinas,vitoria,28/09/2010,1)
4.  trajetos_planejados(1,alberto,rio_de_janeiro,vitoria,28/09/2010,1)
5.  alternativa_trajeto(1,gilberto,1,[voo(campinas,vitoria,28/9/2010)])
6.  alternativa_trajeto(1,alberto,1,[voo(rio_de_janeiro,vitoria,28/9/2010)])
7.  hoteis_disponíveis(1,gilberto,vitoria,28/09/2010,1)
8.  hoteis_disponíveis(1,alberto,vitoria,28/09/2010,1)
9.  alternativa_hotel(1,gilberto,vitoria,28/09/2010,1,[plaza])
10. alternativa_hotel(1,alberto,vitoria,28/09/2010,1,[plaza])

-----------------------------------------------------------------------------------------------------
```

Código 5-14 Estado após verificação da ARE 1

A linha 1 descreve a ARE originalmente inserida. A linha 2 indica que a verificação foi positiva, permanecendo com o estado *are_verificada_com_sucesso*, ou seja, indica que foi verificado que a marcação da ARE é viável. As linhas 3 e 4 indicam que já foram planejadas com sucesso as alternativas de trajeto para cada funcionário. As linhas 5 e 6 apresentam uma alternativa de trajeto para cada funcionário. No caso, há apenas uma alternativa de voo direto para cada um, pois o último parâmetro, correspondente a uma lista de voos, contém apenas um voo. A existência de outras alternativas de voo levaria à inserção de outros fatos, onde o terceiro argumento seria diferente, pois indica o número da alternativa. As linhas 7 e 8 indicam que há hotéis disponíveis para cada funcionário. As linhas 9 e 10 indicam as alternativas de hotéis.

O Módulo de Monitoração e Controle, nesse momento, acionou a regra de inferência 2 - reserva de uma ARE e repetiu o processo de acionamento do MDP e MDE. Assim, o SACCAS efetuou o planejamento e execução até a reserva dos recursos para os dois funcionários alcançando um estado que contém o conjunto de fatos descritos no código 5-15 (além dos fatos iniciais).

```
---------- ESTADO ATUAL DO AMBIENTE SACCAS ----------

1.  are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
2.  reservas_are_efetuadas(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
3.  reserva_func_gerada(1,vitoria,28/09/2010,1,gilberto)
4.  reserva_func_gerada(1,vitoria,28/09/2010,1,alberto)
5.  reserva_voo_gerada(1,gilberto,campinas,vitoria,28/09/2010,1,[voo(campinas,vitoria,28/9/2010)])
6.  reserva_voo_gerada(1,alberto,rio_de_janeiro,vitoria,28/09/2010,1,[voo(rio_de_janeiro,vitoria,28/9/2010)])
7.  voo_reservado(1,gilberto,28/09/2010,campinas,vitoria,_G7153,143702247)
8.  voo_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,_G7142,1859864389)
9.  hotel_reservado(1,gilberto,28/09/2010,vitoria,_G7131,806528972)
10. hotel_reservado(1,alberto,28/09/2010,vitoria,_G7121,409317609)
11. reserva_hotel_gerada(1,gilberto,vitoria,28/09/2010,1,[plaza])
12. reserva_hotel_gerada(1,alberto,vitoria,28/09/2010,1,[plaza])

---------------------------------------------------------------------------------------------------------
```

Código 5-15 Estado após reserva da ARE 1

A linha 1 descreve a ARE originalmente inserida. A linha 2 indica que a reserva de todos os recursos da ARE 1 foram efetuadas, permanecendo com o estado *reservas_are_efetuadas*. As linhas 3 e 4 indicam que os recursos para cada funcionário foram reservados. As linhas no intervalo entre 5 e 6 indicam que os voos para cada funcionário foram reservados. As linhas 7 e 8 indicam as reservas de voo feitas. As linhas no intervalo entre 9 e 12 indicam fatos que representam que a reserva de hotel foi realizada para cada funcionário e qual foi o hotel reservado.

Nesse momento, a reserva da ARE 1 foi totalmente concluída e o MMC continuou sendo executado, sem o acionamento de nenhuma regra de inferência, por não haver nenhuma situação a ser tratada.

Em seguida, foi realizado um novo agendamento de reunião para a mesma data, com um dos dois funcionários presentes na primeira reunião (*Alberto*). A prioridade da segunda reunião fornecida era maior do que a prioridade da primeira ARE, conforme é descrito no código 5-16.

```
1.  assert(are(2,'28/09/2010','sao_paulo',1,2,['alfredo','alberto'])),
2.  assert(estado_are(2,'criada')).
```

Código 5-16 Inclusão de fatos relacionados a ARE 2 no RDC

Na primeira linha, é incluída uma nova ARE com o identificador *2*, para a data do dia *28/09/2010* a ser realizada em *São Paulo*, para dois funcionários, *Alfredo* e *Alberto*, com prioridade *1*. A segunda linha descreve que, a ARE com o identificador *2*, encontra-se no estado *criada.* É possível notar que *Alberto*, nesse dia, já possui uma reunião agendada, em *Vitória*, porém com uma prioridade mais baixa (igual a *2*).

O Módulo de Monitoração e Controle, nesse momento, acionou a regra de inferência 1 – verificação de uma ARE, pois identificou a existência de uma ARE com o estado *criada* e efetuou todo o processo de verificação, acionando o MDP e o MDE. Ao final, foi gerado um estado que, além dos fatos no estado inicial, continha os fatos apresentados no código 5-17.

```
---------- ESTADO ATUAL DO AMBIENTE SACCAS ----------

1.  are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
2.  are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
3.  estado_are(2,criada)
4.  problemas_verificacao_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
5.  reservas_are_efetuadas(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
6.  func_disponivel_are(2,_G269,sao_paulo,28/09/2010,alfredo)
7.  func_nao_disponivel_are(2,_G260,sao_paulo,28/09/2010,alberto)
8.  reserva_func_gerada(1,vitoria,28/09/2010,1,gilberto)
9.  reserva_func_gerada(1,vitoria,28/09/2010,1,alberto)
10. reserva_voo_gerada(1,gilberto,campinas,vitoria,28/09/2010,1,[voo(campinas,vitoria,28/9/2010)])
11.
reserva_voo_gerada(1,alberto,rio_de_janeiro,vitoria,28/09/2010,1,[voo(rio_de_janeiro,vitoria,28/9/2010
)])
12.  voo_reservado(1,gilberto,28/09/2010,campinas,vitoria,_G187,143702247)
13.  voo_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,_G176,1859864389)
14.  hotel_reservado(1,gilberto,28/09/2010,vitoria,_G165,806528972)
15.  hotel_reservado(1,alberto,28/09/2010,vitoria,_G155,409317609)
16.  reserva_hotel_gerada(1,gilberto,vitoria,28/09/2010,1,[plaza])
17.  reserva_hotel_gerada(1,alberto,vitoria,28/09/2010,1,[plaza])

-----------------------------------------------------------------------------------------------------------
```

Código 5-17 Estado após verificação da ARE 2

O código 5-17 apresenta nas linhas 1 e 2 que as duas AREs estão registradas no SACCAS. As linhas 3 e 4 descrevem que a ARE 2 recém criada está com problemas de verificação. A linha 6 descreve que o funcionário *Alfredo* está disponível para a ARE 2. A linha 7 descreve que o funcionário *Alberto* não está disponível, motivo pelo qual a verificação falhou. As demais linhas descrevem estados que não sofreram alterações.

Em seguida, o Módulo de Monitoração e Controle acionou a regra de inferência 3 – tratamento de conflitos entre AREs, pois verificou os estados conflitantes existentes (descritos nas linhas 4 e 7) e efetuou todo o processo de tratamento alcançando estado que contém o seguinte conjunto de fatos apresentado no código 5-18.

```
---------- ESTADO ATUAL DO AMBIENTE SACCAS ----------

1.  are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
2.  are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
3.  estado_are(2,criada)
4.  are_a_ser_cancelada(1)
5.  reservas_are_efetuadas(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
6.  reserva_func_gerada(1,vitoria,28/09/2010,1,gilberto)
7.  reserva_func_gerada(1,vitoria,28/09/2010,1,alberto)
8.  reserva_voo_gerada(1,gilberto,campinas,vitoria,28/09/2010,1,[voo(campinas,vitoria,28/9/2010)])
9.  reserva_voo_gerada(1,alberto,rio_de_janeiro,vitoria,28/09/2010,1,[voo(rio_de_janeiro,vitoria,28/9/2010)])
10. voo_reservado(1,gilberto,28/09/2010,campinas,vitoria,_G520,143702247)
11. voo_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,_G509,1859864389)
12. hotel_reservado(1,gilberto,28/09/2010,vitoria,_G498,806528972)
13. hotel_reservado(1,alberto,28/09/2010,vitoria,_G488,409317609)
14. reserva_hotel_gerada(1,gilberto,vitoria,28/09/2010,1,[plaza])
15. reserva_hotel_gerada(1,alberto,vitoria,28/09/2010,1,[plaza])

-----------------------------------------------------------------------------------------------------------
```

Código 5-18 Estado após tratamento de conflitos entre AREs

O código 5-18 descreve na linha 4, a existência de um novo fato indicando que a ARE 1 está em estado de cancelamento. Dessa forma, o MMC acionou a regra de inferência 4 – cancelamento de uma ARE, satisfazendo a condição *are_a_ser_cancelada(1)* e efetuou todo o processo de cancelamento da ARE 1 acionando o MDP e o MDE e, invocando todos os Serviços Web necessários para o cancelamento, através do MGS. Foi alcançado então um estado contendo os fatos apresentados no código 5-19.

```
---------- ESTADO ATUAL DO AMBIENTE SACCAS ----------

1.  are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
2.  are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
3.  estado_are(2,criada)
4.  estado_are(1,cancelada)

-----------------------------------------------------------------------------------------------------------
```

Código 5-19 Estado após cancelamento da ARE 1

O código 5-19 descreve na linha 3 que a ARE 2 está em estado inicial de criação e a ARE 1 encontra-se cancelada na linha 4. É possível observar que não existem mais reservas associadas a ARE 1. Assim, o MMC acionou a regra de inferência 1 – verificação de uma ARE, pois verificou que a ARE 2 encontra-se em estado de criação e efetuou todo o processo de verificação, similar ao realizado para a ARE 1, alcançando estado com o conjunto de fatos apresentado no código 5-20.

```
---------- ESTADO ATUAL DO AMBIENTE SACCAS ----------

1.  are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
2.  are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
3.  estado_are(1,cancelada)
4.  are_verificada_com_sucesso(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
5.  alternativa_trajeto(2,alfredo,1,[voo(campinas,sao_paulo,28/9/2010)])
6.  alternativa_trajeto(2,alberto,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)])
7.  trajetos_planejados(2,alfredo,campinas,sao_paulo,28/09/2010,1)
8.  trajetos_planejados(2,alberto,rio_de_janeiro,sao_paulo,28/09/2010,1)
9.  alternativa_hotel(2,alfredo,sao_paulo,28/09/2010,1,[palace])
10. alternativa_hotel(2,alberto,sao_paulo,28/09/2010,1,[palace])
11. hoteis_disponíveis(2,alfredo,sao_paulo,28/09/2010,1)
12. hoteis_disponíveis(2,alberto,sao_paulo,28/09/2010,1)

-----------------------------------------------------------------------------------------------------------
```

Código 5-20 Estado após verificação da ARE 2

O código 5-20 apresenta na linha 4 que a ARE 2 foi verificada com sucesso e nas demais linhas os recursos disponíveis para cada funcionário, exceto a linha 3 que permaneceu com o estado *cancelada* referente a ARE 1. Assim, o MMC acionou a regra de inferência 2 – reserva de uma ARE, pois verificou que a ARE 2 encontrava-se verificada com sucesso e efetuou todo o processo de reserva (similar ao processo de reserva da ARE 1) invocando o MDP e o MDE e, invocando todos os Serviços Web necessários, através do MGS, alcançando o conjunto de estados apresentado na figura 5-21.

```
---------- ESTADO ATUAL DO AMBIENTE SACCAS ----------

1.  are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
2.  are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
3.  estado_are(1,cancelada)
4.  reservas_are_efetuadas(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
5.  reserva_func_gerada(2,sao_paulo,28/09/2010,1,alfredo)
6.  reserva_func_gerada(2,sao_paulo,28/09/2010,1,alberto)
7.  reserva_voo_gerada(2,alfredo,campinas,sao_paulo,28/09/2010,1,[voo(campinas,sao_paulo,28/9/2010)])
8.
reserva_voo_gerada(2,alberto,rio_de_janeiro,sao_paulo,28/09/2010,1,[voo(rio_de_janeiro,sao_paulo,28/9/201
0)])
9.  voo_reservado(2,alfredo,28/09/2010,campinas,sao_paulo,_G5371,1767778097)
10. voo_reservado(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,_G5360,1752514520)
11. hotel_reservado(2,alfredo,28/09/2010,sao_paulo,_G5349,1467633152)
12. hotel_reservado(2,alberto,28/09/2010,sao_paulo,_G5339,1265789362)
13. reserva_hotel_gerada(2,alfredo,sao_paulo,28/09/2010,1,[palace])
14. reserva_hotel_gerada(2,alberto,sao_paulo,28/09/2010,1,[palace])

-------------------------------------------------------------------------------------------
```

Código 5-21 Estado após reserva da ARE 2

O código 5-21 apresenta o estado final dos testes. Como não existem regras de inferência que são ativadas por esses estados alcançados, a partir desse momento, o MMC permaneceu sem o acionamento de novas regras de inferência e o teste foi finalizado.

O teste demonstrou todo o processo de verificação e reserva para a ARE 1. Em seguida, a ARE 2 foi fornecida e o SACCAS verificou o conflito entre os dois agendamentos, informando que o primeiro deveria ser cancelado e o segundo confirmado, baseando-se nas prioridades de cada ARE. Como a regra de inferência de cancelamento precede a regra de inferência de confirmação (uma capacidade da descrição de regras de inferência), o SACCAS invocou os Serviços Web de cancelamento desfazendo as reservas realizadas e, em seguida, efetuou a reserva da segunda reunião e os seus respectivos recursos para cada funcionário.

5.5. Resultados Obtidos

Esse exemplo serviu para demonstrar um ciclo completo da solução, exemplificando a idéia de evolução no tempo e a ação do mecanismo de controle sobre o ambiente, inferindo novas atividades a serem realizadas para manter os

estados permitidos. Todo o processo de execução dos testes, incluindo as atividades de monitoração, planejamento e execução do exemplo, ocorreu em menos de 15 segundos, tempo muito similar a execução de composições geradas manualmente que efetuem atividades compatíveis, consistindo seus estados em disco rígido, enquanto as informações são manipuladas por outros sistemas. Apesar de testes de desempenho não fazerem parte do objetivo do trabalho, para o exemplo utilizado, a tarefa de planejar em tempo real a execução de Serviços Web não-determinísticos foi viável, o que possibilita levar em conta a situação real do ambiente e aumentar a chance de sucesso na busca para alcançar os objetivos.

O ciclo completo resultou num total de geração e execução de sete composições, e dezenas de possibilidades de caminhos de finalização da solução. Os Serviços Web foram chamados num total de 16 vezes, sendo metade das chamadas para os Serviços Web do hotel e a outra metade para os Serviços Web da companhia aérea.

Se o mesmo cenário fosse construído manualmente um grande esforço seria necessário na geração de um código que previsse todas as possíveis situações. Em um contexto complexo, é complicado até mesmo prever todas as situações, o que diminui as chances de sucesso. Da forma que está organizado, operadores abstratos podem oferecer soluções alternativas para resolver problemas menores em várias situações. Com a decomposição dos problemas maiores em menores, alternativas para resolver problemas maiores em várias situações podem ser criadas automaticamente.

Como a solução foi construída utilizando listas de tamanhos variados, um agendamento de uma reunião pode possuir *n* funcionários, o que resultará na consulta de *n* verificações quanto à existência de voos, *n* verificações quanto à existência de hotéis, o mesmo para operações de reserva, cancelamento etc. Além disso, a consulta de voos pode retornar um número diversificado de possibilidades, assim como a consulta de hotéis. A solução automaticamente se adapta ao tamanho variável das listas utilizadas (por exemplo, lista de participantes da reunião), o que facilita a sua utilização em cenários mais complexos, onde o número de possibilidades pode ser consideravelmente maior.

Com relação à monitoração e controle de composições, deve ser ressaltado que a abordagem facilitou a quebra de processos longos em partes menores, tratadas por regras de inferência específicas, o que é fundamental para o desempenho do planejamento em tempo real. A alternativa a isso seria gerar uma única composição que, previamente, analisasse todas as possíveis situações decorrentes do não-determinismo dos serviços. Tal plano tenderia a se tornar muito grande, tornando praticamente inviável a sua geração em tempo de execução. Em outras propostas, planeja-se sem levar em conta o não-determinismo (replanejando em caso de problemas, como no CASCOM) ou planeja-se previamente tentando prever todo o comportamento dos serviços envolvidos e criando-se uma composição para ser executada muitas vezes, como no ASTRO. Na proposta aqui apresentada, planeja-se levando em conta o não-determinismo em tempo real, o que aumenta as chances de sucesso. Por outro lado, para reduzir a complexidade dos planos e consequentemente o tempo para sua geração, quebram-se processos longos em tarefas menores, que são iniciadas pelas regras de inferência quando verificam que a tarefa anterior foi concluída.

6. Conclusões

Tendo em vista a evolução da Web em direção à integração cada vez maior de sistemas, a geração de composições de serviços Web automaticamente vem se tornando um tema de pesquisa cada vez mais relevante. As abordagens descritas na literatura normalmente recorrem ao uso de algoritmos de planejamento para fazer essas composições automáticas, os quais, no entanto, podem ter diferentes características em termos de flexibilidade e eficiência, dependendo dos pressupostos que assumem. A motivação fundamental do trabalho desenvolvido nesta dissertação foi o tratamento de composições automáticas de Serviços Web em cenários não-determinísticos, que tendem a ser mais complexos e difíceis de dominar por completo. Buscaram-se mecanismos para gerar e executar composições em tempo real, levando em conta o não-determinismo.

De uma maneira geral, foi descrito como outras abordagens vêm trabalhando com o assunto, focando ou na geração em tempo real, sem levar em conta o não determinismo no planejamento, ou fazendo o planejamento previamente, levando em conta o não-determinismo, mas tentando prever todas as situações a serem tratadas e sem se apegar à necessidade de eficiência e flexibilidade para a geração em tempo real.

Uma simples composição para solucionar um problema específico, em ambientes controlados, pode ser factível de ser gerada em tempo real. Entretanto, em ambientes não-determinísticos, o planejamento tende a ser mais demorado. Por um lado, gerar planos que não levam em conta contingências pode levar os planos a falharem com maior frequência. Mesmo que se aplique planejamento novamente para tratar os problemas, as chances de sucesso ficam menores do que se problemas pudessem ter sido considerados com antecedência. Por outro lado, levar em conta contingências no longo prazo pode exigir grande esforço computacional para gerar

composições gigantescas, que, na prática, não poderiam ser geradas em tempo real. Conforme descrito, abordagens como a do ASTRO optam por uma geração prévia que é então disponibilizada para ser executada, o que reduz a flexibilidade para a incorporação de novos serviços.

Composições geradas previamente ainda possuem o problema de não levar em consideração o estado corrente no momento em que necessitam ser executadas. Precisam ser genéricas o suficiente para permitir a sua utilização no cenário proposto em vários estados, fazendo com que o seu tamanho seja maior do que o necessário para realizar a atividade.

A arquitetura proposta de composição e monitoração automáticas e contínuas de Serviços Web não-determinísticos permite a geração sob demanda de composições para resolver problemas específicos em ambientes não-determinísticos. Dessa forma, a composição gerada é muito menor do que a que seria gerada previamente se tivesse que tratar todas as contingências a longo prazo, integrando os serviços Web disponíveis. Isso viabiliza a geração em tempo real, dando maior flexibilidade para tratar contingências e permitindo que o planejamento leve em conta as informações disponíveis naquele momento a seu favor.

Ocorrendo problemas não mapeados, onde a composição gerada automaticamente não possua mecanismos para lidar com a situação, os mecanismos de monitoração e controle fornecem mecanismos adicionais para tratar a questão. O mecanismo de monitoração e controle conta com regras de inferência que descrevem o que fazer quando situações distintas ocorrem ou fogem do controle da composição, ativando novas tarefas, cujo planejamento para a execução ocorre em tempo real. Essas tarefas podem ser corretivas, quando, por exemplo, uma composição não teve sucesso no seu desfazimento, ou tarefas adicionais, que devem ser executadas se um determinado estado for alcançado. O mecanismo de monitoração e controle consegue portanto dar um suporte flexível ao acompanhamento das atividades ao longo do tempo. Composições geradas previamente têm menos flexibilidade para tratar os problemas, pois não levam em consideração informações conhecidas e/ou adquiridas do ambiente. Outro aspecto extremamente relevante da abordagem proposta é que, ao usar um processo contínuo de monitoração e controle, processos longos podem ser

quebrados em etapas menores, envolvendo a inferência da tarefa a executar, o planejamento e a execução. Com etapas menores, o planejamento tende a ser mais eficiente (podendo ser feito em tempo real), pois o número de contingências a tratar é menor. As etapas podem então se suceder de modo a concluir um processo complexo. No exemplo apresentado no capítulo 5, o processo de marcação de uma reunião executiva foi quebrado em uma etapa em que são verificadas a viabilidade e as alternativas para a realização e outra onde se tenta efetivamente alocar os recursos e agendar a reunião.

6.1. Principais Contribuições

As principais contribuições do trabalho desenvolvido nesta dissertação foram as seguintes:

- **A proposta de uma nova arquitetura para a composição e monitoração de Serviços Web não-determinísticos**: foi proposta a utilização de planejamento não-determinístico na geração de composições automáticas de Serviços Web para permitir que eventualidades possam ser endereçadas, aumentando assim, as chances de se alcançar um objetivo informado. Na arquitetura proposta, a geração automática de composições deve ocorrer sob demanda em tempo de execução, resultando em composições menores e mais específicas para o problema a ser tratado. Como as condições de uso dos Serviços Web utilizados podem mudar a todo o momento, essa abordagem permite ainda que seja utilizado o Serviço Web mais adequado no momento da execução. Além disso, o processo de monitoração contínua permite acomodar o surgimento contínuo de novas demandas, o tratamento de processos longos e a resolução de inconsistências.

- **Implementação de protótipo baseado na arquitetura**: o protótipo encontra-se em estado funcional e pode ser utilizada em outros trabalhos futuros. Embora o Módulo de Interação com Terceiros não esteja implementado, o seu funcionamento pôde ser simulado, validando o funcionamento do protótipo.

- **Estudo de caso**: foi feito um estudo de caso tratando um contexto fictício de marcação de reuniões executivas de uma empresa. Embora fictício, o exemplo é razoavelmente complexo e próximo de situações reais. Serviu para mostrar a utilidade da arquitetura e para apontar a direção de trabalhos futuros.

- **Exploração de domínio de aplicação para planejador não-determinístico**: este trabalho utilizou o algoritmo de planejamento não-determinístico desenvolvido em SILVA (2010) permitindo demonstrar a sua utilização em um novo cenário, diferentemente para o que havia sido proposto inicialmente. O planejamento para domínios não-determinísticos é um tópico de pesquisa importante dentro da Inteligência Artificial. A aplicação do algoritmo nesta dissertação serviu de base para avaliar a flexibilidade e os limites de eficiência do planejador. Além disso, possíveis melhoramentos no algoritmo poderão ser feitos com base na observação da aplicação do algoritmo em situações mais complexas no próprio contexto do estudo de caso realizado.

- **Frentes novas de pesquisa**: esta dissertação contribuiu com novos problemas encontrados e algumas soluções. Alguns pontos foram simplificados para permitir a pesquisa dentro do escopo de uma dissertação de mestrado, mas como descrito na seção a seguir, os resultados obtidos podem ser usados futuramente como base para outros projetos que se destinem a tratar problemas de composições de Serviços Web.

6.2. Trabalhos Futuros

Um primeiro trabalho futuro é o desenvolvimento do Módulo de Interação com Terceiros e o aumento da complexidade do estudo de caso realizado, para testar os limites de complexidade que a abordagem proposta é capaz de lidar. É importante oferecer mecanismos gráficos de fácil utilização para que os usuários possam especificar operadores, mapeá-los a Serviços Web e organizá-los em uma hierarquia de tarefas. Além disso, há um conjunto de aperfeiçoamentos na própria arquitetura que são importantes. O planejamento, a execução e a monitoração de várias

composições de Serviços Web em paralelo é um ponto que merece investigação em especial. Outro ponto é a aderência aos padrões para a especificação da semântica de Serviços Web, que será possível traduzindo-se as especificações dos serviços (em OWL-S, por exemplo) para a estrutura dos operadores com que o SACCAS trabalha.

Novas capacidades de inferência e planejamento poderiam adicionar recursos interessantes ao SACCAS. Em particular, o próprio planejador não-determinístico usado é apenas parte de um módulo de planejamento para *Storytelling* Interativo, onde objetivos a serem atingidos não precisam ser explicitamente tarefas. O módulo usa planejamento de ordem parcial para criar uma rede de tarefas inicial que é então detalhada pelo algoritmo descrito nesta dissertação. Neste módulo, objetivos são inferidos com mecanismos mais sofisticados de análise (com lógica temporal modal) do histórico dos eventos passados e não apenas do estado corrente. Além disso, o módulo incorpora um mecanismo de tratamento de objetivos mais fracos, especificados como tentativas de estabelecer uma certa situação. A incorporação dessas características na composição de Serviços Web pode diminuir o trabalho de especificação do contexto, pois facilitam a especificação de regras de inferência e permitem que o planejamento não fique preso apenas ao que tiver sido levado em consideração ao se criar a rede de tarefas.

Técnicas de modelagem e construção de workflows poderiam sem aplicadas para auxiliar na geração mais eficiente de composições, como por exemplo, a separação proposta entre workflows abstratos e concretos para a geração de composições abstratas que pudessem ser reutilizadas em tempo de execução para a geração de composições concretas, levando em consideração os Serviços Web disponíveis e informações relevantes.

Por fim, uma frente de trabalho futura é a investigação de como outras abordagens de composição podem acomodar um mecanismo de monitoração e controle contínuo com o planejamento em tempo de execução. Mecanismos de monitoração e controle contínuo podem eventualmente ser compatíveis com algoritmos de planejamento não-determinísticos mais sofisticados como os baseados em verificação de modelos, como no ASTRO, e planejamento baseado em cadeias de *Markov* (CASSANDRA *et al.*, 1994), que levam em consideração modelos

probabilísticos. Dessa forma, a composição de Serviços Web complexos tenderia a ser facilitada.

7. Referencias Bibliográficas

AGARWAL, VIKAS, CHAFLE, GIRISH, MITTAL, SUMIT, et al., 2008, "Understanding approaches for web service composition and execution". In: *Proceedings of the 1st Bangalore Annual Compute Conference*, pp.18-20, Bangalore, India, Jan.

ALVES, ALEXANDRE, ASSAF, ARKIN, ASKARY, SID et al, 2007, "Web Services Business Process Execution Language Version 2.0", *OASIS*, Apr.

AKKIRAJU, R., FARRELL, J., MILLER, J. et al, 2005, Web Service Semantics - WSDL-S. In: Technical report, W3C Member Submission 7.

BARRETO, CHARLTON, BULLARD, VAUGHN, ERL, THOMAS et al, 2007, "Web Services Business Process Execution Language Version 2.0 (Primer)", *OASIS*, May.

BERNERS-LEE, TIM, HENDLER, JAMES, LASSILA, ORA, 2001, "The Semantic Web—A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities". *Scientific American*, pp. 29–37.

BERGENTI, FEDERICO, CÁCERES, CÉSAR, FERNÁNDEZ, ALBERTO et al, 2006, "Context-aware Service Coordination for Mobile e-Health Applications". *First European Conference on EHealth*, Fribourg, Switzerland, 12-13 October.

BERTOLI, PIERGIORGIO, CIMATTI, ALESSANDRO, PISTORE, MARCO et al, 2003, "MBP: a Model Based Planner", In: *International Conference on Automated Planning & Scheduling*, Trento, Italy.

BOOTH, DAVID, HAAS, HUGO, MCCABE, FRANCIS et al, 2004, “Web Services Architecture”, *W3C*, Feb.

BRAY, TIM, PAOLI, JEAN, SPERBERB-MCQUEEN et al, 2008, "Extensible Markup Language (XML) 1.0", W3C. http://www.w3.org/TR/REC-xml/

BRATKO, I. *Prolog Programming for Artificial Intelligence*. Addison-Wesley, Reading, Massachusetts, 1986.

CASSANDRA, A. R., KAELBLING, L. P., LITTMAN, M. L., 1994, “Acting optimally in partially observable stochastic domains”. In *Proceedings of the Twelfth National Conference on Artificial Intelligence, (AAAI)* Seattle, WA.

COURBIS, C., FINFELSTEIN, A., 2004, “Towards an Aspect Weaving BPEL Engine”. In: *Proceedings of the Third AOSD Workshop on Aspects, Components, and Patterns for Infrastucture Software (ACP4IS)*, Lancaster, UK, March 2004.

CHAN, K. S. MAY, BISHOP, JUDITH, BARESI, LUCIANO, 2007, "Survey and Comparison of Planning Techniques for Web Services Composition". In: *ISMED*, University of Pretoria, Pretoria, South Africa.

CHEN, L., SHADBOLT, N.R., GOBLE, C.A., et al., 2003, “Toward a Knowledge-Based Approach to Semantic Service Composition,” *Lecture Notes in Computer Science*, v. 2870, pp. 319-334.

CHINNICI, ROBERTO, MOREAU, JEAN-JACQUES, RYMAN, ARTHUR et al, 2007, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", W3C, Jun.

CLEMENT, LUC, HATELY, ANDREW, RIEGEN, CLAUS VON, 2004, "UDDI Spec Technical Committee Draft". *OASIS.*

DCOM, 2010, “Distributed Component Object Model (DCOM) Remote Protocol Specification”. *Microsoft Corporation*, July 11.

DEY, Anind K., 2001,"Understanding and using context”, Georgia Institute of Technology, v.5, n.1, pp. 4–7,Atlanta, GA, USA.

ERL, THOMAS, 2005, *Service-Oriented Architecture, Concepts, Technology, and Design*. 1 ed. Prentice Hall.

FARRELL, J., LAUSEN, H., 2007, "Semantic Annotations for WSDL and XML Schema", W3C. http://www.w3.org/TR/sawsdl/.

FEUERLICHT, GEORGE, 2006, "Service Aggregation Using Relational Operations on Interface Parameters". In: *4th International of Conference Service-Oriented Computing*, pp. 95-103, Chicago, IL, USA.

GANNOD, GERALD C., URBAN, SUSAN D., BURGE, JANET E., 2007, "Issues in the Design of Flexible and Dynamic Service-Oriented Systems". In: *Proceedings of the ICSE Software Development for Service-Oriented Architectures Workshop*.

GHALLAB, M., NAU, D.S., TRAVERSO, P. 2004, *Automated Planning: theory and practice*. Morgan Kaufmann Publishers.

GOTTSCHALK, KARL, 2000, "Web Services architecture overview: The next stage of evolution for e-business", IBM. http://www.ibm.com/developerworks/library/w-ovr/?dwzone=ws

GUDGIN, MARTIN, HADLEY, MARC, MENDELSOHN, NOAH, 2007, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", *W3C*.

HARNEY, JOHN, DOSHI, PRASHANT, 2008, "Speeding Up Web Service Composition with Volatile External Information", In: *Proceeding of the 17th international conference on World Wide Web table of contents*, Beijing, China.

HOFFMANN, J., NEBEL, B., 2001, "The FF Planning System: Fast Plan Generation Through Heuristic Search", v. 14, pp. 253-302.

IRANI, ROMIN, 2001, "Practical considerations in implementing web services," http://www.webservicesarchitect.com/content/articles/irani01.asp

JENNINGS, N. R., FARATIN, P., NORMAN, T. J., et al. 2000, "Autonomous agents for business process management". *Journal of Applied Artificial Intelligence*, v.14, n. 2, pp. 145-189.

JOSUTTIS, NICOLAI M, 2007, SOA in Practice, 1 ed., Sebastopol, O'Reilly.

KLUSCH, MATTHIAS, GERBER, ANDREAS, PEREIRA, HELENA et al., 2005, *CASCOM: Context-Aware Business Application Service Co-ordination in Mobile Computing Environments*. In Report deliverable D3.2: Conceptual Architecture Design.

KLUSH, M., GERBER, A., SCHMIDT, M., 2005, "Semantic Web Service Composition Planning with OWLS-XPlan". In: *Proceedings of the AAAI Fall Symposium on Semantic Web and Agents*, Arlington VA, USA, AAAI Press.

LASSILA, ORA, SWICK, RALPL R., 1999, "Resource Description Framework (RDF) Model and Syntax Specification", W3C. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

LEYMANN, F., 2001, "Web Services Flow Language (WSFL)", *IBM Corp*.

LU, Z., GHOSE, A. K., HYLAND, P. et al., 2006, "Using assumptions in service composition context". In: *Proceedings of the ICSE-2006 Workshop on Service-Oriented Software Engineering*, Shanghai, China.

LONG, D. D. E., MUIR, A., ANDGOLDING, R.A., 1995, "A longitudinal survey of internet host reliability". In: *Proceedings of the 14th Symposium on Reliable Distributed Systems*, pp. 2–9.

MARCONI, ANNAPAOLA, PISTORE, MARCO, TRAVERSO, PAOLO, 2008, "Automated Composition of Web Services: the ASTRO Approach", IEEE Data Eng. Bull, v.31, n.3.

MCGUINNESS, DEBORAH L., HARMELEN, FRANK VAN, 2004, "OWL Web Ontology Language Overview", *W3C*, Feb.

MCILRAITH, S. A., SON, T. C., ZENG, H. 2001. "Semantic web services". *IEEE Intel. Syst*. v. 16, n. 2, pp. 46–53.

MENDLING, M. HAFNER., 2005, "From WS-CDL Choreography to BPEL Process Orchestration", *Journal of Enterprise Information Management (JEIM)*, v. 21, n. 5, pp. 525-542.

MILANOVIC, N., MALEK, M, 2004, “Current solutions for web service composition”, *IEEE Internet Comput,* v.8, n. 6(Dec), pp. 51–59.

NAU, D. S., AU, T. C., Ilghami, O. et al, 2003, "SHOP2: An HTN Planning System", *Journal of Artificial Intelligence Research,* v. 20, pp. 379-404.

PAPAZOGLOU, MIKE P., HEUVEL, WILLEM-JAN VAN DEN, 2007, “Service oriented architectures: approaches, technologies and research issues”. *The VLDB Journal,* pp. 389-415.

RAO, JINGHAI, SU, XIAOMENG, 2004, "A Survey of Automated Web Service Composition Methods". In: *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition*, San Diego, CA, USA.

ROSS-TALBOT, STEVE, FLETCHER, TONY, 2006, "Web Services Choreography Description Language: Primer", W3C. http://www.w3.org/TR/2006/WD-ws-cdl-10-primer-20060619/

SCHROEDER, B., GIBSON, G. A., 2006, “A large-scale study of failures in high-performance computing systems”. In: *Proceedings of the International Conference on Dependable Systems and Networks, pp.* 249–258.

SHENG, QUAN Z., BENATALLAH, BOUALEM, DUMAS, MARLON et al, 2002, “SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment”. *International Conference on Very Large Databases (VLDB).* Hong Kong, China, September.

SILVA, FABIO A. G., 2010, *Geração Não-Determinística de Enredos para Storytelling Interativo*. M.Sc. dissertation, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, Rio de Janeiro, Br.

SIRIN, EVREN, HENDLER, JAMES, PARSIA, BIJAN, 2003, "Semi-automatic Composition of Web Services using Semantic Descriptions" In: *Web Services: Modeling, Architecture and Infrastructure workshop*, Angers, France.

SIRIN, EVREN, PARSIA, BIJAN, WU, DAN, et al., 2004, "HTN planning for Web Service composition using SHOP2," *Web Semantics: Science, Services and Agents on the World Wide Web,* pp. 377-396.

SRIVASTAVA, BIPLAV, KOEHLER, JANA, 2003, "Web Service Composition - Current Solutions and Open Problems", In: *ICAPS 2003 Workshop on Planning for Web Services*, Trento, Italy.

TATE, AUSTIN, HENDLER, JAMES, DRUMMOND, MARK, 1990, "A Review of AI Planning Techniques", *AI Magazine*, v. 11, n. 2. pp. 61-77.

THATE, S., 2001, "XLANG: Web Services for Business Process Design", *Microsoft Corp*.

WEERAWARANA, S., CURBERA, F., LEYMANN, F., et al. 2005. *Web Services Platform Architecture*. Prentice-Hall, Englewood Cliffs, NJ.

WU, D., PARSIA, B., SIRIN, E., et al, 2003, "Automating DAML-S web services composition using SHOP2", In: *Proceedings of 2nd International Santic Web Conference (ISWC2003),* Sanibel Island, Florida, Oct.

WU, ZIXIN, GOMADAM, KARTHIK, RANABAHU, AJITH et al, 2007, "Automatic Composition of Semantic Web Services Using Process Mediation". *Intl. Conf. on Enterprise Information Systems,* Funchal, Madeira, Portugal, 12-16, June.

YANG, Z., DUDDY, K, 1996. "CORBA: A platform for distributed object computing". ACM Oper, v. 30, n. 2, pp. 4-31.

ANEXO I - Código do Módulo de Execução

```
/**MODULO DE EXECUCAO DO PLANO GERADO**************************************/
/*Percorre todo o plano*/

executa_plano([]) :- !.
executa_plano(PLAN) :-
                        PLAN = (ACAO, CONTINUACAO),
                        escreve_log(('-> ACAO A SER EXECUT: ')),
                        writeln(''),write('-> ACAO A SER EXECUT: '),writeln(ACAO),
                        chamaOperacao(ACAO,OUTPUT,EFFECTS,NDETEFFECTS),
                        atualizaBaseEfeitos(EFFECTS, NDETEFFECTS),
                        continuaexecucao(CONTINUACAO).

continuaexecucao([]):-!.
continuaexecucao(CONTINUACAO):-
                        escolhesubplano(CONTINUACAO,SUBPLAN),
                        executa_plano(SUBPLAN), nl.

escolhesubplano(CONTINUACAO,SUBPLAN):-
                        (
                                (
                                        member([COND,SUBPLAN],CONTINUACAO),
                                        write(''),write('?? COND.....: '),write(COND),
                                        escreve_log(('?? COND.....: ',COND)),
                                        chamalista(COND), writeln(' - PASSOU ')
                                );
                                (
                                        CONTINUACAO = [true, (SUBPLAN)]
                                )
                        ),
                        !.

chamaOperacao(ACAO,OUTPUT,EFFECTS,NDETEFFECTS):-
                        escreve_log(('OPERATOR',ID, ACAO, INPUT, OUTPUT, PRECONDS, EFFECTS, NDETEFFECTS)),
                        operator(ID, ACAO, INPUT, OUTPUT, PRECONDS, EFFECTS, NDETEFFECTS, _, _, _, _),
                        escreve_log(('-+- PRE-CONDICOES...: ',PRECONDS)),
                        chamalista(PRECONDS),!,
                        invocaOperacao(ID,ACAO,INPUT,OUTPUT).

chamalista([]):-!.
chamalista([H|T]):-
                        escreve_log(('CHAMA PRECONDICAO',H)),
                        call(H),
                        chamalista(T).

invocaOperacao(_,_,[_],[_]):-!.
invocaOperacao(ID,OPT,INPUT,OUTPUT):-
                        escreve_log(('>> INVOCA OPERACAO..: ',ID,OPT,INPUT, OUTPUT)),
                        tipoOperacao(ID,TIPO,WS),
                        (
                                (
                                        TIPO = 'Externa' ->     invocaOperacaoWS(WS,INPUT,OUTPUT),
                                        write('<< RETORNO OPERACAO.: '),writeln(OUTPUT)
                                );
```

```
                (
                        TIPO = 'Interna' -> write('<< OPERACAO INTERNA.: '),writeln(ID),
                        escreve_log(('<< OPERACAO INTERNA.: ',ID))
                )
        ).

atualizaBaseEfeitos(EFFECTS, NDETEFFECTS):-
        escreve_log(('+ ATUALIZA BASE EFEI: ')),
        escreve_log(('+ EFEITOS AVALIADOS.: ',EFFECTS)),
        insereEfeitos(EFFECTS),
        escreve_log(('+ EFEITOS NDET. AVAL: ',NDETEFFECTS)),
        insereNDETEfeitos(NDETEFFECTS).

insereEfeitos([]) :- !.
insereEfeitos([H_EFFECTS|T_EFFECTS]):-
        insereumEfeito(H_EFFECTS),insereEfeitos(T_EFFECTS).

insereumEfeito(not(H_EFFECTS)):-
        retract(H_EFFECTS),
        escreve_log(('--- RETIRA EFEITO...: ',H_EFFECTS)),!.
insereumEfeito(H_EFFECTS):-
        assertz(H_EFFECTS),
        escreve_log(('--- INSERE EFEITO...: ',H_EFFECTS)),!.


insereNDETEfeitos([]) :- !.
insereNDETEfeitos(NDET_EFFECTS):-
                member((COND,EF), NDET_EFFECTS),
                escreve_log(('- TESTANDO COND NDET: ',COND)),
                call(COND),
                escreve_log(('- CONDICAO NDET.....: ',COND)),
                insereEfeitos(EF),!.


tipoOperacao(ID,TIPO,WS):-
                ID = 1280  -> TIPO = 'Externa',WS='retornarVoosDisponiveis';
                ID = 1290  -> TIPO = 'Externa',WS='retornarHoteisDisponiveis';
                ID = 3140  -> TIPO = 'Externa',WS='reservarVoo';
                ID = 4140  -> TIPO = 'Externa',WS='reservarHotel';
                TIPO = 'Interna'.

/*Arquivos utilizados para comunicação com o JAVA*/
gerar_entrada_saida(Prefixo,Arquivo_entrada,Arquivo_saida):-
        gerar_nome_arquivo(Prefixo,Arquivo),
        string_concat(Arquivo, '.inp', Arquivo_entrada),
        string_concat(Arquivo, '.out', Arquivo_saida).

gerar_nome_arquivo(Prefixo,Arquivo):-I is random(9999999),
        string_concat(Prefixo, '_', A),
        string_concat(A,I, Arquivo).

invocaOperacaoWS(WS,INPUT,Registro):-  escreve_log(('invocaOperacaoWs',WS,INPUT)),
        gerar_entrada_saida(WS,Arquivo_entrada,Arquivo_saida),
        open(Arquivo_entrada, append, Ponteiro),
        write(Ponteiro, (WS,INPUT)),
        nl(Ponteiro),
        close(Ponteiro),!,
        tenta_ler_registro(Arquivo_saida,Registro),!,
        escreve_log(('Arquivo lido' , Arquivo_saida , Registro)).

tenta_ler_registro(PathIn,Record):- %sleep(1),
        ler_registro(PathIn,Record).

ler_registro(PathIn,Record) :-
        (
                not(exists_file(PathIn)),tenta_ler_registro(PathIn,Record)
        );
        (
                open(PathIn, read, FileIn),
```

```
                %write('Lê arquivo: '), writeln(PathIn),
                read(FileIn, Record),
                %writeln(Record),
                close(FileIn)
        ),!.

escreve_log(A):-
        open('composicao_prolog.log', append, Ponteiro),
        write(Ponteiro, (A)),
        nl(Ponteiro),
        nl(Ponteiro),
        close(Ponteiro).
```

ANEXO II - Código do Módulo de Monitoração e Controle

```
mon:-monitorar(50000).
monitorar(NDET_LEVEL):-     sh,
                            politica_db(CONDICAO,HTNINICIAL),
                            write('CONDICAO.: '),writeln(CONDICAO), write('ACAO.....: '),writeln(HTNINICIAL),
                            chamalista(CONDICAO),!,
                            writeln('Passou pela condição - Vai efetuar o planejamento'),nl,
                            writeln(planner(([], []), ([(t2,  HTNINICIAL)], []), PLANO, _, NDET_LEVEL)),nl,
                            planner(([], []), ([(t2,  HTNINICIAL)], []), PLANO, _, NDET_LEVEL),!,
                            nl,writeln('Executar Plano:'),
                            executa_plano(PLANO),!,
                            sh,!.
```

ANEXO III - Conjunto de Estados Iniciais

```
local('rio_de_janeiro').
local('sao_paulo').
local('vitoria').
local('campinas').
local('campos').
local('caxambu').
local('sao_lourenco').
local('belo_horizonte').
local('ouro_preto').
local('recife').

funcionario('alberto').
funcionario('alfredo').
funcionario('gilberto').
funcionario('gustavo').
funcionario('angelo').
funcionario('sean').
funcionario('augusto').
funcionario('fabio').

local_trabalho('alberto','rio_de_janeiro').
local_trabalho('alfredo','campinas').
local_trabalho('gilberto','campinas').
local_trabalho('gustavo','rio_de_janeiro').
local_trabalho('angelo','campinas').
local_trabalho('sean','caxambu').
local_trabalho('augusto','recife').
local_trabalho('fabio','belo_horizonte').

hotel('palace').
hotel('hilton').
hotel('melia').
hotel('plaza').
hotel('mercure').
hotel('hitz').
hotel('oregon').
```

ANEXO IV - Planos Gerados no Exemplo

Plano Gerado 1 – Verificação ARE 1

```
================================================================================

PLAN:

================================================================================


 -> inicia_dependencias_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
 -> gerar_dependencias_are_status(1,28/09/2010,vitoria,2,2,gilberto)
 -> func_disponivel(1,vitoria,28/09/2010,gilberto,1)
 -> gerar_dependencias_are_status(1,28/09/2010,vitoria,2,1,alberto)
 -> func_disponivel(1,vitoria,28/09/2010,alberto,1)
 -> finaliza_dependencias_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
 -> inicia_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
 -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,2,gilberto)
 -> consultar_voos(1,gilberto,campinas,vitoria,28/09/2010,_G1879,_G1880)
 [?][alternativa_trajeto(1,gilberto,_G5696,_G5697)]
   -> consultar_hoteis(1,gilberto,vitoria,28/09/2010,_G5709,_G5710)
   [?][alternativa_hotel(1,gilberto,vitoria,28/09/2010,_G5725,_G5726)]
     -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,1,alberto)
     -> consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G5755,_G5756)
     [?][alternativa_trajeto(1,alberto,_G5769,_G5770)]
       -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5782,_G5783)
       [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5798,_G5799)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5845,_G5846)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
     [?][alternativa_trajeto_inexistente(1,alberto,_G5890,_G5891)]
       -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5903,_G5904)
       [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5919,_G5920)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5966,_G5967)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
   [?][alternativa_hotel_inexistente(1,gilberto,vitoria,28/09/2010,_G6013,_G6014)]
     -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,1,alberto)
     -> consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G6043,_G6044)
     [?][alternativa_trajeto(1,alberto,_G6057,_G6058)]
       -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G6070,_G6071)
       [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G6086,_G6087)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G6133,_G6134)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
```

```
      [?][alternativa_trajeto_inexistente(1,alberto,_G6178,_G6179)]
        -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G6191,_G6192)
        [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G6207,_G6208)]
          -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
          -> finaliza_exec(1,vitoria)
        [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G6254,_G6255)]
          -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
          -> finaliza_exec(1,vitoria)
[?][alternativa_trajeto_inexistente(1,gilberto,_G5093,_G5094)]
   -> consultar_hoteis(1,gilberto,vitoria,28/09/2010,_G5106,_G5107)
   [?][alternativa_hotel(1,gilberto,vitoria,28/09/2010,_G5122,_G5123)]
     -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,1,alberto)
     -> consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G5152,_G5153)
     [?][alternativa_trajeto(1,alberto,_G5166,_G5167)]
       -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5179,_G5180)
       [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5195,_G5196)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5242,_G5243)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
     [?][alternativa_trajeto_inexistente(1,alberto,_G5287,_G5288)]
       -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5300,_G5301)
       [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5316,_G5317)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5363,_G5364)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
   [?][alternativa_hotel_inexistente(1,gilberto,vitoria,28/09/2010,_G5410,_G5411)]
     -> gerar_consultas_are_status(1,28/09/2010,vitoria,2,1,alberto)
     -> consultar_voos(1,alberto,rio_de_janeiro,vitoria,28/09/2010,_G5440,_G5441)
     [?][alternativa_trajeto(1,alberto,_G5454,_G5455)]
       -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5467,_G5468)
       [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5483,_G5484)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5530,_G5531)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
     [?][alternativa_trajeto_inexistente(1,alberto,_G5575,_G5576)]
       -> consultar_hoteis(1,alberto,vitoria,28/09/2010,_G5588,_G5589)
       [?][alternativa_hotel(1,alberto,vitoria,28/09/2010,_G5604,_G5605)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][alternativa_hotel_inexistente(1,alberto,vitoria,28/09/2010,_G5651,_G5652)]
         -> finaliza_consultas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
```

Plano Gerado 2 – Reserva ARE 1

```
================================================================================
PLAN:
================================================================================

 -> inicia_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
 -> efetuar_reservas_are_status(1,28/09/2010,vitoria,2,2,gilberto)
 -> inicia_reservas_voo(1,gilberto,28/09/2010,campinas,vitoria,1,[voo(campinas,vitoria,28/9/2010)],_G623,_G624,_G625)
 -> efetuar_reservas_voo_1(1,gilberto,28/09/2010,campinas,vitoria,1,[voo(campinas,vitoria,28/9/2010)],_G623,_G624,_G625)
 [?][voo_reservado(1,gilberto,28/09/2010,campinas,vitoria,_G7647,_G7648)]
    -> finaliza_reservas_voo(1,gilberto,28/09/2010,campinas,vitoria,1,[voo(campinas,vitoria,28/9/2010)],_G7647,_G7648,_G7665)
    -> inicia_reservas_hotel(1,gilberto,28/09/2010,vitoria,1,[plaza],_G7695,_G7696,_G7697)
    -> efetuar_reservas_hotel_1(1,gilberto,28/09/2010,vitoria,1,[plaza],_G7695,_G7696,_G7697)
```

```
   [?][hotel_reservado(1,gilberto,28/09/2010,vitoria,_G7734,_G7735)]
     -> finaliza_reservas_hotel(1,gilberto,28/09/2010,vitoria,1,[plaza],_G7734,_G7735,_G7751)
     -> efetuar_reservas_are_status(1,28/09/2010,vitoria,2,1,alberto)
     ->
inicia_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G7788,_G7789,_G779
0)
     ->
efetuar_reservas_voo_1(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G7788,_G7789,_
G7790)
     [?][voo_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,_G7839,_G7840)]
       ->
finaliza_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G7839,_G7840,_G7
857)
       -> inicia_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G7887,_G7888,_G7889)
       -> efetuar_reservas_hotel_1(1,alberto,28/09/2010,vitoria,1,[plaza],_G7887,_G7888,_G7889)
       [?][hotel_reservado(1,alberto,28/09/2010,vitoria,_G7926,_G7927)]
         -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G7926,_G7927,_G7943)
         -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][hotel_nao_reservado(1,alberto,28/09/2010,vitoria,plaza)]
         -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8009,_G8010,_G8011)
         -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
     [?][voo_nao_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,voo(rio_de_janeiro,vitoria,28/9/2010))]
       ->
finaliza_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G8089,_G8090,_G8
091)
       -> inicia_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8111,_G8112,_G8113)
       -> efetuar_reservas_hotel_1(1,alberto,28/09/2010,vitoria,1,[plaza],_G8111,_G8112,_G8113)
       [?][hotel_reservado(1,alberto,28/09/2010,vitoria,_G8150,_G8151)]
         -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8150,_G8151,_G8167)
         -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][hotel_nao_reservado(1,alberto,28/09/2010,vitoria,plaza)]
         -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8233,_G8234,_G8235)
         -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
   [?][hotel_nao_reservado(1,gilberto,28/09/2010,vitoria,plaza)]
     -> finaliza_reservas_hotel(1,gilberto,28/09/2010,vitoria,1,[plaza],_G8301,_G8302,_G8303)
     -> efetuar_reservas_are_status(1,28/09/2010,vitoria,2,1,alberto)
     ->
inicia_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G8340,_G8341,_G834
2)
     ->
efetuar_reservas_voo_1(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G8340,_G8341,_
G8342)
     [?][voo_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,_G8391,_G8392)]
       ->
finaliza_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G8391,_G8392,_G8
409)
       -> inicia_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8439,_G8440,_G8441)
       -> efetuar_reservas_hotel_1(1,alberto,28/09/2010,vitoria,1,[plaza],_G8439,_G8440,_G8441)
       [?][hotel_reservado(1,alberto,28/09/2010,vitoria,_G8478,_G8479)]
         -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8478,_G8479,_G8495)
         -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][hotel_nao_reservado(1,alberto,28/09/2010,vitoria,plaza)]
         -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8561,_G8562,_G8563)
         -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
     [?][voo_nao_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,voo(rio_de_janeiro,vitoria,28/9/2010))]
       ->
finaliza_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G8641,_G8642,_G8
643)
       -> inicia_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8663,_G8664,_G8665)
       -> efetuar_reservas_hotel_1(1,alberto,28/09/2010,vitoria,1,[plaza],_G8663,_G8664,_G8665)
       [?][hotel_reservado(1,alberto,28/09/2010,vitoria,_G8702,_G8703)]
         -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8702,_G8703,_G8719)
         -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
```

```
          -> finaliza_exec(1,vitoria)
        [?][hotel_nao_reservado(1,alberto,28/09/2010,vitoria,plaza)]
          -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G8785,_G8786,_G8787)
          -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
          -> finaliza_exec(1,vitoria)
 [?][voo_nao_reservado(1,gilberto,28/09/2010,campinas,vitoria,voo(campinas,vitoria,28/9/2010))]
   -> finaliza_reservas_voo(1,gilberto,28/09/2010,campinas,vitoria,1,[voo(campinas,vitoria,28/9/2010)],_G6480,_G6481,_G6482)
   -> inicia_reservas_hotel(1,gilberto,28/09/2010,vitoria,1,[plaza],_G6502,_G6503,_G6504)
   -> efetuar_reservas_hotel_1(1,gilberto,28/09/2010,vitoria,1,[plaza],_G6502,_G6503,_G6504)
   [?][hotel_reservado(1,gilberto,28/09/2010,vitoria,_G6541,_G6542)]
     -> finaliza_reservas_hotel(1,gilberto,28/09/2010,vitoria,1,[plaza],_G6541,_G6542,_G6558)
     -> efetuar_reservas_are_status(1,28/09/2010,vitoria,2,1,alberto)
     ->
inicia_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G6595,_G6596,_G659
7)
     ->
efetuar_reservas_voo_1(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G6595,_G6596,_
G6597)
      [?][voo_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,_G6646,_G6647)]
        ->
finaliza_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G6646,_G6647,_G6
664)
        -> inicia_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G6694,_G6695,_G6696)
        -> efetuar_reservas_hotel_1(1,alberto,28/09/2010,vitoria,1,[plaza],_G6694,_G6695,_G6696)
        [?][hotel_reservado(1,alberto,28/09/2010,vitoria,_G6733,_G6734)]
          -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G6733,_G6734,_G6750)
          -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
          -> finaliza_exec(1,vitoria)
        [?][hotel_nao_reservado(1,alberto,28/09/2010,vitoria,plaza)]
          -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G6816,_G6817,_G6818)
          -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
          -> finaliza_exec(1,vitoria)
      [?][voo_nao_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,voo(rio_de_janeiro,vitoria,28/9/2010))]
        ->
finaliza_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G6896,_G6897,_G6
898)
        -> inicia_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G6918,_G6919,_G6920)
        -> efetuar_reservas_hotel_1(1,alberto,28/09/2010,vitoria,1,[plaza],_G6918,_G6919,_G6920)
        [?][hotel_reservado(1,alberto,28/09/2010,vitoria,_G6957,_G6958)]
          -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G6957,_G6958,_G6974)
          -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
          -> finaliza_exec(1,vitoria)
        [?][hotel_nao_reservado(1,alberto,28/09/2010,vitoria,plaza)]
          -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G7040,_G7041,_G7042)
          -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
          -> finaliza_exec(1,vitoria)
   [?][hotel_nao_reservado(1,gilberto,28/09/2010,vitoria,plaza)]
     -> finaliza_reservas_hotel(1,gilberto,28/09/2010,vitoria,1,[plaza],_G7108,_G7109,_G7110)
     -> efetuar_reservas_are_status(1,28/09/2010,vitoria,2,1,alberto)
     ->
inicia_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G7147,_G7148,_G714
9)
     ->
efetuar_reservas_voo_1(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G7147,_G7148,_
G7149)
      [?][voo_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,_G7198,_G7199)]
        ->
finaliza_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G7198,_G7199,_G7
216)
        -> inicia_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G7246,_G7247,_G7248)
        -> efetuar_reservas_hotel_1(1,alberto,28/09/2010,vitoria,1,[plaza],_G7246,_G7247,_G7248)
        [?][hotel_reservado(1,alberto,28/09/2010,vitoria,_G7285,_G7286)]
          -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G7285,_G7286,_G7302)
          -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
          -> finaliza_exec(1,vitoria)
        [?][hotel_nao_reservado(1,alberto,28/09/2010,vitoria,plaza)]
          -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G7368,_G7369,_G7370)
          -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
          -> finaliza_exec(1,vitoria)
```

```
    [?][voo_nao_reservado(1,alberto,28/09/2010,rio_de_janeiro,vitoria,voo(rio_de_janeiro,vitoria,28/9/2010))]
      ->
finaliza_reservas_voo(1,alberto,28/09/2010,rio_de_janeiro,vitoria,1,[voo(rio_de_janeiro,vitoria,28/9/2010)],_G7448,_G7449,_G7
450)
      -> inicia_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G7470,_G7471,_G7472)
      -> efetuar_reservas_hotel_1(1,alberto,28/09/2010,vitoria,1,[plaza],_G7470,_G7471,_G7472)
      [?][hotel_reservado(1,alberto,28/09/2010,vitoria,_G7509,_G7510)]
        -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G7509,_G7510,_G7526)
        -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
      [?][hotel_nao_reservado(1,alberto,28/09/2010,vitoria,plaza)]
        -> finaliza_reservas_hotel(1,alberto,28/09/2010,vitoria,1,[plaza],_G7592,_G7593,_G7594)
        -> finaliza_reservas_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
```

Plano Gerado 3 – Verificação ARE 2

```
================================================================================
PLAN:
================================================================================

 -> inicia_dependencias_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
 -> gerar_dependencias_are_status(2,28/09/2010,sao_paulo,1,2,alfredo)
 -> func_disponivel(2,sao_paulo,28/09/2010,alfredo,1)
 -> gerar_dependencias_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
 -> func_nao_disponivel(2,sao_paulo,28/09/2010,alberto,-1)
 -> finaliza_dependencias_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
 -> processa_consultas_are_nao_permitida(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
 -> finaliza_exec(2,sao_paulo)
```

Plano Gerado 4 – Tratamento de Conflitos entre AREs

```
================================================================================
PLAN:
================================================================================

 -> acao_confimacao_are(2,1)
 -> acao_cancelamento_are(2,1)
 -> limpar_efeitos_conflitos_are(2)
```

Plano Gerado 5 – Cancelamento ARE 1

```
================================================================================
PLAN:
================================================================================

 -> iniciar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
 -> efetuar_cancelamentos_are_status(1,28/09/2010,vitoria,2,2,gilberto)
 -> existe_voo_a_ser_cancelado(1,gilberto,28/09/2010,_G633,vitoria,_G635,_G636,_G637)
 [?][voo_cancelado(1,gilberto,28/09/2010,_G4089,vitoria,_G4091,_G4092)]
   -> existe_hotel_a_ser_cancelado(1,gilberto,28/09/2010,vitoria,_G4104,806528972,_G4106)
   [?][hotel_cancelado(1,gilberto,28/09/2010,vitoria,_G4121,806528972)]
     -> limpar_efeitos_cancelamento_are(1,gilberto)
     -> efetuar_cancelamentos_are_status(1,28/09/2010,vitoria,2,1,alberto)
     -> existe_voo_a_ser_cancelado(1,alberto,28/09/2010,_G4161,vitoria,_G4163,_G4164,_G4165)
```

```
    [?][voo_cancelado(1,alberto,28/09/2010,_G4179,vitoria,_G4181,_G4182)]
      -> existe_hotel_a_ser_cancelado(1,alberto,28/09/2010,vitoria,_G4194,409317609,_G4196)
      [?][hotel_cancelado(1,alberto,28/09/2010,vitoria,_G4211,409317609)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
      [?][hotel_nao_cancelado(1,alberto,28/09/2010,vitoria,_G4270)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
    [?][voo_nao_cancelado(1,alberto,28/09/2010,_G4327,vitoria,_G4329)]
      -> existe_hotel_a_ser_cancelado(1,alberto,28/09/2010,vitoria,_G4341,409317609,_G4343)
      [?][hotel_cancelado(1,alberto,28/09/2010,vitoria,_G4358,409317609)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
      [?][hotel_nao_cancelado(1,alberto,28/09/2010,vitoria,_G4417)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
  [?][hotel_nao_cancelado(1,gilberto,28/09/2010,vitoria,_G4475)]
    -> limpar_efeitos_cancelamento_are(1,gilberto)
    -> efetuar_cancelamentos_are_status(1,28/09/2010,vitoria,2,1,alberto)
    -> existe_voo_a_ser_cancelado(1,alberto,28/09/2010,_G4514,vitoria,_G4516,_G4517,_G4518)
    [?][voo_cancelado(1,alberto,28/09/2010,_G4532,vitoria,_G4534,_G4535)]
      -> existe_hotel_a_ser_cancelado(1,alberto,28/09/2010,vitoria,_G4547,409317609,_G4549)
      [?][hotel_cancelado(1,alberto,28/09/2010,vitoria,_G4564,409317609)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
      [?][hotel_nao_cancelado(1,alberto,28/09/2010,vitoria,_G4623)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
    [?][voo_nao_cancelado(1,alberto,28/09/2010,_G4680,vitoria,_G4682)]
      -> existe_hotel_a_ser_cancelado(1,alberto,28/09/2010,vitoria,_G4694,409317609,_G4696)
      [?][hotel_cancelado(1,alberto,28/09/2010,vitoria,_G4711,409317609)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
      [?][hotel_nao_cancelado(1,alberto,28/09/2010,vitoria,_G4770)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
[?][voo_nao_cancelado(1,gilberto,28/09/2010,_G3352,vitoria,_G3354)]
  -> existe_hotel_a_ser_cancelado(1,gilberto,28/09/2010,vitoria,_G3366,806528972,_G3368)
  [?][hotel_cancelado(1,gilberto,28/09/2010,vitoria,_G3383,806528972)]
    -> limpar_efeitos_cancelamento_are(1,gilberto)
    -> efetuar_cancelamentos_are_status(1,28/09/2010,vitoria,2,1,alberto)
    -> existe_voo_a_ser_cancelado(1,alberto,28/09/2010,_G3423,vitoria,_G3425,_G3426,_G3427)
    [?][voo_cancelado(1,alberto,28/09/2010,_G3441,vitoria,_G3443,_G3444)]
      -> existe_hotel_a_ser_cancelado(1,alberto,28/09/2010,vitoria,_G3456,409317609,_G3458)
      [?][hotel_cancelado(1,alberto,28/09/2010,vitoria,_G3473,409317609)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
      [?][hotel_nao_cancelado(1,alberto,28/09/2010,vitoria,_G3532)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
    [?][voo_nao_cancelado(1,alberto,28/09/2010,_G3589,vitoria,_G3591)]
      -> existe_hotel_a_ser_cancelado(1,alberto,28/09/2010,vitoria,_G3603,409317609,_G3605)
      [?][hotel_cancelado(1,alberto,28/09/2010,vitoria,_G3620,409317609)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
        -> finaliza_exec(1,vitoria)
      [?][hotel_nao_cancelado(1,alberto,28/09/2010,vitoria,_G3679)]
        -> limpar_efeitos_cancelamento_are(1,alberto)
        -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
```

```
        -> finaliza_exec(1,vitoria)
   [?][hotel_nao_cancelado(1,gilberto,28/09/2010,vitoria,_G3737)]
     -> limpar_efeitos_cancelamento_are(1,gilberto)
     -> efetuar_cancelamentos_are_status(1,28/09/2010,vitoria,2,1,alberto)
     -> existe_voo_a_ser_cancelado(1,alberto,28/09/2010,_G3776,vitoria,_G3778,_G3779,_G3780)
     [?][voo_cancelado(1,alberto,28/09/2010,_G3794,vitoria,_G3796,_G3797)]
       -> existe_hotel_a_ser_cancelado(1,alberto,28/09/2010,vitoria,_G3809,409317609,_G3811)
       [?][hotel_cancelado(1,alberto,28/09/2010,vitoria,_G3826,409317609)]
         -> limpar_efeitos_cancelamento_are(1,alberto)
         -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][hotel_nao_cancelado(1,alberto,28/09/2010,vitoria,_G3885)]
         -> limpar_efeitos_cancelamento_are(1,alberto)
         -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
     [?][voo_nao_cancelado(1,alberto,28/09/2010,_G3942,vitoria,_G3944)]
       -> existe_hotel_a_ser_cancelado(1,alberto,28/09/2010,vitoria,_G3956,409317609,_G3958)
       [?][hotel_cancelado(1,alberto,28/09/2010,vitoria,_G3973,409317609)]
         -> limpar_efeitos_cancelamento_are(1,alberto)
         -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
       [?][hotel_nao_cancelado(1,alberto,28/09/2010,vitoria,_G4032)]
         -> limpar_efeitos_cancelamento_are(1,alberto)
         -> finalizar_cancelamentos_are(1,28/09/2010,vitoria,2,2,[gilberto,alberto])
         -> finaliza_exec(1,vitoria)
```

Plano Gerado 6 – Verificação ARE 2

```
================================================================================
PLAN:
================================================================================

 -> inicia_dependencias_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
 -> gerar_dependencias_are_status(2,28/09/2010,sao_paulo,1,2,alfredo)
 -> func_disponivel(2,sao_paulo,28/09/2010,alfredo,1)
 -> gerar_dependencias_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
 -> func_disponivel(2,sao_paulo,28/09/2010,alberto,1)
 -> finaliza_dependencias_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
 -> inicia_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
 -> gerar_consultas_are_status(2,28/09/2010,sao_paulo,1,2,alfredo)
 -> consultar_voos(2,alfredo,campinas,sao_paulo,28/09/2010,_G1879,_G1880)
[?][alternativa_trajeto(2,alfredo,_G5696,_G5697)]
   -> consultar_hoteis(2,alfredo,sao_paulo,28/09/2010,_G5709,_G5710)
   [?][alternativa_hotel(2,alfredo,sao_paulo,28/09/2010,_G5725,_G5726)]
     -> gerar_consultas_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
     -> consultar_voos(2,alberto,rio_de_janeiro,sao_paulo,28/09/2010,_G5755,_G5756)
     [?][alternativa_trajeto(2,alberto,_G5769,_G5770)]
       -> consultar_hoteis(2,alberto,sao_paulo,28/09/2010,_G5782,_G5783)
       [?][alternativa_hotel(2,alberto,sao_paulo,28/09/2010,_G5798,_G5799)]
         -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
       [?][alternativa_hotel_inexistente(2,alberto,sao_paulo,28/09/2010,_G5845,_G5846)]
         -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
     [?][alternativa_trajeto_inexistente(2,alberto,_G5890,_G5891)]
       -> consultar_hoteis(2,alberto,sao_paulo,28/09/2010,_G5903,_G5904)
       [?][alternativa_hotel(2,alberto,sao_paulo,28/09/2010,_G5919,_G5920)]
         -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
       [?][alternativa_hotel_inexistente(2,alberto,sao_paulo,28/09/2010,_G5966,_G5967)]
         -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
   [?][alternativa_hotel_inexistente(2,alfredo,sao_paulo,28/09/2010,_G6013,_G6014)]
```

```
    -> gerar_consultas_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
    -> consultar_voos(2,alberto,rio_de_janeiro,sao_paulo,28/09/2010,_G6043,_G6044)
    [?][alternativa_trajeto(2,alberto,_G6057,_G6058)]
      -> consultar_hoteis(2,alberto,sao_paulo,28/09/2010,_G6070,_G6071)
      [?][alternativa_hotel(2,alberto,sao_paulo,28/09/2010,_G6086,_G6087)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
      [?][alternativa_hotel_inexistente(2,alberto,sao_paulo,28/09/2010,_G6133,_G6134)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
    [?][alternativa_trajeto_inexistente(2,alberto,_G6178,_G6179)]
      -> consultar_hoteis(2,alberto,sao_paulo,28/09/2010,_G6191,_G6192)
      [?][alternativa_hotel(2,alberto,sao_paulo,28/09/2010,_G6207,_G6208)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
      [?][alternativa_hotel_inexistente(2,alberto,sao_paulo,28/09/2010,_G6254,_G6255)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
[?][alternativa_trajeto_inexistente(2,alfredo,_G5093,_G5094)]
  -> consultar_hoteis(2,alfredo,sao_paulo,28/09/2010,_G5106,_G5107)
  [?][alternativa_hotel(2,alfredo,sao_paulo,28/09/2010,_G5122,_G5123)]
    -> gerar_consultas_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
    -> consultar_voos(2,alberto,rio_de_janeiro,sao_paulo,28/09/2010,_G5152,_G5153)
    [?][alternativa_trajeto(2,alberto,_G5166,_G5167)]
      -> consultar_hoteis(2,alberto,sao_paulo,28/09/2010,_G5179,_G5180)
      [?][alternativa_hotel(2,alberto,sao_paulo,28/09/2010,_G5195,_G5196)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
      [?][alternativa_hotel_inexistente(2,alberto,sao_paulo,28/09/2010,_G5242,_G5243)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
    [?][alternativa_trajeto_inexistente(2,alberto,_G5287,_G5288)]
      -> consultar_hoteis(2,alberto,sao_paulo,28/09/2010,_G5300,_G5301)
      [?][alternativa_hotel(2,alberto,sao_paulo,28/09/2010,_G5316,_G5317)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
      [?][alternativa_hotel_inexistente(2,alberto,sao_paulo,28/09/2010,_G5363,_G5364)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
  [?][alternativa_hotel_inexistente(2,alfredo,sao_paulo,28/09/2010,_G5410,_G5411)]
    -> gerar_consultas_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
    -> consultar_voos(2,alberto,rio_de_janeiro,sao_paulo,28/09/2010,_G5440,_G5441)
    [?][alternativa_trajeto(2,alberto,_G5454,_G5455)]
      -> consultar_hoteis(2,alberto,sao_paulo,28/09/2010,_G5467,_G5468)
      [?][alternativa_hotel(2,alberto,sao_paulo,28/09/2010,_G5483,_G5484)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
      [?][alternativa_hotel_inexistente(2,alberto,sao_paulo,28/09/2010,_G5530,_G5531)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
    [?][alternativa_trajeto_inexistente(2,alberto,_G5575,_G5576)]
      -> consultar_hoteis(2,alberto,sao_paulo,28/09/2010,_G5588,_G5589)
      [?][alternativa_hotel(2,alberto,sao_paulo,28/09/2010,_G5604,_G5605)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
      [?][alternativa_hotel_inexistente(2,alberto,sao_paulo,28/09/2010,_G5651,_G5652)]
        -> finaliza_consultas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
        -> finaliza_exec(2,sao_paulo)
```

Plano Gerado 7 – Reserva ARE 2

```
================================================================================
PLAN:
```

```
==========================================================================

 -> inicia_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
 -> efetuar_reservas_are_status(2,28/09/2010,sao_paulo,1,2,alfredo)
 -> inicia_reservas_voo(2,alfredo,28/09/2010,campinas,sao_paulo,1,[voo(campinas,sao_paulo,28/9/2010)],_G623,_G624,_G625)
 ->
efetuar_reservas_voo_1(2,alfredo,28/09/2010,campinas,sao_paulo,1,[voo(campinas,sao_paulo,28/9/2010)],_G623,_G624,_G625)
 [?][voo_reservado(2,alfredo,28/09/2010,campinas,sao_paulo,_G7647,_G7648)]
   ->
finaliza_reservas_voo(2,alfredo,28/09/2010,campinas,sao_paulo,1,[voo(campinas,sao_paulo,28/9/2010)],_G7647,_G7648,_G766
5)
   -> inicia_reservas_hotel(2,alfredo,28/09/2010,sao_paulo,1,[palace],_G7695,_G7696,_G7697)
   -> efetuar_reservas_hotel_1(2,alfredo,28/09/2010,sao_paulo,1,[palace],_G7695,_G7696,_G7697)
   [?][hotel_reservado(2,alfredo,28/09/2010,sao_paulo,_G7734,_G7735)]
     -> finaliza_reservas_hotel(2,alfredo,28/09/2010,sao_paulo,1,[palace],_G7734,_G7735,_G7751)
     -> efetuar_reservas_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
     ->
inicia_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G7788,_G778
9,_G7790)
     ->
efetuar_reservas_voo_1(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G7788,_G
7789,_G7790)
     [?][voo_reservado(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,_G7839,_G7840)]
       ->
finaliza_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G7839,_G78
40,_G7857)
       -> inicia_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7887,_G7888,_G7889)
       -> efetuar_reservas_hotel_1(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7887,_G7888,_G7889)
       [?][hotel_reservado(2,alberto,28/09/2010,sao_paulo,_G7926,_G7927)]
         -> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7926,_G7927,_G7943)
         -> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
       [?][hotel_nao_reservado(2,alberto,28/09/2010,sao_paulo,palace)]
         -> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8009,_G8010,_G8011)
         -> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
     [?][voo_nao_reservado(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,voo(rio_de_janeiro,sao_paulo,28/9/2010))]
       ->
finaliza_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G8089,_G80
90,_G8091)
       -> inicia_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8111,_G8112,_G8113)
       -> efetuar_reservas_hotel_1(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8111,_G8112,_G8113)
       [?][hotel_reservado(2,alberto,28/09/2010,sao_paulo,_G8150,_G8151)]
         -> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8150,_G8151,_G8167)
         -> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
       [?][hotel_nao_reservado(2,alberto,28/09/2010,sao_paulo,palace)]
         -> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8233,_G8234,_G8235)
         -> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
   [?][hotel_nao_reservado(2,alfredo,28/09/2010,sao_paulo,palace)]
     -> finaliza_reservas_hotel(2,alfredo,28/09/2010,sao_paulo,1,[palace],_G8301,_G8302,_G8303)
     -> efetuar_reservas_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
     ->
inicia_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G8340,_G834
1,_G8342)
     ->
efetuar_reservas_voo_1(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G8340,_G
8341,_G8342)
     [?][voo_reservado(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,_G8391,_G8392)]
       ->
finaliza_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G8391,_G83
92,_G8409)
       -> inicia_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8439,_G8440,_G8441)
       -> efetuar_reservas_hotel_1(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8439,_G8440,_G8441)
       [?][hotel_reservado(2,alberto,28/09/2010,sao_paulo,_G8478,_G8479)]
         -> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8478,_G8479,_G8495)
         -> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
```

[?][hotel_nao_reservado(2,alberto,28/09/2010,sao_paulo,palace)]
-> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8561,_G8562,_G8563)
-> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
-> finaliza_exec(2,sao_paulo)
[?][voo_nao_reservado(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,voo(rio_de_janeiro,sao_paulo,28/9/2010))]
->
finaliza_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G8641,_G8642,_G8643)
-> inicia_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8663,_G8664,_G8665)
-> efetuar_reservas_hotel_1(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8663,_G8664,_G8665)
[?][hotel_reservado(2,alberto,28/09/2010,sao_paulo,_G8702,_G8703)]
-> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8702,_G8703,_G8719)
-> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
-> finaliza_exec(2,sao_paulo)
[?][hotel_nao_reservado(2,alberto,28/09/2010,sao_paulo,palace)]
-> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G8785,_G8786,_G8787)
-> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
-> finaliza_exec(2,sao_paulo)
[?][voo_nao_reservado(2,alfredo,28/09/2010,campinas,sao_paulo,voo(campinas,sao_paulo,28/9/2010))]
->
finaliza_reservas_voo(2,alfredo,28/09/2010,campinas,sao_paulo,1,[voo(campinas,sao_paulo,28/9/2010)],_G6480,_G6481,_G6482)
-> inicia_reservas_hotel(2,alfredo,28/09/2010,sao_paulo,1,[palace],_G6502,_G6503,_G6504)
-> efetuar_reservas_hotel_1(2,alfredo,28/09/2010,sao_paulo,1,[palace],_G6502,_G6503,_G6504)
[?][hotel_reservado(2,alfredo,28/09/2010,sao_paulo,_G6541,_G6542)]
-> finaliza_reservas_hotel(2,alfredo,28/09/2010,sao_paulo,1,[palace],_G6541,_G6542,_G6558)
-> efetuar_reservas_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
->
inicia_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G6595,_G6596,_G6597)
->
efetuar_reservas_voo_1(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G6595,_G6596,_G6597)
[?][voo_reservado(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,_G6646,_G6647)]
->
finaliza_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G6646,_G6647,_G6664)
-> inicia_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G6694,_G6695,_G6696)
-> efetuar_reservas_hotel_1(2,alberto,28/09/2010,sao_paulo,1,[palace],_G6694,_G6695,_G6696)
[?][hotel_reservado(2,alberto,28/09/2010,sao_paulo,_G6733,_G6734)]
-> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G6733,_G6734,_G6750)
-> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
-> finaliza_exec(2,sao_paulo)
[?][hotel_nao_reservado(2,alberto,28/09/2010,sao_paulo,palace)]
-> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G6816,_G6817,_G6818)
-> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
-> finaliza_exec(2,sao_paulo)
[?][voo_nao_reservado(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,voo(rio_de_janeiro,sao_paulo,28/9/2010))]
->
finaliza_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G6896,_G6897,_G6898)
-> inicia_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G6918,_G6919,_G6920)
-> efetuar_reservas_hotel_1(2,alberto,28/09/2010,sao_paulo,1,[palace],_G6918,_G6919,_G6920)
[?][hotel_reservado(2,alberto,28/09/2010,sao_paulo,_G6957,_G6958)]
-> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G6957,_G6958,_G6974)
-> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
-> finaliza_exec(2,sao_paulo)
[?][hotel_nao_reservado(2,alberto,28/09/2010,sao_paulo,palace)]
-> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7040,_G7041,_G7042)
-> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
-> finaliza_exec(2,sao_paulo)
[?][hotel_nao_reservado(2,alfredo,28/09/2010,sao_paulo,palace)]
-> finaliza_reservas_hotel(2,alfredo,28/09/2010,sao_paulo,1,[palace],_G7108,_G7109,_G7110)
-> efetuar_reservas_are_status(2,28/09/2010,sao_paulo,1,1,alberto)
->
inicia_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G7147,_G7148,_G7149)

```
    ->
efetuar_reservas_voo_1(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G7147,_G
7148,_G7149)
    [?][voo_reservado(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,_G7198,_G7199)]
      ->
finaliza_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G7198,_G71
99,_G7216)
       -> inicia_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7246,_G7247,_G7248)
       -> efetuar_reservas_hotel_1(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7246,_G7247,_G7248)
       [?][hotel_reservado(2,alberto,28/09/2010,sao_paulo,_G7285,_G7286)]
         -> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7285,_G7286,_G7302)
         -> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
       [?][hotel_nao_reservado(2,alberto,28/09/2010,sao_paulo,palace)]
         -> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7368,_G7369,_G7370)
         -> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
    [?][voo_nao_reservado(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,voo(rio_de_janeiro,sao_paulo,28/9/2010))]
      ->
finaliza_reservas_voo(2,alberto,28/09/2010,rio_de_janeiro,sao_paulo,1,[voo(rio_de_janeiro,sao_paulo,28/9/2010)],_G7448,_G74
49,_G7450)
       -> inicia_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7470,_G7471,_G7472)
       -> efetuar_reservas_hotel_1(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7470,_G7471,_G7472)
       [?][hotel_reservado(2,alberto,28/09/2010,sao_paulo,_G7509,_G7510)]
         -> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7509,_G7510,_G7526)
         -> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
       [?][hotel_nao_reservado(2,alberto,28/09/2010,sao_paulo,palace)]
         -> finaliza_reservas_hotel(2,alberto,28/09/2010,sao_paulo,1,[palace],_G7592,_G7593,_G7594)
         -> finaliza_reservas_are(2,28/09/2010,sao_paulo,1,2,[alfredo,alberto])
         -> finaliza_exec(2,sao_paulo)
```

ANEXO V - Operadores Utilizados na Verificação da ARE

```
/**ETAPA 0 - DEFINICAO DE ESPECIALIZACOES**********************************/

/*Operadores específicos para causar loop*/
specialise(gerar_dependencias_are_1(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                        gerar_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)).
specialise(gerar_dependencias_are_n(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                        gerar_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)).

specialise(gerar_consultas_are_1(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                        gerar_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)).
specialise(gerar_consultas_are_n(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                        gerar_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)).



/*Operadores de possibilidades*/

specialise(    func_disponivel(ARE,DESTINO, DATA, FUNC, DISP_OK),
                        verificar_disp_func(ARE,DESTINO, DATA, FUNC, DISP_OK)).
specialise(func_nao_disponivel(ARE,DESTINO, DATA, FUNC, DISP_OK),
                        verificar_disp_func(ARE,DESTINO, DATA, FUNC, DISP_OK)).

specialise( processa_consultas_are_permitida(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                                processa_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)).
specialise( processa_consultas_are_nao_permitida(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                                processa_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)).


/**ETAPA 1 - OPERADORES DE VERIFICACAO DA ARE*******************************/

operator(1000,
                        finaliza_exec(ARE,DESTINO),
                        [],[],
                        [local(DESTINO)],
                        [],[],10,[],[],[]):-write(' 1000').

operator(1100, /*ID*/
                        verificar_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                        [], /*INPUT*/
                        [], /*OUTPUT*/
                        [ /*PRECONDS*/
                                local(DESTINO),
                                estado_are(ARE,'criada'),
                                are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)
                        ],
                        [ /*EFFECTS*/ ],[ /*NDET_EFFECTS*/],
                        10, /*COST*/
                        [], /*MAIN_EFFECTS*/
                        [
                                (f1,            processa_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)),
/*Verifica todos os funcs da ARE*/
```

```
                (f2,            processa_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)),
        /*Consultas para todos os funcs da ARE*/
                (f3, finaliza_exec(ARE,DESTINO))

        ], /*SUBTASKS*/
        [
                (f1,f2),(f2,f3)
        ]):-write(' 1100'). /*ORDER*/


/**ETAPA 1.1 - OPERADORES DE DISPONIBILIDADE DA ARE*******************************/

operator(1110, /*ID*/
        processa_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
        [], /*INPUT*/
        [], /*OUTPUT*/
        [ /*PRECONDS*/
                local(DESTINO),
                are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)
        ],
        [ /*EFFECTS*/
        ],
        [ /*NDET_EFFECTS*/
        ],
        10, /*COST*/
        [], /*MAIN_EFFECTS*/
        [
                (f1, inicia_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)),
                (f2, gerar_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)),
                (f3, finaliza_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS))
        ], /*SUBTASKS*/
        [
                (f1,f2),(f2,f3)
        ]):-write(' 1110'). /*ORDER*/


operator(1120, /*ID*/
        inicia_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
        [], /*INPUT*/
        [], /*OUTPUT*/
        [ /*PRECONDS*/
                local(DESTINO)
        ],
        [ /*EFFECTS*/
                dependencias_are_a_gerar(ARE,NUM_FUNC)
        ],
        [ /*NDET_EFFECTS*/],
        10, /*COST*/
        [], /*MAIN_EFFECTS*/
        [], /*SUBTASKS*/
        []):-write(' 1120'),write(dependencias_are_a_gerar(ARE,NUM_FUNC)). /*ORDER*/


operator(1130, /*ID*/
        finaliza_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
        [], /*INPUT*/
        [], /*OUTPUT*/
        [ /*PRECONDS*/
                %local(DESTINO),
                dependencias_are_a_gerar(ARE,0)
        ],
        [ /*EFFECTS*/
                not(dependencias_are_a_gerar(ARE,0))
        ],
        [ /*NDET_EFFECTS*/],
        10, /*COST*/
        [], /*MAIN_EFFECTS*/
        [], /*SUBTASKS*/
        []):-write(' 1130'). /*ORDER*/
```

```
operator(1140, /*ID*/
                        gerar_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                        [], /*INPUT*/ [], /*OUTPUT*/
                        [ /*PRECONDS*/
                                dependencias_are_a_gerar(ARE,NUM_FUNC)
                        ],
                        [ /*EFFECTS*/ ],
                        [ /*NDET_EFFECTS*/],
                        10, /*COST*/
                        [], /*MAIN_EFFECTS*/ [], /*SUBTASKS*/ []):-write(' 1140'). /*ORDER*/

operator(1150, /*ID*/
                        gerar_dependencias_are_1(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,[FUNC]), /*TASK*/
                        [], /*INPUT*/ [], /*OUTPUT*/
                        [ /*PRECONDS*/
                                %local(DESTINO)
                                dependencias_are_a_gerar(ARE,NUM_FUNC)

                        ],
                        [ /*EFFECTS*/ ],
                        [ /*NDET_EFFECTS*/],
                        10, /*COST*/
                        [], /*MAIN_EFFECTS*/
                        [ /*SUBTASKS*/
                                (f1,gerar_dependencias_are_status(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNC)),
                                (f2,verificar_disp_func(ARE,DESTINO, DATA, FUNC, DISP_OK))
                        ],
                        [               /*ORDER*/
                                (f1,f2)
                        ]):-write(' 1150').  /*ORDER*/


operator(1160, /*ID*/
                        gerar_dependencias_are_n(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                        [], /*INPUT*/ [], /*OUTPUT*/
                        [ /*PRECONDS*/
                                %local(DESTINO),
                          dependencias_are_a_gerar(ARE,NUM_FUNC)
                        ],
                        [ /*EFFECTS*/  ],
                        [ /*NDET_EFFECTS*/],
                        10, /*COST*/
                        [], /*MAIN_EFFECTS*/
                        [
                                (f1,gerar_dependencias_are_1(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,[HFUNC])),
                                (f2,gerar_dependencias_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC2,TFUNCS))
                        ], /*SUBTASKS*/
                        [
                                (f1,f2)
                        ]):-NUM_FUNC > 1,
                                write(' 1160'), %nl, write(dependencias_are_a_gerar(ARE,NUM_FUNC)),
                                NUM_FUNC2 is NUM_FUNC -1,
                                FUNCS = [HFUNC|TFUNCS]. /*ORDER*/

operator(1170, /*ID*/
                        gerar_dependencias_are_status(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNC), /*TASK*/
                        [], /*INPUT*/ [], /*OUTPUT*/
                        [ /*PRECONDS*/
                                %local(DESTINO),
                                dependencias_are_a_gerar(ARE,NUM_FUNC)
                        ],
                        [ /*EFFECTS*/
                                dependencias_are_a_gerar(ARE,NUM_FUNC2),
                                not(dependencias_are_a_gerar(ARE,NUM_FUNC))
                        ],
                        [ /*NDET_EFFECTS*/],
                        10, /*COST*/
                        [], /*MAIN_EFFECTS*/
```

```
                [ /*SUBTASKS*/],
                [/*ORDER*/]):-NUM_FUNC2 is NUM_FUNC -1,write(' 1170'). /*ORDER*/

operator(1180, /*ID*/
                verificar_disp_func(ARE,DESTINO, DATA, FUNC, DISP_OK), /*TASK*/
                [], /*INPUT*/ [], /*OUTPUT*/
                [ /*PRECONDS*/
                        local(DESTINO)
                ],
                [ /*EFFECTS*/],
                [ /*NDET_EFFECTS*/],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/ [], /*SUBTASKS*/ []):-write(' 1180'). /*ORDER*/

operator(1182, /*ID*/
                func_disponivel(ARE,DESTINO, DATA, FUNC, DISP_OK), /*TASK*/
                [], /*INPUT*/ [], /*OUTPUT*/
                [ /*PRECONDS*/
                        local(DESTINO)
                ],
                [ /*EFFECTS*/
                        func_disponivel_are(ARE, ORIGEM, DESTINO, DATA, FUNC)
                ],
                [ /*NDET_EFFECTS*/],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [
                ], /*SUBTASKS*/
                [/*ORDER*/
                ]):-
                        not((db(are(ARE1,DATA1,_,_,_,FUNCS1)),
                           ARE\=ARE1,
                           not(estado_are(ARE1,'cancelada')),
                           DATA=DATA1,
                           member(FUNC, FUNCS1))),
                           DISP_OK = 1,
                           write(' 1182').

operator(1184, /*ID*/
                func_nao_disponivel(ARE,DESTINO, DATA, FUNC, DISP_OK),
                [], /*INPUT*/
                [], /*OUTPUT*/
                [ /*PRECONDS*/
                        local(DESTINO)
                ],
                [ /*EFFECTS*/
                        func_nao_disponivel_are(ARE, ORIGEM, DESTINO, DATA, FUNC)
                ],
                [ /*NDET_EFFECTS*/],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [], /*SUBTASKS*/
                []):-
                         db(are(ARE1,DATA1,_,_,_,FUNCS1)),
                         ARE\=ARE1,
                         not(estado_are(ARE1,'cancelada')),
                         DATA=DATA1,
                         member(FUNC, FUNCS1),DISP_OK = -1,
                         write(' 1184'). /*ORDER*/



/**ETAPA 1.2 - OPERADORES DE CONSULTAS NECESSARIAS PARA A ARE**********************/

operator(1210, /*ID*/
                processa_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                [], /*INPUT*/
                [], /*OUTPUT*/
                [ /*PRECONDS*/
```

```
                        local(DESTINO),
                        are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)
                ],
                [ /*EFFECTS*/
                ],
                [ /*NDET_EFFECTS*/
                ],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [
                ], /*SUBTASKS*/
                [
                ]):-write(' 1210'). /*ORDER*/

operator(1212, /*ID*/
                processa_consultas_are_permitida(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                [], /*INPUT*/
                [], /*OUTPUT*/
                [ /*PRECONDS*/
                        local(DESTINO),
                        are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                        not(func_nao_disponivel_are(ARE, _, _, _, _)),
                        func_disponivel_are(ARE, _, _, _, _)
                ],
                [ /*EFFECTS*/
                ],
                [ /*NDET_EFFECTS*/
                ],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [
                        (f1, inicia_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)),
                        (f2, gerar_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)),
                        (f3, finaliza_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS))
                ], /*SUBTASKS*/
                [
                        (f1,f2),(f2,f3)
                ]):-write(' 1212'). /*ORDER*/

operator(1214, /*ID*/
                processa_consultas_are_nao_permitida(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                [], /*INPUT*/
                [], /*OUTPUT*/
                [ /*PRECONDS*/
                        local(DESTINO),
                        are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                        func_nao_disponivel_are(ARE, _, _, _, _)
                ],
                [ /*EFFECTS*/
                        problemas_verificacao_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS)
                ],
                [ /*NDET_EFFECTS*/
                ],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [
                ], /*SUBTASKS*/
                [
                ]):-write(' 1214'). /*ORDER*/


operator(1220, /*ID*/
                inicia_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                [], /*INPUT*/
                [], /*OUTPUT*/
                [ /*PRECONDS*/
                        local(DESTINO)
                ],
                [ /*EFFECTS*/
```

```
                    consultas_are_a_gerar(ARE,NUM_FUNC)
                ],
                [ /*NDET_EFFECTS*/],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [], /*SUBTASKS*/
                []):-write(' 1220'). /*ORDER*/


operator(1230, /*ID*/
                finaliza_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                [], /*INPUT*/ [], /*OUTPUT*/
                [ /*PRECONDS*/
                    local(DESTINO),
                    consultas_are_a_gerar(ARE,0)
                ],
                [ /*EFFECTS*/
                    are_verificada_com_sucesso(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS),
                    not(consultas_are_a_gerar(ARE,0)),
                    not(estado_are(ARE,'criada'))
                ],
                [ /*NDET_EFFECTS*/],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [], /*SUBTASKS*/
                []):-write(' 1230'). /*ORDER*/

operator(1240, /*ID*/
                gerar_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                [], /*INPUT*/ [], /*OUTPUT*/
                [ /*PRECONDS*/
                    local(DESTINO) /*are(...)*/
                ],
                [ /*EFFECTS*/ ],
                [ /*NDET_EFFECTS*/],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/ [], /*SUBTASKS*/ []):-write(' 1240'). /*ORDER*/

operator(1250, /*ID*/
                gerar_consultas_are_1(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,[FUNC]), /*TASK*/
                [], /*INPUT*/ [], /*OUTPUT*/
                [ /*PRECONDS*/
                    local(DESTINO)
                ],
                [ /*EFFECTS*/ ],
                [ /*NDET_EFFECTS*/],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [ /*SUBTASKS*/
                    (f1,gerar_consultas_are_status(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNC)),
                    (f2,consultar_voos(ARE,FUNC,ORIGEM, DESTINO, DATA, NUM_ALTS_VOOS, ALTS_VOOS)),
                    (f3,consultar_hoteis(ARE,FUNC, DESTINO, DATA, NUM_ALT_HOT, ALTS_HOT))
                ],
                [           /*ORDER*/
                    (f1,f2),(f2,f3)
                ]):-write(' 1250').  /*ORDER*/


operator(1260, /*ID*/
                gerar_consultas_are_n(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNCS), /*TASK*/
                [], /*INPUT*/ [], /*OUTPUT*/
                [ /*PRECONDS*/
                    local(DESTINO),
                  consultas_are_a_gerar(ARE,NUM_FUNC)
                ],
                [ /*EFFECTS*/  ],
                [ /*NDET_EFFECTS*/],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
```

```
                [
                        (f1,gerar_consultas_are_1(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,[HFUNC])),
                        (f2,gerar_consultas_are(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC2,TFUNCS))
                ], /*SUBTASKS*/
                [
                        (f1,f2)
                ]):-NUM_FUNC > 1,
                        %write(NUM_FUNC - FUNCS),nl,
                        NUM_FUNC2 is NUM_FUNC -1,
                        FUNCS = [HFUNC|TFUNCS],write(' 1260'). /*ORDER*/

operator(1270, /*ID*/
                gerar_consultas_are_status(ARE,DATA,DESTINO,PRIORIDADE,NUM_FUNC,FUNC), /*TASK*/
                [], /*INPUT*/ [], /*OUTPUT*/
                [ /*PRECONDS*/
                        local(DESTINO),
                        consultas_are_a_gerar(ARE,NUM_FUNC),
                        func_disponivel_are(ARE, ORIGEM, DESTINO, DATA, FUNC)
                ],
                [ /*EFFECTS*/
                        consultas_are_a_gerar(ARE,NUM_FUNC2),
                        not(consultas_are_a_gerar(ARE,NUM_FUNC)),
                        not(func_disponivel_are(ARE, ORIGEM, DESTINO, DATA, FUNC))
                ],
                [ /*NDET_EFFECTS*/],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [ /*SUBTASKS*/],
                [/*ORDER*/]):-NUM_FUNC2 is NUM_FUNC -1,write(' 1270'). /*ORDER*/


operator(1280, /*ID*/
                consultar_voos(ARE,FUNC,ORIGEM, DESTINO, DATA, NUM_ALTS_VOOS, ALTS_VOOS), /*TASK*/
                [ORIGEM,DESTINO,DATA], /*INPUT*/
                [NUM_ALTS_VOOS,ALTS_VOOS], /*OUTPUT*/
                [ /*PRECONDS*/
                  local(ORIGEM), local(DESTINO), funcionario(FUNC)
                ],
                [ /*EFFECTS*/
                   trajetos_planejados(ARE,FUNC,ORIGEM, DESTINO, DATA, NUM_ALTS_VOOS)
                ],
                [ /*NDET_EFFECTS*/

                        ((NUM_ALTS_VOOS >= 1),
                                [               alternativa_trajeto(ARE, FUNC, NUM_ALTS_VOOS, ALTS_VOOS)   ]),
                        ((NUM_ALTS_VOOS < 1),
                          [   alternativa_trajeto_inexistente(ARE, FUNC, NUM_ALTS_VOOS, ALTS_VOOS)  ])
                ],
                10, /*COST*/
                [], /*MAIN_EFFECTS*/
                [], /*SUBTASKS*/
                []):-db(local_trabalho(FUNC,ORIGEM)),write(' 1280'). /*ORDER*/

operator(1290, /*ID*/
                consultar_hoteis(ARE,FUNC, DESTINO, DATA, NUM_ALT_HOT, ALTS_HOT), /*TASK*/
                [DESTINO, DATA], /*INPUT*/
                [NUM_ALT_HOT,ALTS_HOT], /*OUTPUT*/
                [ /*PRECONDS*/
                  local(DESTINO), funcionario(FUNC)
                ],
                [ /*EFFECTS*/
                  hoteis_disponíveis(ARE,FUNC, DESTINO, DATA, NUM_ALT_HOT)
                ],
                [ /*NDET_EFFECTS*/

                        ((NUM_ALT_HOT >= 1),
                        [alternativa_hotel(ARE,FUNC, DESTINO, DATA, NUM_ALT_HOT, ALTS_HOT) ]),
                        ((NUM_ALT_HOT < 1),
                        [alternativa_hotel_inexistente(ARE,FUNC, DESTINO, DATA, NUM_ALT_HOT, ALTS_HOT) ])
```

```
],
10, /*COST*/
[], /*MAIN_EFFECTS*/
[], /*SUBTASKS*/
[]):-write(' 1290'). /*ORDER*/
```