Geração automática de Modelos de Processo em Sistemas de Workflow

Fabiano Silvério Ribeiro Alves

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Ciência da Computação da Universidade Federal de Uberlândia

Programa de Pós–graduação em Ciência da Computação
Faculdade de Computação
Universidade Federal de Uberlândia
Minas Gerais – Brasil

Fevereiro de 2007

Fabiano Silvério Ribeiro Alves

Geração Automática de Modelos de Processo em Sistemas de Workflow

Dissertação apresentada ao programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Inteligência Artificial.

Orientadora: Profa. Dra. Márcia Aparecida Fernandes

UBERLÂNDIA – MINAS GERAIS – BRASIL
Universidade Federal de Uberlândia - UFU
Fevereiro 2007

Fabiano Silvério Ribeiro Alves

Geração automática de Modelos de Processo em Sistemas de Workflow

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, para obtenção do título de Mestre.

Área de concentração: Inteligência Artificial.

Banca Examinadora:

Uberlândia, 07 de Fevereiro de 2007.

Profa. Márcia Aparecida Fernandes, Dra. – UFU (presidente)

Prof. João Antônio de Vasconcelos, Dr. – UFMG

Profa. Gina Maira Barbosa de Oliveira, Dra. – UFU

Dedicado à

Minha família, minha esposa Fernanda e minha maravilhosa filha Maria Fernanda.

Geração automática de Modelos de Processo em Sistemas de Workflow

Fabiano Silvério Ribeiro Alves

Resumo

Sistemas de Workflow têm sido amplamente utilizados; porém, a geração automática de modelos de processo, especialmente aplicado ao domínio de Web Service é ainda uma área a ser explorada. Problemas de Workflow são difíceis devido ao tamanho das instâncias que geram um grande espaço de busca e exigem muito tempo de processamento. Assim, este problema pode ser tratado com o uso de técnicas de planejamento em IA associado a algoritmos genéticos. A primeira técnica tem muita semelhança com workflow e a segunda é adequada para problemas com grande espaço de busca. Neste contexto, este trabalho apresenta uma arquitetura baseada no uso de um planejador genético para obter a geração automática de modelos para Workflow.

Palavras-chave: Workflow, Planejamento, Planejador Genético, Web Service.

Automatic generation of Models of Process in Workflow Systems

Fabiano Silvério Ribeiro Alves

Abstract

Workflow systems have been widely employed; however, the automatic generation of process models, specially applied to the web service domains is still an area to be explored. Workflow problems are hard due to the size of the instances that generates a wide search space and requires a long time to be processed. Thus, this problem can be coped with the use of AI planning techniques associated with genetic algorithm. The first technique has many similarities with workflow and the second one is adequate for problems with a large search space. In this context, this work presents an architecture based on the use of a genetic planner in order to obtain the automatic generation of models for Workflow.

Keywords: Workflow, Planning, Genetic Planner, Web Service.

Copyright

Agradecimentos

Gostaria de expressar meus sinceros agradecimentos à professora Márcia Aparecida pela orientação, paciência, amizade e pelo conhecimento e experiência transferidos.

Agradecimentos a todos que me deram apoio para a realização deste trabalho, em especial aos meus pais e irmãos que, mesmo distantes, sempre me apoiaram.

Agradeço também aos colegas da UFU com os quais tive o privilégio de conviver nestes anos. Aos Professores do programa de Mestrado e também aos amigos Kairon e Paulo pelas longas jornadas de trabalho e estudo.

E acima de tudo, agradeço a Deus por não deixar faltar força e vontade para que este trabalho se concluísse.

"*As grandes obras são executadas, não pela força, mas pela perseverança*"

J. Jahnson

Sumário

Lista de siglas, acrônimos e abreviaturas

Abreviatura	Detalhes
ADL	*Action Description Language*
AG	*Algoritmo Genético*
API	*Application Programming Interface*
CSCW	*Computer Supported Cooperative Work*
DTD	*Data Type Definition*
FF	*Fast Forward*
FNC	*Fórmula Normal Conjuntiva*
FND	*Fórmula Normal Disjuntiva*
GPS	*General Problema Solver*
HSP	*Heuristic Search Planner*
IA	*Inteligência Artificial*
IDE	*Integrated Development Environment*
LPO	*Lógica de Primeira Ordem*
PDDL	*Planning Domain Definition Language*
PG	*Programação Genética*
POND	*Partially Observable Non-Deterministic Planner*
POP	*Partial Order Planner*
RET	*Representação Explícita de Transições*
RIT	*Representação Implícita de Transições*
RPC	*Remote Procedure Call*
SAT	*SATisfiability*

SCR	*Situated Control Rules*
SGBD	*Sistema Gerência de Banco de Dados*
SGWf	*Sistema Gerência de Workflow*
STRIPS	*Stanford Research Institute Problem Solver*
SOAP	*Simple Object Access Protocol*
TI	*Tecnologia da Informação*
UDDI	*Universal Description, Discovery and Integration*
URL	*Uniform Resource Locator*
Web	*Abreviatura para designar a world wide web (www).*
WAPI	*Workflow Application Programming Interface*
WES	*Workflow Enactment Service*
WfMC	*Workflow Management Coalition*
WPDL	*Workflow Process Definition Language*
WSDL	*Web Services Description Language*
WWW	*Word Wide Web*
XML	*eXtendible Markup Language*
XPDL	*XML Process Definition Language*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A execução de atividades complexas, como por exemplo aquelas relacionadas aos sistemas de produção, a entrega de um serviço ou a geração de conhecimento, exige geralmente a participação de mais de um recurso para atingir o objetivo. Isto é um fato comum dentro de nossa sociedade, em especial nas empresas, onde há negócios ou processos de produção que se caracterizam por serem conjuntos de tarefas que cooperam para atingir o resultado planejado.

A complexidade dos mecanismos do setor produtivo, aliado aos fatores da globalização e suas conseqüências do ponto de vista do aumento da concorrência, fazem com que as empresas busquem cada vez mais metodologia e ferramentas que facilitem o seu trabalho e as mantenham competitivas; gerenciando de forma eficaz suas atividades e proporcionando serviços ou produtos de forma rápida, com custos mais baixos e com mais qualidade e segurança. Isto traz a necessidade de prover a administração e a melhoria dos processos de produção. Neste contexto as ferramentas de Workflow têm um papel importante, desde que estão associadas à automação e à gerência de processos.

Um sistema de workflow pode ser visto a partir de suas funcionalidades que consiste das fases de modelagem de processos, execução e monitoramento. Muitas ferramentas foram desenvolvidas para serem úteis na execução e monitoramento de processos; porém, a geração de modelos de processos, sobretudo automática, permanece uma área onde poucos trabalhos foram desenvolvidos.

Recentemente, tem crescido o interesse em aplicar técnicas de IA na modelagem de processos em Workflow, especialmente as que envolve Planejamento. Planejar em

Inteligência Artificial, significa encontrar a combinação correta de uma seqüencia de ações parametrizadas que conduzem de um estado inicial fornecido para um estado final desejado. Embora o uso desta técnica permita ao usuário economizar muito tempo na modelagem de processos em Sistema de Workflow, há ainda muitas melhorias que podem ser feitas. Aspectos tal como fluxo condicional e fluxo paralelo entre atividades não são muito explorados em trabalhos anteriores e que são importantes quando é considerado a modelagem de processos.

Outra característica importante a ser considerada é que problemas de planejamento são de complexidade reconhecidamente difícil. O espaço de busca, além de ser exponencial, geralmente apresenta um crescimento extremamente rápido, o que torna a sua aplicação em domínio de workflow bastante onerosa devido à diversidade e tamanho das instâncias destes problemas.

Assim, técnicas de Planejamento podem ser associadas com técnicas de Computação Evolutiva, particularmente Algoritmo Genético. A primeira técnica tem muitas semelhanças com workflow e a segunda trata de forma satisfatória problemas com um grande espaço de busca. Outro fator do tratamento de situações genéricas de Planejamento a partir de Algoritmos Genéticos é o relaxamento de restrições de otimização de solução, presentes em vários planejadores atuais, o que facilita a busca pela solução do AG. Os Algoritmos Genéticos realizam uma busca global, enquanto os planejadores realizam esta busca localmente.

Assim, este trabalho tem como objetivo investigar e apresentar uma arquitetura para a geração automática dos modelos de processo em sistema de workflow. A geração está baseada em um planejador genético para identificar e gerar automaticamente modelos de processos de acordo com as atividades disponíveis.

Uma característica importante da arquitetura é a flexibilidade de representação que além de permitir considerar aspectos tal como fluxo condicional e paralelo entre atividades, também possibilita a aplicação em diferentes domínios do conhecimento, como por exemplo o domínio de Web Service. Esta é uma aplicação importante para o problema de workflow, desde que a Internet se tornou uma plataforma comum e global onde recursos são disponíveis como tecnologia de web service. O Workflow de Web Services habilita integração de aplicação interna e aplicação que transpõem as fronteiras organizacionais.

Este trabalho é organizado como segue. O Capítulo 2 discute os conceitos básicos da tecnologia de workflow. O Capítulo 3 discute os conceitos básicos da tecnologia de Planejamento em Inteligência Artificial, apresentando algumas das principais linguagens e abordagens de Planejamento. O Capítulo 4 mostra um mapeamento entre conceitos das áreas de Workflow e Planejamento. Neste capítulo, a arquitetura proposta que utiliza um Planejador Genético no auxílio à modelagem de processos em Workflow é descrita, realizando também experimentos de comparação com um particular planejador condicional. O Capítulo 5 mostra a flexibilidade de representação da arquitetura proposta através de estudo de casos computacionais aplicados no domínio de Web Service. O Capítulo 6 conclui o trabalho, analisando seu impacto potencial, visando à melhoria de desempenho e qualidade dos sistemas de workflow futuros.

Capítulo 2

Sistemas de Workflow

De acordo com Allen [4], Workflow é uma disciplina, uma prática e um conceito. É também uma tecnologia valiosa que, se corretamente implementada pode trazer benefícios significativos nas operações de uma organização.

Sistemas de Workflow têm suas origens a partir das pesquisas em automação de escritórios nos anos 70, quando o principal objetivo era oferecer soluções para gerar, distribuir e compartilhar documentos dentro de uma organização [52]. Na década de 80, sofreram a influência das pesquisas de Groupware e trabalho cooperativo apoiado em computadores, conhecido pela sigla CSCW, que tornou as ferramentas de workflow um instrumento de coordenação de trabalho em equipe. Na década de 90, continuou sua evolução tirando vantagem do rápido crescimento das redes de computadores (Internet/Intranet). As interações inter e intra-organizacionais, baseadas na exploração do potencial da WWW, levaram as pesquisas em workflow a um novo patamar voltado para definição de arquiteturas distribuídas de execução de processos [34]. O processamento distribuído e a interoperabilidade de aplicações trouxeram então novos desafios para os sistemas de workflow e sua construção.

Este capítulo trata dos conceitos básicos de workflow, estabelecendo um conjunto de definições que servem como referência para o restante desse trabalho. A seção 2.1 apresenta os conceitos envolvidos em workflow e também um modelo de referência proposto pela WfMC, que é uma entidade dedicada com o desenvolvimento de padrões e terminologias nesta área. A seção 2.3 expõe a relação existente entre a linguagem XML e Workflow, apresentando a forma de representação de modelo de Workflow em XML utilizada no presente trabalho. A seção 2.4 apresenta a execução

distribuída de processos em Workflow, através da composição de serviços que estão disponíveis em uma rede de computadores.

2.1 O que é workflow?

Este trabalho, considera a definição workflow proposta pela WfMC [61]. De acordo com esta entidade, "*Workflow é a automação de processos de negócios, no todo ou em parte, no qual documentos, informações ou atividades são passadas de um participante para outro, de acordo com um conjunto de regras*".

Esse conceito reflete o que existe em várias outras definições da literatura de workflow, e por estarem diretamente relacionados à área de negócios em organizações, sistemas de workflow têm sido indicados como ferramenta para o apoio computacional a processos de negócio.

Ao conjunto de atividades que são executadas para a realização de um trabalho, chama-se *processo*, que, em uma organização, pode ser entendido, também, como sendo um procedimento em que documentos, informações e tarefas são passadas entre os participantes para a realização de um objetivo de negócio qualquer. Tem-se, então, o que se denomina processo de negócio [34, 57].

Os processos de negócio são representados por fluxos de trabalho, ou seja, modelos que especificam: as tarefas que compõem o processo, a ordem e as condições que as tarefas devem ser executadas, os executores de cada tarefa, as ferramentas a serem utilizadas e as informações manipuladas durante sua execução. Os fluxos de trabalho, que representam os processos de negócio, podem ser interpretados/automatizados por sistemas de workflow como mostrado na Figura 2.1.

Os fluxos de trabalho, em situações de negócio específicas, estão contidos no sistemas de workflow, que têm por objetivo auxiliar as organizações na especificação, execução, monitoração e coordenação desses fluxos em um ambiente organizacional [60].

Sistemas de workflow possuem grande relevância para o cotidiano das organizações atualmente, uma vez que podem interagir de modo pró-ativo com seus usuários, controlando o processo de negócio, emitindo avisos sobre os mais diferentes temas e auxiliando a execução de uma tarefa.

Assim como os sistemas de bancos de dados são desenvolvidos e usados com

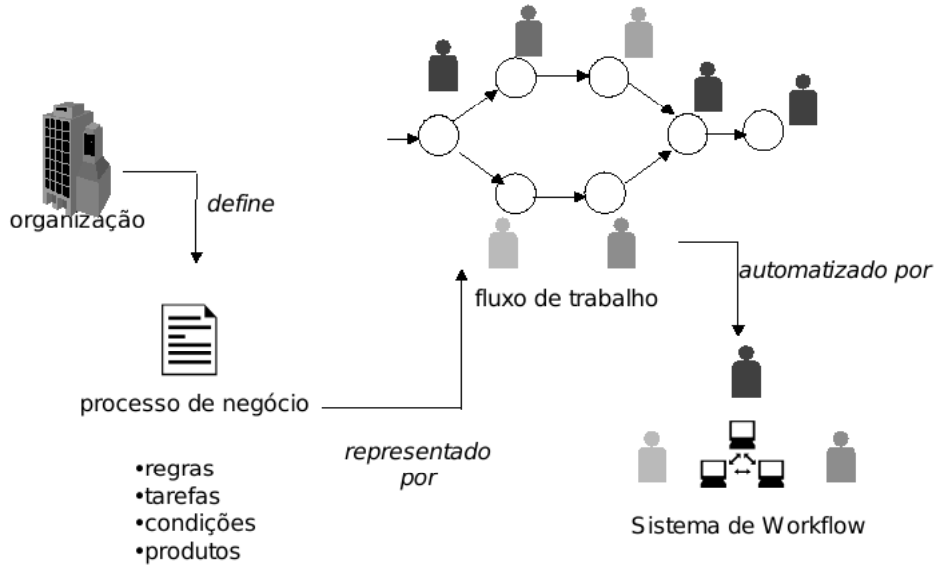


Figura 2.1: Processos de negócio automatizados por sistemas de workflow

o auxílio de um sistema de gerência de banco de dados (SGBD), um sistema de gerência de workflow (SGWf) é utilizado para implementar sistemas de workflow.

O sistema de gerência de workflow atua como um *sistema operacional*, controlando os processos entre os vários recursos (pessoas ou aplicações) [57]. A ele portanto, é destinada a logística dos processos. De acordo com Rob Allen, em [4], um sistema de gerência de workflow compreende:

> "Um sistema que define, cria e gerencia a execução de workflows através do uso de software, executando em um ou mais motores de workflow, que é capaz de interpretar a definição de processo, interagir com os participantes do workflow e, quando necessário, invocar o uso de ferramentas de Tecnologia da Informação e aplicações"

Deve-se notar que existe uma leve diferença entre sistema de gerência de workflow (SGWf) e sistema de workflow. A palavra "sistema" no contexto de workflow, segundo Van Der Aalst et. al. [57], significa todas as pessoas, máquinas e sistemas de informação computadorizados (ou não) que executam processos particulares. Entre esses processos particulares pode estar, por exemplo, um processo para análise de currículos de candidatos à vaga de emprego em uma organização (Figura 2.2). Assim sendo, a automação desse processo é um workflow que, em conjunto com outras

ferramentas, inclusive um SGWf, compõem o sistema de workflow para seleção dos candidatos.

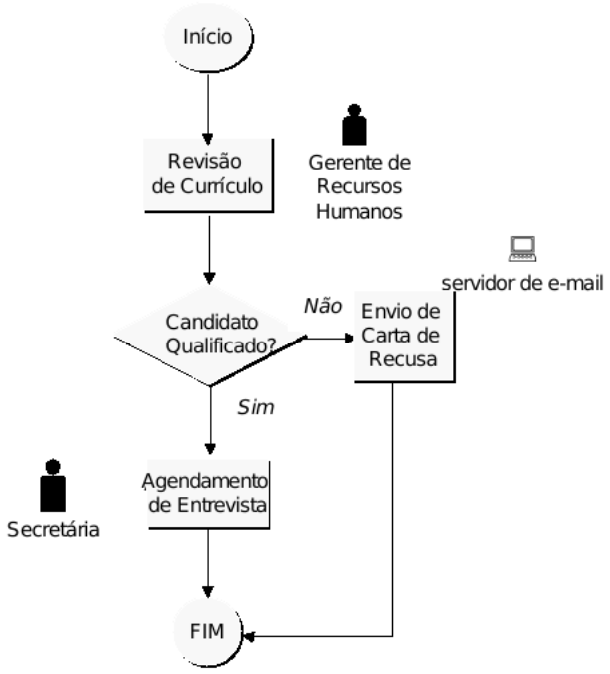


Figura 2.2: Processo de negócio para análise de currículos

Existem no mercado vários produtos diferentes que se colocam como ferramentas de gerência de workflow, eles possuem características comuns entre si no que diz respeito às suas funções principais [55]. A WfMC identifica que, em um nível mais alto, todos os SGWfs têm as seguintes funcionalidades, que se interagem de acordo com o diagrama da Figura 2.3.

- **Modelagem** - funções de "tempo de construção", que lidam com a definição e modelagem dos processos de workflow e suas tarefas constituintes. A modelagem de processo em *Workflow* consiste na organização das atividades inerentes ao problema, com os recursos responsáveis pela sua execução de forma a conduzir o processo de seu estado inicial até o estado meta.

- **Execução** - funções de "tempo de execução", que gerenciam e executam os processos de workflow em um ambiente operacional, além de sequenciar e distribuir as tarefas necessárias para a execução desses processos. São essas funções, que cuidam de criar uma instância de processo para lidar com algum objetivo específico, distribuir suas tarefas para os usuários participantes e acompanhá-lo

até o término de sua execução;

- **Monitoramento** - funções de "interação" (em tempo de execução) com os usuários participantes e com ferramentas computacionais para o processamento dos vários passos de cada tarefa.

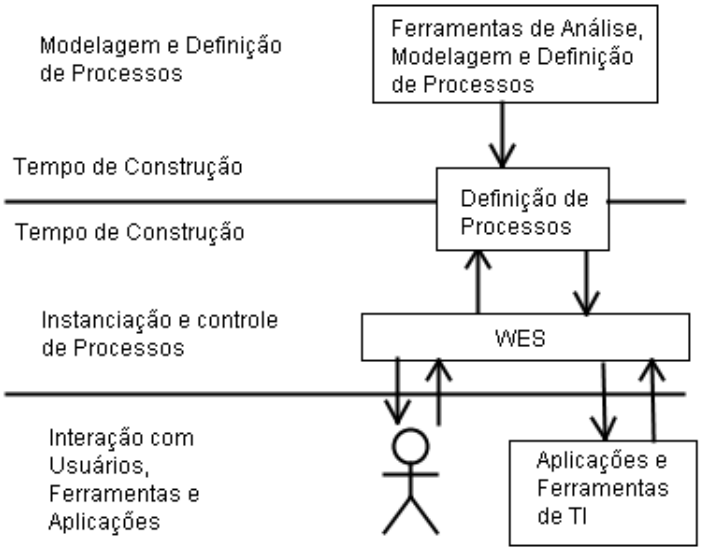


Figura 2.3: Relação entre as principais funções de um SGWf

O componente mais importante que compõe um sistema de gerenciamento de workflow, de acordo com o modelo de referência da WfMC é o WES. Composto por um ou mais motores de workflow, o WES é responsável por gerenciá-los e integrá-los, provendo o suporte necessário para a execução dos processos.

O motor de workflow é o ponto central de um SGWf [57], pois fornece o ambiente de execução necessário para os processos e concentra funcionalidades tais como a criação de instâncias de processos, coordenação e roteamento de atividades, geração de ítens de trabalho e controle de recursos.

2.1.1 O Modelo de Referência da WfMC

Afim de estabelecer padrões para a área de workflow, a WfMC desenvolveu um modelo de referência [60]. O objetivo principal desse modelo é identificar os muitos componentes que se integram para formar um ambiente de gerência e execução de *workflows*. O modelo é descrito em termos de suas respectivas interfaces e formatos

de intercâmbio necessários para promover a interoperabilidade necessária entre as várias ferramentas de workflow de desenvolvedores diferentes.

O modelo de referência da WfMC, apresentado na Figura 2.4, define cinco interfaces, cada uma cuidando da interação do WES com outros componentes do modelo. Os motores de workflow ficam dentro do WES, que, por sua vez, ficam dentro do WAPI. A WAPI representa a proposta da WfMC de um meio para a interoperação entre as ferramentas de workflow, e consiste em um conjunto de funções através das quais os componentes do modelo interagem uns com os outros.

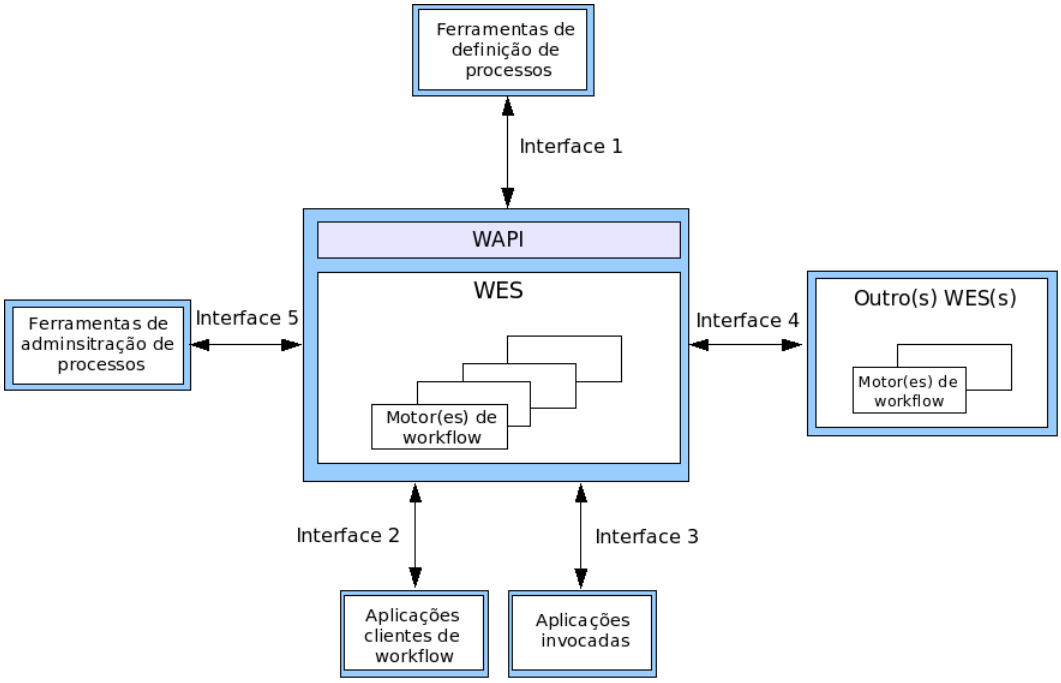


Figura 2.4: Modelo de referência da WfMC

Cada interface é responsável por uma parte do sistema. A interface 1 envolve as ferramentas de definição de processos que usam uma linguagem padrão, entendida por ferramenta de modelagem de workflow e por serviços de execução de workflow. A interface 2 dedica-se aos aplicativos-cliente de workflow, que se comunicam com o motor de workflow por meio do gerenciador da lista de trabalho. Esta interface padroniza as chamadas do gerenciador da lista, como, por exemplo, a recuperação de itens, e as declarações de início e término de atividades. Assim, um aplicativo construído por um terceiro pode utilizar os serviços de gerência de workflow de qualquer produto que suporte este padrão.

A interface 3 acessa as aplicações externas, invocadas pelo serviço de execução

de workflow para a realização de atividades específicas. Na interface 4 define-se a interação de um WES com outros externos a este, para a possível troca e/ou execução conjunta de processos comuns. Com esta interface, torna-se possível a execução de um processo através de vários SGWf diferentes. Por exemplo, várias organizações poderiam fazer parte de um único processo, aumentando ainda mais o grau de coordenação entre si. Por último, a interface 5 envolve a comunicação com ferramentas de administração e monitoramento dos processos executados. Assim, diversos aplicativos de outros fabricantes que suportem este padrão podem facilmente utilizar os serviços de gerência de workflow.

A WAPI e todas as suas interfaces são bem conhecidas pelos desenvolvedores de produtos de workflow, porém a WfMC ainda está trabalhando no processo de padronização do modelo.

2.2 Elementos de workflow

Sistemas de workflow auxiliam na gerência de tarefas que, quando encadeadas e distribuídas aos executores, visam cumprir um certo objetivo. Esse objetivo caracteriza um serviço ou trabalho (organizacional ou não).

Existem muitos e diferentes tipos de trabalho no mundo real, assim como escrever um relatório, fazer um bolo, projetar uma casa ou selecionar candidatos a uma vaga de emprego. Para todos esses exemplos, há processos e trabalhos que modificam ou criam objetos: um relatório, um bolo, uma casa, um empregado; ou seja,"coisas" genéricas que podem ser nomeadas ou particularizadas como, por exemplo: escrever o relatório de vendas do ano anterior, fazer um bolo de chocolate, projetar uma casa de três quartos, uma sala e uma cozinha, ou selecionar um candidato à vaga de analista de sistemas.

Um workflow constitui-se de um processo genérico com objetivos globais, como é o caso do processo "fazer um bolo"em que o bolo pode ser de chocolate, côco, etc. Quando o processo genérico é utilizado para um fim específico e contextualizado, tem-se o que se chama de *caso* ou *instância* de processo.

A tarefa é uma unidade lógica de trabalho que é executada por um recurso [57]. Dessa forma, são as tarefas que, coletivamente, completam o objetivo do workflow [12]. Exemplos de tarefas incluem: escrever uma carta, avaliar um relatório,

imprimir um documento, fechar uma venda, etc.

Uma tarefa refere-se a uma especificação genérica de algo a ser feito de maneira manual, automática ou semi-automática. Van Der Aalst et. al. [57] identificam a tarefa em dois tipos que, dependendo de seu estado em uma instância de processo de workflow, podem ser um item de trabalho ou uma atividade.

O item de trabalho é a combinação de um caso (instância de processo) e uma tarefa que aguarda para ser executada, conforme a representação da Figura 2.5. Dessa forma, um item de trabalho é uma tarefa que ainda não está em execução, mas pode ser executada a qualquer momento. Em geral, itens de trabalho são contidos em listas de trabalho até que sejam selecionados para a execução. Uma atividade, ao contrário do item de trabalho, refere-se à tarefa em estado de realização dentro da instância de processo. Portanto, a atividade constitui-se do item de trabalho que foi selecionado da lista de trabalho e está sendo executado. É importante notar que, diferente da tarefa, o item de trabalho e a atividade estão ligados a uma instância específica.

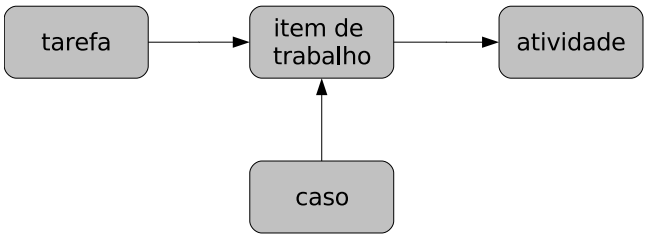


Figura 2.5: Relacionamento entre os termos tarefa, caso, item de trabalho e atividade

Em processos de negócio, o trabalho é direcionado aos participantes, também chamados de recursos [57, 61]. Assim como existe um modelo de processos (i.e., a ordem lógica e temporal das atividades que são necessárias para processar um objetivo de negócio), os recursos são descritos em um modelo de recursos [65]. Dessa forma, o modelo de recursos contém a definição dos recursos humanos e tecnológicos que estão envolvidos na execução do sistema de workflow (modelo de processos) como participantes.

De acordo com a WfMC [61], a partir do modelo de recursos da organização, os participantes do workflow podem ser divididos em dois tipos: os participantes concretos e os abstratos.

Os participantes concretos caracterizam o aspecto estático do modelo de recur-

sos conhecidos em tempo de construção do modelo de processos. Portanto, são as pessoas, máquinas e software que compõem o corpo da organização e que executarão as tarefas a eles atribuídas. Os participantes abstratos refletem os aspectos dinâmicos do sistema de workflow e são conhecidos em tempo de execução de um processo. Estes são determinados e atribuídos aos participantes concretos durante a execução do workflow, e, em geral, incluem recursos, conjunto de recursos, unidades organizacionais e papéis.

Recurso é o nome genérico para uma pessoa, máquina ou grupo de pessoas e/ou máquinas que podem realizar tarefas específicas [57]. No entanto, as tarefas também podem ser atribuídas a um conjunto de recursos ou aos recursos que estão inseridos em uma unidade organizacional ou departamentos, como por exemplo, o departamento de vendas, secretaria, tesouraria ou a gerência.

Em geral, os recursos trabalham de acordo com o seu perfil organizacional [64]. Neste caso, somente as tarefas que são específicas daquele perfil organizacional podem ser executadas. Van Der Aalst et. al. [57] fazem a divisão de recursos em grupos; os recursos agrupados compartilham os mesmos cargos, qualificações ou funções operacionais dentro da organização e geram a noção de classes de recursos. A essas classes de recursos com o mesmo perfil, executando tarefas específicas, dá-se o nome de papel (ou *role*, em inglês) [57, 65].

Uma das funções básicas de um sistema de workflow é coordenar fluxos de trabalho. Todas as vezes em que um recurso ocupado por uma atividade termina a sua execução, o sistema direciona o fluxo de trabalho a outro recurso, até que o objetivo seja alcançado. Dessa forma, cada instância de processo segue um caminho que é especificado na definição de processos por meio do roteamento das tarefas que o compõem.

Os principais tipos de roteamento e que estão presentes neste trabalho são sequencial, paralelo ou condicional [57]:

- **Roteamento Sequencial (SEQ)**: ocorre quando duas ou mais atividades devem ser executadas em seqüência.
- **Roteamento Paralelo (AND)**: quando duas ou mais atividades podem ser executadas independentemente e simultaneamente; são disparadas em determinado ponto do processo e ressincronizadas em um ponto mais adiante.

- **Roteamento Condicional ou Seletivo (OR)**: acontece quando a próxima (ou próximas) atividade a ser executada depende de uma escolha, que por sua vez depende das propriedades específicas de cada caso; o resultado dessa escolha determina qual seria essa próxima atividade.

2.3 XML e Workflow

A utilização da linguagem XML [50] auxilia no suporte ao intercâmbio de informações, facilitando a interoperabilidade entre diferentes sistemas. XML é uma linguagem de marcação baseada em texto, simples e flexível, onde os dados são marcados através de caracteres de marcação, que contêm o significado destes dados. Tal marcação faz com que diferentes sistemas possam trocar informações de uma maneira fácil. Além disso, XML, é independente de plataforma ou tecnologia.

Edelweiss et. al. [16] propõem uma divisão de modelos de workflow representados em XML em duas abordagens. Essa divisão considera a forma de representação dos elementos como critério. Tais abordagens são as seguintes:

- Representação Explícita de Transições (RET) - A principal característica das linguagens desta abordagem é que existe um elemento XML para denotar uma transição de forma explícita, ou seja, ligando uma atividade e outra diretamente. Dessa forma, este elemento deve possuir atributos que indiquem o elemento de origem, do qual parte, e o elemento de destino, no qual chega.

- Representação Implícita de Transições (RIT) - Os modelos deste grupo possuem características contrárias à abordagem anterior, pois não possuem elementos para representar as transições de forma explícita. Estes possuem estruturas que determinam como será construído o fluxo de controle do workflow. Cada estrutura possui sua semântica, a qual deve ser conhecida pelo mecanismo workflow.

Observe que na Figura 2.6 há uma parte de um workflow representado de forma gráfica 2.6(a), e dois modelos XML equivalentes, um representando RET 2.6(b) e outro representado RIT 2.6(c). Na representação explícita 2.6(b), observa-se a presença do elemento transição que explicita as transições do diagrama apresentado

em 2.6(a). Na representação implícita 2.6(c), não existe um elemento que represente as transições de forma explícita, mas existem estruturas que determinam como será o fluxo de controle do workflow, neste caso, *seq*, em que os elementos em seu interior são executados em seqüência e divisão, em que os elementos em seu interior serão executados em paralelo.

(a) Diagrama (b) RET (c) RIT

Figura 2.6: Representações das abordagens RET e RIT a partir de um diagrama representando parte de um workflow(a)

Existem várias linguagens e modelos propostos para representar aspectos da tecnologia de workflow, adotaremos uma representação baseada na abordagem RET denominada XPDL.

XPDL é uma linguagem utilizada para representação de processos de negócio em XML desenvolvida pela WfMC. Como exemplo desta linguagem, a definição de uma atividade é mostrada na Figura 2.7.

Esta linguagem integra a chamada Interface 1 da Figura 2.4, responsável pela parte de intercâmbio entre diferentes ferramentas de modelagem de workflow. Sendo assim, XPDL funciona como um modelo comum, de forma que qualquer ferramenta

$< ActivityId = "coletar" Name = "$**coletar**$\gg$

$< Description >$

$Coletar\ informações\ reclamação\ telefônica < /Description >$

$< Performer > System < /Performer >$

$< StartMode >$

$< Automatic/ >$

$< /StartMode >$

$< FinishMode >$

$< Automatic/ >$

$< /FinishMode >$

$< SimulationInformation >$

$< Cost > Alto < /Cost >$

$< TimeEstimation >$

$< Duration > 10 < /Duration >$

$< /TimeEstimation >$

$< /SimulationInformation >$

$< ExtendedAttributes >$

$< ExtendedAttribute >$

$\vdots$

$< /ExtendedAttribute >$

$< /ExtendedAttributes >$

$< /Activity >$

Figura 2.7: Uma atividades definida em XPDL

pode importar um processo descrito nessa linguagem ou exportar um processo em sua linguagem nativa. Uma característica importante desta linguagem é a possibilidade de definição de atributos extendido, que permite a integração desta com outras técnicas.

2.4 Workflow de Web Service

O desenvolvimento da Internet nos últimos anos resultou em um grande impacto nos sistemas de gerenciamento de Workflow, uma vez que os Workflow podem ser executados através da Internet [30].

A Internet vem sendo utilizada como meio para que as organizações disponibi-

lizem seus recursos de negócio como serviço, tornando-se uma plataforma comum global para comunicação e execução de várias atividades. Qualquer empresa pode disponibilizar partes de suas operações, assim como pode encontrar dinamicamente novos serviços com os quais pode interagir, possibilitando a composição de serviços externos existentes, dentro de uma aplicação interna, o que estimula o reuso e rápido desenvolvimento [7]. Quanto mais surgem aplicações e ferramentas baseadas na Web, mais e mais serviços são disponibilizados através da Internet [13].

Um componente-chave para a arquitetura de computação baseada em Web é a tecnologia Web Service. Um Web Service é uma interface que descreve uma coleção de operações que são acessíveis através de uma rede de computadores [6]. Um Web Service pode executar uma tarefa específica (ou um conjunto de tarefas), descrita através de um padrão, uma notação formal XML, denominada descrição de serviço. A descrição do serviço fornece todos os detalhes necessários para sua interação, incluindo o formato das mensagens que detalham as operações, protocolos de transporte e localização.

A tecnologia Web Service acelera a integração de aplicações internas e externas à empresa, uma vez que fornece um modelo de programação com linguagem neutra e ambiente neutro, ou seja, independente de linguagem e plataforma. Sistemas de negócio fracamente acoplados e flexíveis podem ser desenvolvidos utilizando-se Web Services para realizar a integração da aplicação. Web Services podem ser aplicados como uma tecnologia "de empacotamento"ao redor de aplicações existentes. Desta forma, novas soluções podem ser rapidamente desenvolvidas ou recompostas para endereçar novas oportunidades [22].

A arquitetura orientada a serviços é a arquitetura na qual os Web Services são inseridos e que incentivam o reuso de componentes de software [33]. Esta arquitetura permite que processos de negócios estejam acessíveis dentro de empresas e entre empresas. O paradigma de Web Services permite, ainda, que aplicações e serviços que estejam executando em diferentes plataformas se comuniquem e operem em conjunto de uma maneira fácil.

2.4.1 Arquitetura orientada a serviço

Conforme Leymann et. al. [33], o primeiro passo importante da arquitetura orientada a serviço foi o desenvolvimento do conceito de objetos como blocos de construção, que combinam dados e funções dentro de uma unidade encapsulada. Os conceitos de classes, herança e polimorfismo foram introduzidos, permitindo dessa maneira a construção de "redes"de classes.

Na orientação a objetos, componentes de software fornecem modularidade a projetos de software orientados a objeto. Assim, componentes ou módulos devem ser projetados e implementados de uma maneira que possam ser reusados em muitos programas diferentes. Possibilitando que programas possam ser construídos com um mínimo de desenvolvimento a "partir do esboço".

A arquitetura orientada a serviços mudou dramaticamente tais conceitos, uma vez que disponibiliza serviços na Web, os Web Services. Assim sendo, faz-se necessário que serviços sejam descritos e publicados de modo que qualquer um seja capaz de localizá-los e invocá-los. Assim, deve possuir as seguintes características [33]:

- Descoberta dinâmica dos serviços registrados. A pesquisa por serviços pode ser realizada de acordo com certos critérios, como, por exemplo critérios de negócios, tempo de entrega e preço, dentre outros.
- Organização dos serviços, de modo que se possa entender facilmente o que um serviço oferece.
- Descrição dos serviços, de modo que um serviço possa ser invocado de maneira correta. Isto inclui formatos e protocolos para invocar Web Services.

A Figura 2.8 ilustra uma arquitetura orientada a serviço, apresentando três papéis ou funções e três operações. Como é mostrado na figura, os três papéis são provedor de serviço, requerente de serviço e registro de serviços. Os objetos são o serviço e a descrição do serviço. As operações que são executadas pelos atores fazendo uso desses objetos são publicar, encontrar e conectar.

O Web Service e sua definição são criados pelo provedor de serviço, que então pode publicar o serviço num registro de serviços [22]. O registro de serviços é o lugar em que todas informações sobre todos os serviços disponíveis são mantidas. Assim,

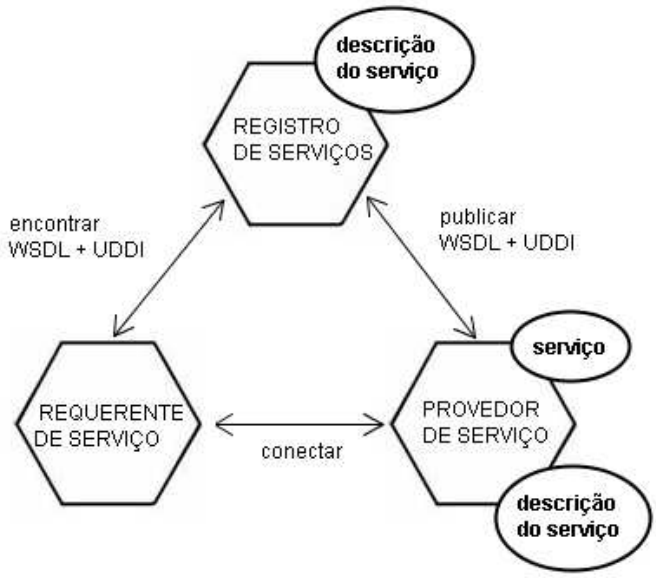


Figura 2.8: Atores, objetos e operações de Web Service

quando um provedor de serviços deseja oferecer serviços, este os publica colocando as entradas apropriadas em tal registro.

Uma vez publicado, um requerente pode localizar um serviço que atenda a seus requisitos através do registro de serviços, fornecendo ao requerente a descrição do serviço e uma URL para o serviço propriamente dito [22].

Quando um requerente localiza um serviço que atenda suas necessidades, este se conecta ao provedor do serviço, utilizando as informações de conexão armazenadas no registro de serviços, quando então pode fazer uso do serviço selecionado. As informações de conexão contêm especificações do protocolo que o requerente deverá utilizar, assim como a estrutura das mensagens de requisição e das respostas com os resultados.

Segundo Sun [49], SOAP, UDDI e WSDL são as principias tecnologias que estão sendo utilizadas atualmente no contexto de Web Services. A tecnologia SOAP [51] é um protocolo baseado em XML e HTTP/SMTP para transporte de mensagens. É independente de linguagem de programação e da plataforma na qual a aplicação está sendo executada, e não necessita de qualquer tecnologia específica. SOAP define um mecanismo para a comunicação em Web Services através da Internet. Especifica o formato de mensagens que são trocadas entre o requerente, o provedor e o registro de serviços. Em particular, SOAP descreve como HTTP pode ser utilizado

para disponibilizar um mecanismo que realize a chamada a procedimento remoto (RPC) através da Internet [33]. Para que seja possível a troca de mensagens, SOAP define um envelope. Um envelope SOAP é formado pelo *body* (corpo), dentro do qual a mensagem está incluída, e também opcionalmente pode conter um *header* (cabeçalho), *Soap-especific Header*. O envelope completo, *body* mais *header*, é um documento XML.

UDDI é uma especificação padrão para registrar, descrever e procurar Web Service [56]. Um provedor que registra seus serviços deve fornecer informações sobre o negócio e os serviços, deve informar como se conectar, além de prover informações técnicas sobre o serviço. Tais informações são armazenadas num formato comum que consiste em 3 partes, a saber:

- Páginas Brancas - Fornece informações gerais sobre o negócio, como nome, descrição, telefone e assim por diante.
- Páginas Amarelas - Descreve os negócios em termos de taxonomias padronizadas. Esta informação pode seguir um padrão de categorização de empresas, de modo que os serviços possam ser localizados por categoria, empresa ou localização geográfica.
- Páginas Verdes - Possui informações sobre o serviço, de como se conectar com o provedor e informações técnicas específicas do serviço.

É importante notar que UDDI é uma especificação para um registro, não é um repositório. Sua função é a mesma que a de um catálogo, permitindo que clientes encontrem serviços disponíveis. Não são repositórios porque não contêm os serviços, apenas informações sobre os mesmos. Web Services podem funcionar sem o registro UDDI, embora este possa ser utilizado para encontrar provedores de serviços e suas descrições em WSDL [58].

WSDL estabelece um padrão para especificar os detalhes de um Web Service [58]. É uma linguagem baseada em XML usada para especificar interfaces, informações sobre conexões e detalhes de desenvolvimento de Web Services. A sua vantagem é permitir que Web Services definidos com WSDL possam ser utilizados por clientes que não possuem conhecimento prévio do serviço. Uma descrição WSDL de Web Services provê toda informação necessária para invocar um serviço. Tipo de porta,

as operações que o tipo de porta suporta e a estrutura das mensagens de entrada e saída descrevem um serviço particular de uma forma abstrata [33].

2.4.2 Modelagem de Workflow de Web Service

À medida que a adoção do uso de Web Services se acelera, o conjunto de serviços aumenta, incentivando o desenvolvimento de modelos mais dinâmicos para aplicações *just-in-time*, ou seja, aplicações que devem ser desenvolvidas num momento exato, e integração de negócios através da Internet [22].

A capacidade de negócios pode ser definida com um conjunto de serviços que trabalham de maneira coerente em busca de um propósito de negócio definido. Tal conceito ajuda na mudança de uma visão funcional de uma empresa fechada, tradicional, para uma visão orientada a serviço de uma empresa aberta, capacitada para negócios eletrônicos. O que permite que haja uma evolução de aplicações projetadas de forma fortemente acopladas, inflexíveis, para aplicações fracamente acopladas, com conjunto flexível de interfaces que facilita o compartilhamento de capacidades de negócios [3]. O gerenciamento e colaboração de serviços, permitido pela computação baseada em serviços, permite que negócios desagregados possam melhorar a forma de cooperação entre si, aumentando assim a produtividade.

A tecnologia de workflow é utilizada para coordenar as iterações entre os Web Services, que de uma forma composta, visam alcançar um objetivo comum dentro da organização. E os Web Services representam os passos lógicos que compõem o Workflow, ou seja, os passos que devem ser executados para que se alcance um objetivo de negócio dentro da organização.

A utilização de Web Service para realizar processos de negócios em parceria, fornece uma flexibilidade na sua constituição. Com a utilização de Web Services é possível que uma organização, tendo determinado que, ao terceirizar alguns serviços as operações da companhia poderão ser realizadas de forma mais eficiente, busque num registro UDDI de preferência parceiros de negócios que ofereçam tais serviços, e realiza a seleção baseados em seus critérios.

A Figura 2.9 apresenta uma visão geral da relação existente entre o registro de serviços , o provedor de serviço, o Modelador que realiza a modelagem dos processos de workflow e o modelo de Workflow representado através do gráfico. Considera-

se que os Web Services correspondem às atividades no processo de Workflow. O Modelador busca as informações dos Web Services disponíveis no registro UDDI, realizando a modelagem dos processos. Esses serviços conectam, em tempo de execução, com os respectivos provedores de serviços.

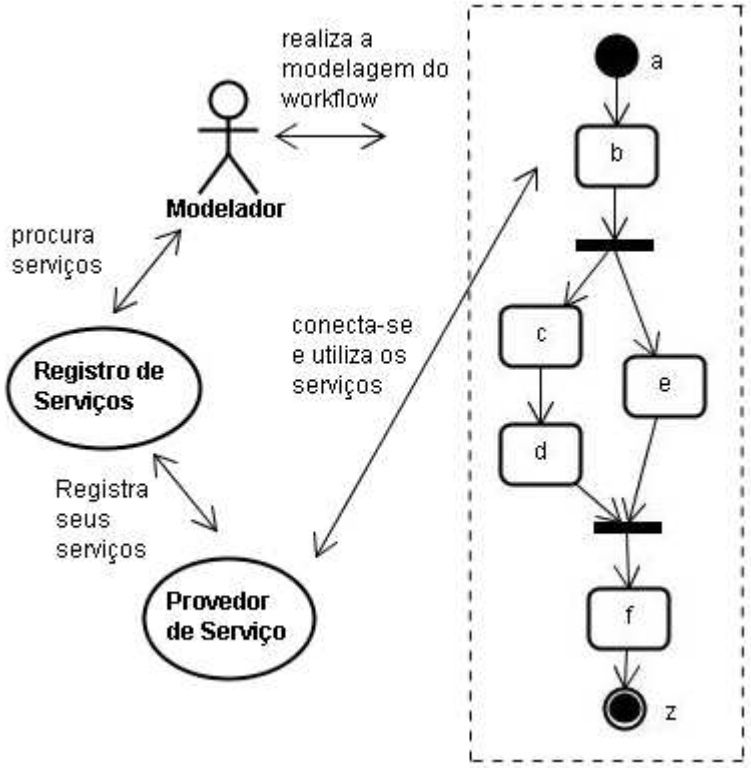


Figura 2.9: Iteração entre o Registro de Serviços, o Provedor de Serviço, o Modelador e o Workflow

A Figura 2.10 fornece uma visão geral de como ocorre a troca de mensagens entre o requerente e o provedor do Web Service. Considera-se um processo Workflow simples que deve executar três Web Services em sequência. No momento de sua execução, cada Web Service realiza uma conexão com o respectivo provedor previamente definido. Os Web Services necessários para modelagem de processos de negócio são encontrados no registro UDDI baseados na categoria do problema a qual pertencem.

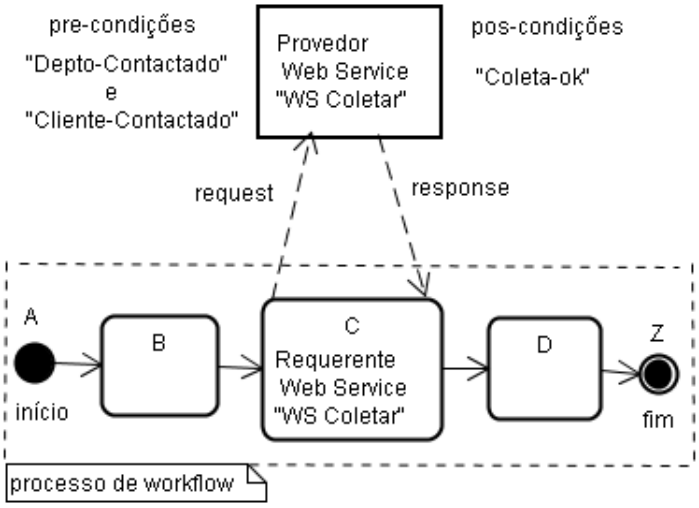


Figura 2.10: Conexão e execução entre o requerente e o provedor do Web Service.

2.5 Considerações finais

O processo de modelagem, em geral, é realizado manualmente por especialistas em processos, não garantindo a adequação desejada, podendo estar sujeita à falhas e definições ineficientes. Além disso, em geral, a modelagem de processos complexos exige muito tempo e uma análise detalhada, devido ao grande número de atividades envolvidas. Agravando também, quando estas atividades estão fisicamente distantes. Para auxiliar no problema de modelagem dos fluxos de processos tem crescido o interesse em aplicar técnicas de Planejamento em Inteligência Artificial [41, 43, 45].

Outra característica considerada neste capítulo foi a aplicação de workflow no domínio de web service. Além dessa aplicação possibilitar a integração de serviços internos e serviços que transpõem as fronteiras organizacionais, torna o processo mais flexível, pois decisões de projeto que estejam encapsuladas em Web Services que compõem o processo, podem ser revistas e alteradas sem afetar outras partes do processo.

Capítulo 3

Planejamento em IA

O planejamento em IA envolve determinar um conjunto ordenado de ações que quando executadas a partir de um estado inicial que satisfaça as circunstâncias dadas, resulte num estado final que satisfaça a meta [46].

Planejamento é uma área de pesquisa em IA que originou a partir de trabalhos científicos que tentavam criar, principalmente com o uso da Lógica de Primeira Ordem, solucionadores gerais de problemas, como por exemplo, o GPS [15]. O objetivo de tais sistemas era encontrar, de forma automática, soluções para os mais diversos problemas nos mais diferentes domínios.

As técnicas de planejamento em IA foram evoluindo ao longo do tempo, mas em qualquer um dos modelos uma seqüência de ações é um plano se o resultado de sua execução atinge a meta. O plano é visto então como a solução de um problema de planejamento. O problema de Planejamento é encontrar uma seqüência de ações, chamada plano, que resolva um determinado problema proposto. Mais especificamente, um Sistema de Planejamento é aquele que recebe como entrada uma descrição de um estado inicial do mundo, de um objetivo e de um conjunto de ações que podem ser aplicadas e gera como saída um plano, que é a seqüência de ações que transforma o estado inicial no estado final. Cada ação tem as suas próprias pré e pós condições de execução. As definições de planejamento apresentadas em [46] complementam esta definição.

Este capítulo apresenta conceitos e técnicas de planejamento IA e algumas abordagens de planejadores. A seção 3.1 apresenta conceitos envolvidos em Planejamento em IA e também algumas linguagens envolvidas em suas representações. A seção

3.2 expõe algumas propostas de planejadores e a seção 3.3 apresenta algumas abordagens evolutivas para Planejamento em IA.

3.1 O que é Planejamento em IA ?

Um problema de planejamento geralmente é representado na forma de uma tripla (A, I, G), onde A é um conjunto de ações que podem ser executadas, I é uma descrição do estado corrente do mundo (estado inicial) e G é uma descrição do objetivo (estado meta).

Um planejador é um sistema que implementa um algoritmo de planejamento. Este recebe como entrada um problema de planejamento e gera como saída um plano, ou seja, uma seqüência ordenada de ações que, quando aplicadas a um mundo que satisfaça o estado inicial, conduzem ao estado final, onde a meta é válida. Esta é a formulação de um planejamento clássico, cujas ações são determinísticas, ou seja, os efeitos previstos são os obtidos e o mundo é representado fielmente pelo estado interno do planejador.

O ambiente de planejamento geralmente são do tipo Clássico e não-Clássico. O ambiente de Planejamento Clássico é aquele completamente observável, determinístico, finito, estático (mudanças do mundo ocorrem apenas como resultado do efeito de uma ação), e discreto (no tempo, ações, objetos, e efeitos). O ambiente de Planejamento não-Clássico é mais complexo, por envolver ambiente parcialmente observável ou estatístico, e por isto necessita de um conjunto de algoritmos e ações desenhados para estas situações.

3.1.1 A Linguagem dos Problemas de Planejamento em IA

As considerações feitas sobre as dificuldades de se resolver problemas de planejamento sugerem que a representação de problemas, estados, ações e metas deveria ser de tal forma que os algoritmos possam fazer uso da estrutura lógica do problema. A chave é encontrar uma linguagem expressiva o suficiente para descrever uma grande variedade de problemas, mas restritiva o bastante para suportar os algoritmos eficientemente.

Junto com as técnicas de planejamento também tem evoluído sua forma de repre-

sentação através de uma linguagem de representação. Na origem destas linguagens está o STRIPS clássico [17], a linguagem ADL [44] e mais recentemente PDDL [20], dentre outras.

◦ **Linguagem STRIPS**

Criada no início dos anos 70, a linguagem STRIPS [17] foi inicialmente utilizada em agentes robóticos e é considerada "clássica"em planejamento. A sintaxe desta linguagem é dada através da Representação de Estados, Representação de Metas, Ações e Domínio do problema

Representação de Estados: O mundo é decomposto em estados lógicos, e um estado é representado como uma conjunção de literais positivos instanciados, ou seja, predicados aplicados sobre constantes. Em (3.1) tem-se um exemplo do uso de literais proposicionais que poderia representar o estado inicial do problema do "pneu furado". Também é permitido o uso de literais da lógica de primeira-ordem, como apresentado em (3.2) que poderia representar um estado no problema de "entrega de pacote".

$$pneu\_furado \wedge estepe\_cheio \tag{3.1}$$

$$Em(\text{Avião}_1, Rio) \wedge Em(\text{Avião}_2, \text{São\_}Paulo) \tag{3.2}$$

No caso de literais da lógica de primeira-ordem estes devem ser instanciados e não podem incluir funções. As expressões (3.3) e (3.4) são exemplos de literais não permitidos.

$$Em(x, y) \tag{3.3}$$

$$Em(\text{Avião}(1), \text{São\_}Paulo) \tag{3.4}$$

É considerada a hipótese de mundo-fechado, isto é, qualquer condição que não seja mencionada em um estado é assumida como falsa. Assim, em um estado inicial dado por (3.5) o literal $\neg Em(\text{Avião}_1, Rio)$ pode ser suprimido da descrição.

$$Em(\text{Avião}_1, \text{São\_}Paulo) \wedge \neg Em(\text{Avião}_1, Rio) \tag{3.5}$$

Representação de Metas: Uma meta é uma representação parcial de um estado. É representada, da mesma forma que um estado, como uma conjunção de

literais instanciados e positivos, como por exemplo em (3.6).

$$Em(Pacote_1, \text{São\_}Paulo) \tag{3.6}$$

Um estado s satisfaz uma meta g se s contém todos os literais de g, mesmo que s contenha outros literais. Considerando o exemplo da meta (3.6), observa-se que esta é satisfeita pelo estado do mundo em (3.7).

$$Em(\text{Avião}_1, \text{São\_}Paulo) \wedge Em(Pacote_1, \text{São\_}Paulo) \tag{3.7}$$

Representação de Ações: Uma ação é representada em termos das pré-condições que tenha que assegurar antes que possa ser executada e dos efeitos que resultam de sua execução. Por exemplo, uma ação para fazer um avião voar de uma localização para outra pode ser descrita como em (3.8).

Ação (Voar(p,de,para),
PRÉ-CONDIÇÃO: Em(p,de) ∧ Avião(p) ∧ Aeroporto(de) ∧ Aeroporto(para)
EFEITO: ¬Em(p,de) ∧ Em(p,para)) (3.8)

Esta construção é chamada de operador, pois representa um número de diferentes ações que podem ser derivadas pela instanciação das variáveis *(p, de, para)*. Em geral, um operador consiste em três partes:

1. Descrição da ação: Nome da ação e lista de parâmetros;
2. Pré-condição: uma conjunção de literais positivos, que não contêm funções e que devem ser verdadeiros antes que a ação seja aplicada. As variáveis presentes nestes literais devem fazer parte da lista de parâmetros;
3. Efeito[1]: É uma conjunção de literais, sem funções, que descrevem como o estado é modificado quando a ação é executada. As variáveis presentes nestes literais também devem fazer parte da lista de parâmetros.

Domínio: O conjunto formado pelos operadores é conhecido como domínio do problema, por representar o conjunto de operadores que podem atuar no estado corrente do mundo para gerar o estado subseqüente. As várias restrições impostas pela

[1]Na definição original da linguagem STRIPS a lista de efeitos era representada por duas listas (add e delete). A primeira, era a lista de inclusão contendo os literais positivos, enquanto a segunda era a lista de remoção contendo os literais negativos.

linguagem STRIPS eram aceitas com a perspectiva de produzir algoritmos de planejamento mais simples e mais eficientes, sem tornar difícil a descrição de problemas reais. Uma das restrições mais importante é a que diz que os literais devem ser livres de funções. Com esta restrição, tem-se a segurança que qualquer *operador* para um determinado problema pode ser transformado em um conjunto finito de ações puramente proposicionais, sem variáveis. Por exemplo, no domínio do problema de transporte aéreo de cargas com 10 aviões e cinco aeroportos, pode-se traduzir o operador *Voar(p,de,para)* em $10 * 5 * 5 = 250$ ações puramente proposicionais.

Semântica: Se um estado satisfaz as pré-condições de uma ação, diz-se que esta é aplicável, caso contrário, a ação não tem efeito. Para decidir pela aplicação, ou não, de um operador que utilize lógica de primeira-ordem nas pré-condições, é necessária a substituição das variáveis na pré-condição. Seja o exemplo (3.9) um estado corrente.

$$\begin{aligned} &Em(P_1, RIO) \wedge Em(P_2, BHZ) \wedge \text{Avião}(P_1) \wedge \text{Avião}(P_2) \\ &\wedge Aeroporto(RIO) \wedge Aeroporto(BHZ) \end{aligned} \tag{3.9}$$

Sendo as pré-condições do operador (3.10) satisfeitas em (3.9), com as substituições de (3.11), a ação é aplicável.

$$\begin{aligned} &\textit{Ação (Voar(p,de,para),} \\ &\quad \textit{PRÉ-CONDIÇÃO: Em(p,de)} \wedge \textit{Avião(p)} \wedge \textit{Aeroporto(de)} \wedge \textit{Aeroporto(para)} \\ &\quad \textit{EFEITO:} \neg \textit{Em(p,de)} \wedge \textit{Em(p,para)}\ ) \end{aligned} \tag{3.10}$$

$$\{p/P1, de/RIO, para/BHZ\} \tag{3.11}$$

Quando uma ação aplicável é executada, o estado corrente *s* é transformado em um estado *s'*, que é o mesmo de *s*, exceto que qualquer literal P positivo no efeito do *operador* é adicionado a *s'* e qualquer literal negativo $\neg P$ é removido de *s*\. Assim, depois de aplicar $Voar(P_1, RIO, BHZ)$, o estado atual é dado por (3.12), observando que os literais não mencionados na conjunção da cláusula efeito permanecem inalterados.

$$\begin{aligned} &Em(P1, BHZ) \wedge Em(P2, BHZ) \wedge \text{Avião}(P1) \wedge \text{Avião}(P2) \\ &\wedge Aeroporto((RIO) \wedge Aeroporto((BHZ)) \end{aligned} \tag{3.12}$$

O algoritmo do planejador STRIPS empilha os literais da meta e sucessivamente aplica ações ao estado corrente até que algum literal seja válido ou todas as alternativas sejam tentadas. Caracterizando-o como uma busca no espaço de estados do problema.

O planejador STRIPS é um planejador linear. O conceito de linearidade vem da "suposição de linearidade", pois cada literal da meta é tratado como uma submeta e considerado independente dos demais. Portanto, um planejador é chamado de linear ou não-linear, se considera ou não algum tipo de interação entre as submetas de acordo com a "suposição de linearidade".

Uma das limitações do STRIPS está em considerar a suposição de linearidade, ou seja, todos os literais são independentes uns dos outros. Entretanto, há interações de literais que podem forçar que eles tenham que ser atingidos em certa ordem ou satisfeitos juntos. A mais conhecida destas limitações é a "Anomalia de Sussman" [46].

◦ ADL - Action Description Language

Com a evolução da área de planejamento, percebeu-se que a representação STRIPS não era suficientemente expressiva para alguns domínios reais. Como resultado, surgiram várias variantes desta linguagem. Talvez a sucessora mais importante tenha sido a Linguagem ADL [44]. Em ADL, a ação de "Voar"pode ser dada por (3.13). A pré-condição *(de ≠ para)* evita que um vôo seja feito entre um mesmo aeroporto. Isto não pode ser expresso de forma sucinta em STRIPS. A Figura 3.1 mostra uma comparação entre as linguagens STRIPS e ADL [46].

$$
\begin{array}{l}
\textit{Ação (Voar(p: Avião, de: Aeroporto, para: Aeroporto,} \\
\quad \textit{PRÉ-CONDIÇÃO: Em(p,de)} \wedge \textit{(de} \neq \textit{para)} \\
\quad \textit{EFEITO:} \neg \textit{Em(p,para)} \wedge \textit{Em(p,para)} \\
)
\end{array}
\tag{3.13}
$$

A possibilidade de representar "efeitos condicionais"permitem a redução da quantidade de ações instanciadas para resolver um problema, visto que uma mesma ação pode ser utilizada para diferentes situações.

◦ PDDL

Os vários formalismos de planejamento usados em IA foram sistematizados em uma sintaxe padrão chamada de Linguagem de Definição do Domínio de Planejamento, ou PDDL, criada por [20] para a primeira versão da competição mundial de planejamento automático - *AI Planning Systems 98*. O objetivo dessa linguagem é estabelecer um padrão de notação para a descrição de domínios utilizados, que permita

Linguagem STRIPS	**Linguagem** ADL
Somente literais positivos nos estados **Pobre ∧ Desconhecido**	Literais positivos e negativos nos estados: **¬Ricos ∧ ¬Famosos**
Hipótese de mundo fechado: Literais não mencionados são considerados *falsos*.	Hipótese de mundo aberto: Literais não mencionados desconhecidos.
O efeito **P ∧ ¬Q** significa adicionar **P** e remover **Q**.	O efeito **P ∧ ¬Q** significa adicionar **P** e **¬Q** e remover **¬P** e **Q**.
A meta só pode ter literais aterrados: **Rico ∧ Famoso**	É permitido o uso de quantificadores na meta: **∃Em(P1,x) ∧ Em(P2,x)** onde **P1** e **P2** estão no mesmo lugar.
A meta é uma conjunção: **Rico ∧ Famoso**	A meta permite conjunções e disjunções: **¬Pobre ∧ (Rico ∨ Esperto)**
Efeitos são conjunções.	Permite efeitos condicionais: **when P: E** Significa que **E** é um efeito somente se **P** for satisfeito.
Não suporta verificação de igualdade	O predicado de igualdade (**x = y**) é suportado.
Não suporta tipos.	Variáveis podem ter tipos, com em: (**p: Avião**).

Figura 3.1: Comparação entre as linguagens STRIPS e ADL.

aos pesquisadores trocar *benchmarks*[2] de problemas e comparar resultados de desempenho dos planejadores durante competições. Desde a sua criação, a linguagem PDDL vem se tornando uma referência para a maioria dos planejadores.

Em 2002 surgiu uma extensão importante, a PDDL 2.1 [19], cujas características mais importantes são a possibilidade de definir tempo de duração para as ações e descrever os efeitos do tempo sobre as ações. Além destas, apresenta modificações no tratamento de expressões numéricas e permite o uso de uma função de otimização do plano (denominada métrica), descrita na própria representação do problema. Esta versão de PDDL estabelece quatro níveis.

- Nível 1: Corresponde aos níveis proposicionais e de ADL da versão anterior;
- Nível 2: Estabelece uma sintaxe para manipular expressões numéricas. Tais expressões são compostas por operadores aritméticos e funções numéricas. Estas funções permitem associar valores a objetos do problema.

[2]Teste de desempenho de sistemas

- Nível 3: Introduz o uso de ações temporais discretas. Esta nova funcionalidade permite definir em que instante, durante ou depois, da execução da ação os seus efeitos ocorrem, e por quanto tempo permanecem válidos,
- Nível 4: Permite modelar efeitos contínuos para ações temporais. É introduzido um símbolo ♯t que representa o tempo decorrido durante a execução da ação temporal.

Uma extensão mais recente denominada PDDL+ [18] introduziu um quinto nível. Este nível permite definir ações temporais em termos da inicialização e término de processos que são atividades que enquanto duram, alteram continuamente os valores das expressões numéricas do estado corrente. A versão PDDL 2.2 [14] introduz dois novos conceitos para a competição de 2004: predicados derivados e temporização de literais. Os primeiros referem-se à criação de regras que permitem deduzir valores independentes das ações do planejamento, de acordo com regras do tipo Se X é verdadeiro então Y é verdadeiro. A segunda refere-se às restrições de tempo a partir da qual determinado literal é assumido como verdadeiro. Por exemplo, a expressão (3.14) refere-se a uma janela de tempo na qual uma loja estaria aberta.

$$(: init(às\ 9\ (loja\_aberta))(às\ 20\ (not(loja\_aberta)))) \tag{3.14}$$

3.2 Buscas e Planejadores

Diversos sistemas de planejamento foram propostos ao longo dos tempos com o intuito de encontrar soluções automáticas para diferentes problemas [27]. O planejador clássico PRODIGY, derivado da sintaxe STRIPS, foi utilizado por Aler et. al. [1] para desenvolver SHAMASH, uma ferramenta que mais se aproxima da arquitetura proposta neste trabalho. Planejamento através de grafos, como o *GraphPlan* [9], utiliza um mecanismo de representação de intervalos de tempo e extração de planos em grafos. Planejadores por satisfabilidade, como *SATplan* [29] realizam verificação de satisfabilidade de expressões na FNC ou FND. O planejador *Blackbox* [28], sucessor do *SATplan*, combinava a abordagem de Satisfatibilidade com *GraphPlan*. Planejadores baseados em heurísticas, como *HSP* [10] e *FF* [24], dentre outros.

Outros planejadores surgiram ao longo do tempo, com o objetivo de adequar o planejamento às situações de incertezas em ambientes dinâmicos do mundo. Algumas abordagens destacam-se, tais como monitoramento e replanejamento, planejamento reativo, planejamento contingente e outras. As técnicas de abordagem reativa surgiram com o objetivo de introduzir a capacidade de selecionar uma ação, um exemplo é o *planejamento universal* [47], baseado na geração de uma árvore de decisão binária (BDDs). Técnicas de planejamento contingente condicional consideram a possibilidade de uma ação possuir efeitos disjuntivos, como o planejador *Warplan-C*, o planejador *Contingent-FF* [24], o planejador não-determinístico *POND*[3] [11].

Nesta seção serão apresentadas algumas das técnicas de planejamento em IA e que servem de referência para o restante do trabalho.

3.2.1 Buscas em Planejamento

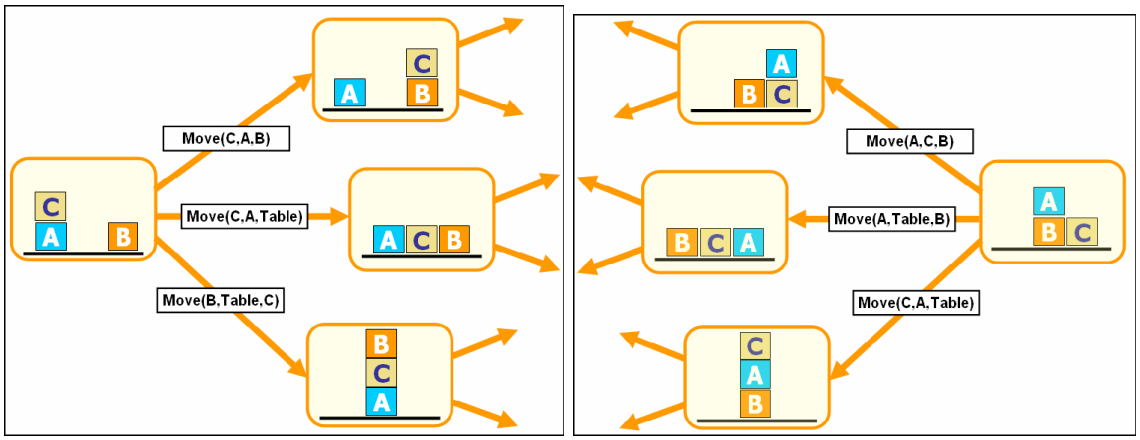
As abordagens principais para solução de um problema de planejamento podem ser a busca em espaço de estado (*state-space)* ou em espaço de plano (*plan-space*). A diferença entre ambas é que na busca em espaço de estado, o problema de planejamento pode ser representado como um grafo cujos vértices são os estados e as arestas são as ações. Já no espaço de plano, os vértices são formados por planos parciais e os ramos são procedimentos de refinamento desses planos.

As descrições das ações em um problema de planejamento especificam tanto pré-condições quanto efeitos, possibilitando fazer a busca em qualquer direção. Uma busca progressiva parte do estado inicial em direção a meta. Uma busca regressiva faz o sentido contrário, da meta para o estado inicial. Pode-se também fazer uso das representações da meta e das ações para produzir automaticamente heurísticas efetivas.

◦ Busca Progressiva

Planejamento utilizando busca com encadeamento progressivo, ou planejamento progressivo, é semelhante à abordagem dos resolvedores de problema. Começa-se com o estado inicial do problema e o algoritmo seleciona sucessivamente as ações que

[3]POND - Partially Observable Non-Deterministic Planner.



(a) Busca Progressiva (b) Busca Regressiva

Figura 3.2: Ilustração das duas abordagens de busca

conduzem ao estado meta, gerando assim uma seqüência de ações que será a solução do problema.

Esta busca pode ser muito ineficiente para resolver problemas de planejamento, desde que não distingue ações irrelevantes, considerando todas as ações aplicáveis em cada estado e não utiliza heurísticas, inviabilizando os planejadores em termos de tempo e memória, pois o processo torna-se exponencial.

◦ Busca Regressiva

Nos resolvedores de problemas, a busca regressiva pode ser de difícil implementação quando o estado da meta é descrito por um conjunto de restrições, e não por uma lista explícita de literais. Pode ser particularmente difícil gerar uma descrição dos possíveis predecessores do estado meta. Porém, a representação STRIPS torna este processo mais fácil, pois, conjuntos de estados podem ser descritos pelos literais que devem ser verdadeiros nesses estados.

A principal vantagem da busca regressiva é a possibilidade de considerar apenas ações relevantes. Uma ação é relevante para uma meta se seu efeito produz pelo menos um dos literais da meta. No exemplo da Figura 3.2(b), todas as três ações predecessoras produzem pelo menos uma pré-condição da meta.

Uma busca regressiva que permite ações irrelevantes poderá ser completa, mas será muito menos eficiente. A restrição para ações relevantes não altera a completude da busca, mas significa que a busca tem com freqüência um fator de ramificação

muito menor comparado com à busca progressiva.

Além de escolher apenas as ações que produzem os literais desejados, deve-se garantir que estas ações não desfaçam nenhum literal necessário. Uma ação que satisfaça esta restrição é chamada consistente.

A questão principal no planejamento regressivo é identificar o conjunto de estados no qual sob os quais a aplicação de uma ação produz um estado que contém a meta. Encontrar este conjunto de estados é chamado regressão da meta através da ação.

Qualquer algoritmo padrão de busca pode ser usado para realizar a busca. A busca finaliza quando a descrição de um predecessor é satisfeita pelo estado inicial do problema de planejamento.

◦ Heurísticas para Busca

As buscas regressiva ou progressiva não são eficientes devido à ausência de funções heurísticas. A chave para se determinar uma boa função heurística para direcionar o procedimento de busca é definir uma função que, de maneira rápida e simples, encontre o custo para alcançar a meta a partir de um determinado estado. Ou seja, medir a distância entre os estados inicial e final. No planejador STRIPS, o custo de cada ação é uniforme, ou seja, tem valor unitário. Assim, a distância é o número de ações.

O princípio básico consiste em, dados os efeitos das ações, a meta que deve ser alcançada, supor o número de ações necessárias para satisfazer todas as pré-condições da meta. Determinar o número exato é NP-Difícil [54] . Isto ocorre, porque a função tem a mesma complexidade inerente para solucionar o problema, pois se o custo de um caminho ótimo é conhecido, a solução para o problema de planejamento também o é. É possível encontrar uma estimativa razoável na maioria das vezes. O ideal seria criar uma heurística admissível - uma que não superestime - que forneça um custo inferior ao custo ótimo. Isto poderia ser usado com a busca A* [46] para obter soluções ótimas.

Duas abordagens diferentes podem ser tentadas. A primeira considera que um algoritmo baseado completamente na estratégia dividir-e-conquistar possa funcionar. Isto é chamado de hipótese de independência de submetas, o custo para resolver uma conjunção de submetas aproxima-se da soma dos custos de resolver cada submeta

independentemente. A hipótese de independência de submetas pode ser otimista ou pessimista. É otimista quando há interações negativas entre o subplanos para cada submeta - por exemplo, quando uma ação em um subplano remove uma meta alcançada por outro subplano. É pessimista, e então inadmissível, quando subplanos contêm ações redundantes - por exemplo, duas ações que poderiam ser substituídas por uma única ação no plano combinado.

A abordagem mais promissora é aquela que considera uma versão simplificada do problema de planejamento, isto é um problema relaxado. O custo da solução ótima para o problema relaxado fornece uma heurística admissível para o problema original. Como visto anteriormente, aplicar uma ação significa alterar o estado corrente pela inclusão dos literais positivos da sua lista de efeito e pela remoção dos negativos. Desde que representações explícitas de pré-condições e efeitos estão disponíveis, o processo trabalhará modificando essas representações.

A idéia mais simples é relaxar o problema removendo todas as pré-condições das ações. Então, toda ação sempre será aplicável, e qualquer literal pode ser alcançado em um passo. Se nenhuma ação pode ser aplicada então a meta é impossível de se atingir. Isto poderia implicar que o número de passos exigidos para resolver uma conjunção de metas fosse igual ao número de metas insatisfeitas, mas existem algumas situações que precisam ser consideradas:

- Pode haver duas ações, cada uma das quais remove o literal da meta alcançada pela outra;
- Alguma ação pode atingir metas múltiplas.

Todo problema relaxado é uma versão simplificada do problema original, mas não equivale ao mesmo, ou seja, sua solução não equivale à solução do problema original.

Também é possível gerar problemas relaxados removendo os efeitos negativos sem remover pré-condições. Significa que, se uma ação tem o efeito $A \wedge \neg B$ no problema original, terá o efeito A no problema relaxado. Desta forma não existe por que preocupar-se com interações negativas entre subplanos, já que nenhuma ação pode remover os literais atingidos por outra ação. Esta heurística é bastante precisa, mas calculá-la envolve executar de fato um algoritmo simples de planejamento. Mas,

a busca em um problema relaxado é com freqüência rápida o suficiente, que o custo torna-se viável.

As heurísticas descritas podem ser usadas nas buscas progressivas e regressivas. Até o momento os planejadores progressivos que utilizam a heurística baseada em ignorar a lista de remoção são os mais bem sucedidos. Porém, nenhum algoritmo será eficiente com todos os tipos de problemas. Mas, um número relativamente maior de problemas reais pode ser resolvido com os métodos de heurística apresentados, se comparado com a quantidade que podia ser resolvida até poucos anos atrás.

Apesar do risco de não produzir um plano completo ou ótimo, esta estratégia fornece um ganho de desempenho considerável, gerando planejadores extremamente rápidos, como a família de planejadores HSP [10] e FF [24].

Segundo Bonet et. al. [10], planejadores baseados em busca heurística podem ser caracterizados em três dimensões: a direção da busca (progressiva ou regressiva), o algoritmo de busca utilizado (com freqüência é alguma versão da busca-primeira ou subida de encosta), e o tipo de função heurística utilizada. A primeira versão do HSP, por exemplo, fazia uma busca progressiva no espaço de estados, utilizando um algoritmo de subida de encosta [46], e uma heurística não admissível derivada do problema relaxado, onde as listas de remoção eram ignoradas. O planejador FF é outro exemplo de planejador que faz a busca progressiva, mas utilizando-se de um algoritmo de subida de encosta diferente, assim como de uma heurística não admissível baseada em um problema relaxado, que por sua vez é baseado na abordagem do GraphPlan.

O procedimento utilizado pelo HSP em sua primeira versão é bastante simples. A cada passo são calculados os custos para cada sucessor possível na árvore de busca, um dos sucessores de menor custo é escolhido para a expansão no próximo passo. O processo repete-se até a obtenção do estado que contém a meta. Para melhor desempenho do planejador, o algoritmo de busca é incrementado para memorizar os nós já visitados, evitando-se assim um reprocessamento desnecessário, e para reiniciar a busca em outro estado quando um *mínimo local* for obtido [10].

No restante desta seção serão apresentadas algumas das principais técnicas de planejamento IA, com o objetivo de traçar uma visão deste conceito.

3.2.2 Planejador PRODIGY

O planejador Prodigy [38] foi desenvolvido no final dos anos oitenta por um projeto da Universidade de Carneggie Mellon. Esse trabalho preserva a idéia inicial dos operadores STRIPS, representando os estados de um mundo a partir de um conjunto de literais e definindo uma série de operadores (pré-condição↔ação↔pós-condições) aplicáveis a esses estados. A geração de planos pelo PRODIGY é baseada em busca com *backtracking*, realizada a partir de duas rotinas complementares: a simulação de planos e a regressão encadeada de estados. A simulação de planos é a aplicação de ações a partir do estado inicial, procurando atingir o estado final (tal qual o STRIPS). A Regressão é uma heurística que gera estados intermediários, a partir do conjunto de pré-condições das ações que supostamente geraram o estado final. Na geração do PRODIGY, uma busca em profundidade é realizada nos dois sentidos, do estado inicial ao final e vice−versa, como podemos ver na Figura 3.3.

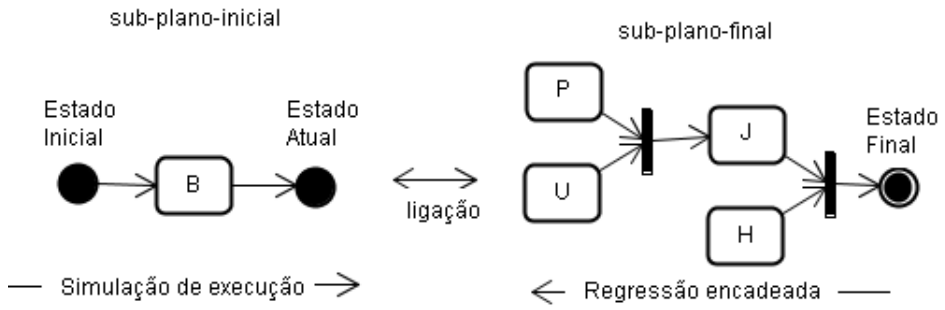


Figura 3.3: Geração planos Prodigy

Apesar de extremamente veloz se comparado aos planejadores de sua época, PRODIGY herdou do STRIPS alguns problemas em relação ao espaço de busca. Uma das principais dificuldades do Prodigy é a ordenação dos operadores que forma o sub-plano final. Uma vez que um desses operadores é movido ao sub-plano inicial, torna-se antecessor dos demais operadores do sub-plano final. Assim, quando dois ou mais operadores puderem ser satisfeitos no sub-plano inicial, o algoritmo deve escolher a ordem de aplicação entre esses operadores. Para garantir a completude, todas as combinações devem ser exploradas (backtracking), aumentando muito a complexidade do algoritmo para o pior caso [37].

3.2.3 Planejamento de Ordem Parcial (POP)

Com relação à ordenação de ações, os planos podem ser classificados como de ordem total ou de ordem parcial. Os procedimentos de busca, tanto progressiva quanto regressiva, são formas particulares da busca de *planos totalmente-ordenados*. Planejadores de ordem-total obtêm uma seqüência única de ações que conectam diretamente o estado inicial à meta, ou no sentido contrário. Isto significa que não podem tirar proveito da decomposição de problema em subproblemas, pois têm que tomar decisões sobre como ordenar as ações de todos os subproblemas. Seria preferível uma abordagem que tratasse independentemente várias submetas, resolvendo-as com diferentes subplanos, e então os combinando.

O plano gerado por um Planejador de Ordem Parcial é um conjunto de ações com os vínculos causais. Estes garantem que submetas já satisfeitas não sejam desfeitas durante o restante do planejamento e do refinamento do plano parcial em andamento. Muitas vezes, esta abordagem é confundida com o planejamento *não-linear*. A linearidade depende das interações entre as sub-metas, enquanto a ordem parcial refere-se à ordenação das ações em um plano.

Um planejador de ordem parcial usa a estratégia de comprometimento mínimo[4] em relação ao tempo, que significa não comprometer nada, até que seja realmente necessário, ou seja, atrasar uma escolha durante uma busca. Existe ainda o conceito de comprometimento-mínimo em relação à instanciação de variáveis, onde as variáveis são mantidas livres e não instanciadas até que alguma ação exija o contrário. Isto diminui o número de *backtrackings*.

O Planejamento de Ordem-Parcial pode ser definido como uma busca no espaço de planos, diferentemente das buscas progressiva ou regressiva, que utilizam um espaço de estado. No espaço de planos, os vértices do grafo de busca, representam planos parcialmente ordenados e as arestas representam operações de refinamento de planos, como a inclusão de uma ação num plano.

Inicia-se com um plano vazio e avalia-se uma forma de refinar o plano até que a solução do problema seja encontrada, ou seja, um plano completo. Assim, o estado inicial representa um plano vazio e a meta representa o plano completo, ou

[4]Tradução de *least commitment*

seja, a solução. Enquanto planejadores baseados na busca por estados retornam um caminho que forma a seqüência de ações, os baseados no espaço de planos retornam o vértice que contém o *estado-meta*, neste caso a meta é definida como o estado onde todas as ações sejam suportadas por ligações causais.

Goal(Sapato_Direito_Calçado ∧ Sapato_Esquerdo_Calçado)
Init ()
Action(Calçar_Sapato_Direito,
    PRECOND: Meia_Direita_Calçada
    EFFECT: Sapato_Direito_Calçada
Action(Calçar_Meia_Direita,
    EFFECT: Meia_Direita_Calçada
Action(Calçar_Sapato_Esquerdo,
    PRECOND: Meia_Esquerda_Calçada
    EFFECT: Sapato_Esquedo_Calçado
Action(Calçar_Meia_Esquerda,
    EFFECT: Meia_Esquerda_Calçada

Figura 3.4: Problema de planejamento descrito em ADL.

Um planejador que utiliza esta abordagem inicia sua busca a partir de um plano nulo, onde existem apenas duas ações, *início* e *fim*. A primeira não tem pré-condições e seus efeitos são os literais do estado inicial. A ação *fim* tem como pré-condição os literais do estado meta e como efeito uma lista vazia. Considere o problema clássico de calçar um par de sapatos. Podemos descrevê-lo como um problema de planejamento formal conforme a Figura 3.4.

Um planejador de ordem parcial é capaz de propor duas seqüências independentes de ações. Uma composta por *(Calçar_Meia_Direta, Calçar_Sapato_Direto)* e outra por *(Calçar_Meia_Esquerda, Calçar_Sapato_Esquerdo)*. Então as duas seqüências podem ser combinadas para se gerar o plano final ordenado. Fazendo isto, o planejador manipula as duas subseqüências independentemente, sem se comprometer se uma ação em uma seqüência está antes ou depois de uma ação na outra. A Figura 3.5 mostra o plano de ordem-parcial que é a solução do problema. Observe que a solução é representada como um gráfico de ações, não como uma seqüência única.

As ações "vazias"(início e fim), marcam o começo e o término do plano. A solução de ordem-parcial corresponde a seis possíveis planos de ordem-total; cada um destes é chamado de uma *linearização* do plano de ordem-parcial. O plano final da Figura 3.5 é mostrado a na Figura 3.6.

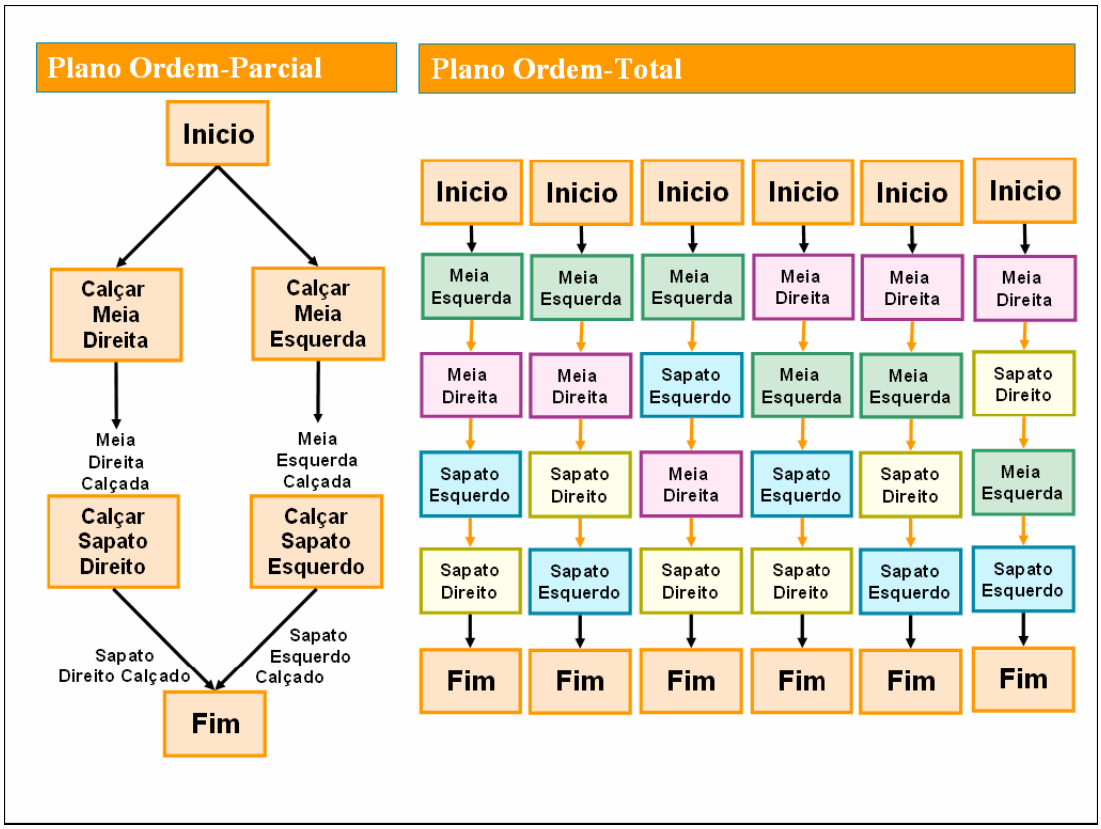


Figura 3.5: Plano de ordem-parcial e suas linearizações para o problema das meias e sapatos.

Como visto anteriormente, no planejamento de ordem-parcial é utilizado a busca por planos, onde cada estado refere-se a um plano (completo ou incompleto). Estes estados são descritos por um conjunto de ações, conjunto de restrições de ações, conjunto de vínculos causais, conjunto de pré-condições abertas, plano inicial, função sucessor e teste de meta.

Actions: { Calçar_Meia_Direita, Calçar_Sapato_Direito,
Calçar_Meia_Esquerda, Calçar_Sapato_Esquerdo,
início, fim}

Orderings: { Calçar_Meia_Direita $\prec$ Calçar_Sapato_Direito
Calçar_Meia_Esquerda $\prec$ Calçar_Sapato_Esquerdo

Links: { Calçar_Meia_Direita $\overset{Meia\_Direita\_Calcada}{\longrightarrow}$ Calçar_Sapato_Direito
Calçar_Meia_Esquerda $\overset{Meia\_Esquerda\_Calcada}{\longrightarrow}$ Calçar_Sapato_Esquerdo
Calçar_Sapato_Direito $\overset{Sapato\_Direito\_Calcado}{\longrightarrow}$ fim
Calçar_Sapato_Esquerdo $\overset{Sapato\_Esquerdo\_Calcado}{\longrightarrow}$ fim }

Open Preconditions: {}

Figura 3.6: Estado final do planejamento do problema das meias e sapatos.

3.2.4 Planejamento em Ambientes Dinâmicos

O planejamento clássico tem o inconveniente de trabalhar apenas com ambientes estáticos e determinísticos, ou seja, parte-se da premissa que todas as ações previstas serão executadas e que seus efeitos correspondem exatamente à previsão. Além disto, presume-se que o estado atual é modificado apenas pelas ações do plano, nenhuma outra modificação é introduzida durante a execução.

É óbvio que este cenário não corresponde ao mundo real, onde a presença de incertezas é regra, não a exceção. Desta forma, surgiram diferentes abordagens com o objetivo de adequar o planejamento às situações práticas. Existem algumas formas de classificar tais abordagens, tais como monitoramento e replanejamento, planejamento reativo, planejamento contingente e outras. Mas entre estas destaca-se o planejamento contingente condicional.

◦ Planejamento Contingente

O planejamento contingente é aquele no qual, planos com ramificações alternativas, podem ser gerados antecipadamente de acordo com situações, que possam ser antevistas. Tais situações devem ser descobertas durante a execução e com base no estado atual. São duas as abordagens principais para o planejamento contingente: o *planejamento condicional* e o *planejamento probabilístico*. Mas entre estas destaca-se o planejamento condicional.

No *planejamento condicional*, é utilizado o monitoramento do ambiente para

identificar situações não previstas ou desconhecidas nos estados anteriores, ou mesmo durante a criação do plano. Não são, portanto, uma seqüência única de ações, mas uma definição completa de várias linhas de execução possíveis para as situações que não se tem conhecimento completo sobre o resultado de uma ação ou do estado corrente do mundo.

O problema principal desta abordagem, é que, a complexidade para geração dos planos tende a crescer exponencialmente em relação direta com a quantidade de ações condicionais introduzidas. Porém, alguns trabalhos recentes têm obtido bons resultados nesta área. Alguns autores estão buscando estender as abordagens de planejamento clássico para o planejamento condicional. Dois exemplos, são as variações do planejador *FF*, conhecida por *Contingent-FF* [24] e o *POND*[5] [11].

O Planejador *POND* executa uma busca progressiva no espaço de crença utilizando o algoritmo *LAO** em uma representação de estados feitas em Diagramas de Decisão Binários (BDDs). A busca é guiada por uma heurística baseada em um Grafo de Registro de Incerteza (LUG[6]), que possibilita executar planejamento condicional. O Contingent-FF é um sistema de planejamento independente de domínio. O sistema extende o planejador clássico *FF* com a habilidade de tratar a incerteza no estado inicial e no efeito das ações, mas, com a capacidade de obter informações do estado corrente por ações de sensoriamento. Isto é feito extendendo a heurística do *FF* com a capacidade de tratar uma busca no espaço de crenças, por um raciocínio baseado na solução de uma FNC a qual representa a semântica da seqüência de ações.

O planejador contingente condicional POND será comparado ao planejador proposto neste trabalho. POND é um planejador não determístico que retorna todos os planos possíveis. O retorno de todos os planos possíveis é uma característica a ser tratada na comparação com o planejador proposto no trabalho. O objetivo é avaliar a eficiência da extração dos modelos; no caso específico medindo fluxos com ações seqüencias e diferentes ações condicionais.

[5]POND - Partially Observable Non-Deterministic planner.

[6]LUG - Labelled Uncertainty Graph.

3.3 Abordagem Evolutiva para Planejamento em IA

Nas seções anteriores apresentou-se algumas técnicas de planejamento e planejadores desde sua origem até as mais atuais. Apesar da grande variedade de técnicas, tem-se que, em geral, os sistemas de Planejamento em IA baseados na sintaxe proposta por STRIPS são compostos, em sua maioria, por métodos de busca que visam encontrar uma seqüência de ações que transforma o estado atual de um sistema no estado final desejado.

Apesar de alguns destes algoritmos serem geralmente muito rápidos e capazes de lidar com grandes instâncias de problemas, existem ainda problemas que não são tratados, principalmente aqueles para os quais a árvore de busca no espaço é extremamente grande. Uma técnica pouco explorada em Planejamento e que se mostra adequada para estes problemas é a Computação Evolutiva [46].

De acordo com a teoria da evolução, a capacidade de reprodução é o principal fator de evolução dos seres vivos. Por evolução entende-se, a adaptação de um indivíduo ao meio ambiente. Na adaptação por reprodução, incorre-se no fato dos seres vivos modificarem-se à medida que vão se reproduzindo (Darwinismo). A adaptação não ocorre de forma direta, ou seja, os indivíduos se modificam de acordo com as necessidades e essa modificações passam diretamente para os descendentes (Lamarkismo).

Os propósitos, a estrutura geral e os princípios de operação dos algoritmos evolutivos têm uma estrutura básica comum: realizam reprodução, impõem variações aleatórias, promovem competição e executam seleção de indivíduos de uma dada população. Sempre que estes quatro processos estiverem presentes, seja na natureza ou em uma simulação computacional, a evolução é o produto resultante. Em particular, ao tratar técnicas para a solução de problemas de otimização, vinculando o uso de computação evolutiva, o problema a ser resolvido faz o papel do ambiente, e cada indivíduo da população é associado a uma solução-candidata. Sendo assim, um indivíduo estará mais adaptado ao ambiente, sempre que corresponder a uma solução mais eficaz para o problema. Com a evolução, espera-se a cada geração obter soluções candidatas mais e mais eficazes. Neste contexto, um algoritmo evolutivo exerce o papel de um processo poderoso de busca iterativa e, em paralelo, adequada para o tratamento de problemas de otimização.

Na verdade, todo problema suficientemente complexo, a ponto de dificultar a produção de uma formulação matemática abrangente e o atendimento de requisitos básicos de tratabilidade por ferramentas convencionais, se transforma em um candidato para ser abordado a partir da computação evolutiva, já que a aplicação de técnicas de solução conhecidas, dedicadas e capazes de garantir a obtenção de uma solução ótima, não é possível nestes casos.

As abordagens evolutivas diferem em alguns aspectos, dentre os quais se destacam: estruturas de dados utilizadas para codificar um indivíduo, operadores genéticos empregados, métodos para criar a população inicial e métodos para selecionar indivíduos para a geração seguinte. Entretanto, compartilham o mesmo princípio comum: uma população de indivíduos sofre algumas transformações e durante a evolução os indivíduos competem pela sobrevivência.

Assim, nesta seção será apresentado uma breve descrição de duas técnicas de computação evolutiva, algoritmo genético e programação genética, com aplicação em problemas de planejamento.

3.3.1 Planejamento com Algoritmo Genético

Uma das técnica da Computação Evolutiva é o Algoritmo Genético, introduzido por Holland em 1975 [25] com o objetivo de formalizar matematicamente e explicar processos de adaptação em sistemas naturais e desenvolver sistemas artificiais (simulados em computador) que retenham os mecanismos originais encontrados em sistemas naturais.

AG's são sistemas adaptativos por excelência, ou seja, adaptam-se a determinadas condições que resultam na solução de um problema específico. Os algoritmos genéticos empregam uma terminologia originada da teoria da evolução natural, onde os indivíduos mais adaptados ao meio ambiente têm mais chances de sobreviver, e da genética, onde o conhecimento está armazenado no grupo genético do indivíduo, responsável pelas suas características.

O processo de evolução executado por AG's corresponde a um procedimento de busca em um espaço de soluções potenciais para o problema, que ocorre em diferentes pontos deste espaço simultaneamente, caracterizando esta busca como sendo em paralelo. Em geral, AG's são muito eficientes para busca de soluções

ótimas, ou aproximadamente ótimas em uma grande variedade de problemas, pois não impõem muitas das limitações encontradas nos métodos de busca tradicionais, além de realizar a busca em nível global.

Um AG padrão pode ser visto em Algoritmo 1. Na verdade, este é uma estrutura comum em técnicas de Computação Evolutiva. Neste procedimento genérico, tem-se uma população de indivíduos $P(t) = \{x_1, ...x_n\}$ na iteração (geração) t. Cada indivíduo representa um candidato à solução do problema e, em qualquer implementação computacional, assume a forma de alguma estrutura de dados. Cada solução x_i é avaliada e produz alguma medida de adaptação (fitness). Então, uma nova população é formada na iteração $t+1$ pela seleção dos indivíduos mais adaptados. Alguns indivíduos da população são submetidos a um processo de alteração por meio de operadores genéticos para formar novas soluções. Existem transformações unárias m_i (mutação) que criam novos indivíduos através de pequenas modificações de atributos em um indivíduo ($m_i : S \rightarrow S$), e transformações de ordem superior c_j (cruzamento), que criam novos indivíduos através da combinação de dois ou mais indivíduos ($c_j : S \times \ldots \times S \rightarrow S$). Após um número de gerações, a condição de parada, em geral, um número pré-determinado de gerações, deve ser atendida, indicando a existência, na população, de indivíduos que representem soluções aceitáveis para o problema.

Algoritmo 1 Procedimento evolutivo genérico

1: $t \leftarrow 0$
2: Inicialize $P(t)$
3: Avalie $P(t)$
4: **while não** condição parada **faça**
5: $\quad t \leftarrow t + 1$
6: $\quad$ Seleciona $P(t)$ a partir de $P(t-1)$
7: $\quad$ Altere $P(t)$
8: $\quad$ Avalie $P(t)$
9: **fim while**

O ponto de partida para a utilização do AG, como ferramenta para solução de problemas é a representação destes problemas de maneira que os AG possam atuar

adequadamente. Um indivíduo da população é representado por um cromossomo, que contém a codificação (genótipo) de uma possível solução do problema (fenótipo). Os atributos do cromossomo são conhecidos como gens. Os possíveis valores que um determinado gen pode assumir são denominados alelos. Um exemplo de codificação pode ser o proposto por Holland [25], onde as soluções candidatas são codificadas em arranjos binários de tamanho fixo, onde cada elemento de um vetor denota a presença (1) ou ausência (0) de uma determinada característica, isto é, o genótipo. Os elementos podem ser combinados formando as características reais do individuo ou o fenótipo.

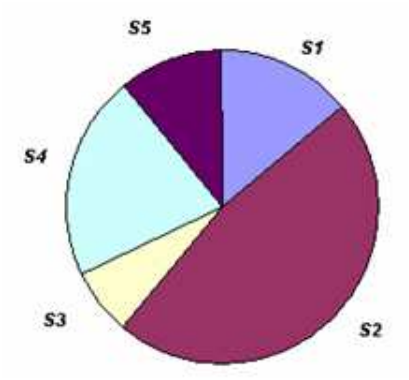
◦ **Mecanismo de Seleção**

Através da seleção, determinam-se quais indivíduos conseguirão se reproduzir, gerando um número determinado de descendentes para a próxima geração, com uma probabilidade determinada pelo índice de aptidão. Em outras palavras, os indivíduos com maiores valores de adaptação têm maiores chances de se reproduzir.

Um método de seleção muito utilizado é o Método da Roleta [35], onde indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através de um sorteio de roleta. Neste método, cada indivíduo da população é representado na roleta, proporcionalmente ao seu índice de aptidão. Assim, aos indivíduos com alta aptidão são dadas porções maiores da roleta, enquanto aos de baixa aptidão são dados porções relativamente menores. Finalmente, é girado a roleta por um determinado número de vezes, dependendo do tamanho da população, e são escolhidos aqueles sorteados como indivíduos que participarão da próxima geração. Exemplo do método da roleta pode ser visto na Figura 3.7.

◦ **Operadores Genéticos**

Um conjunto de operações é necessário para que, dada uma população, seja possível gerar populações sucessivas que (espera-se) melhorem a aptidão. Os operadores cruzamento e mutação são utilizados para assegurar que a nova geração tenha novos indivíduos, mas possa, de alguma forma, também preservar características dos ancestrais, ou seja, a população se diversifica e mantém características de adaptação adquiridas pelas gerações anteriores. Para prevenir que os melhores indivíduos desa-



Individuo (Si)	Aptidão f(Si)
S1	2,23
S2	7,27
S3	1,05
S4	3,35
S5	1,69

Figura 3.7: Exemplo do Método da Roleta

pareçam da população pela manipulação destes operadores, utiliza-se a reprodução elitista, um operador que mantém nas gerações seguintes, os indivíduos mais adaptados.

∗ **Operador Mutação**

O operador de mutação é necessário para a introdução e manutenção da diversidade genética na população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, e assim, introduzindo novos elementos na população. Considerando codificação binária, o operador de mutação padrão simplesmente troca o valor de um gen em um cromossomo [25]. Assim, se um gen selecionado para mutação tem valor 1, o seu valor passará a ser 0 após a aplicação da mutação, e vice−versa, como é ilustrado na Figura 3.8.

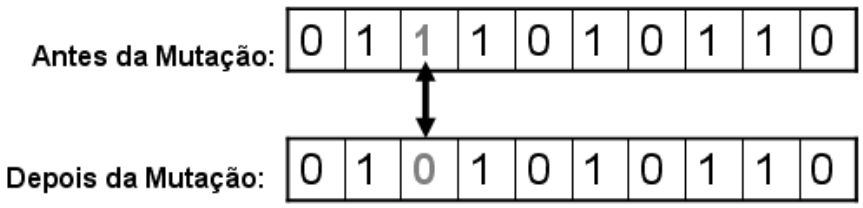


Figura 3.8: Operador de Mutação

Desta forma, a mutação assegura a probabilidade de atingir qualquer ponto do espaço de busca, além de contornar o problema de mínimos locais, pois com este mecanismo, altera-se levemente a direção da busca. O operador de mutação é aplicado aos indivíduos com uma probabilidade dada pela taxa de mutação, que em

geral é baixa.

∗ **Operador Cruzamento**

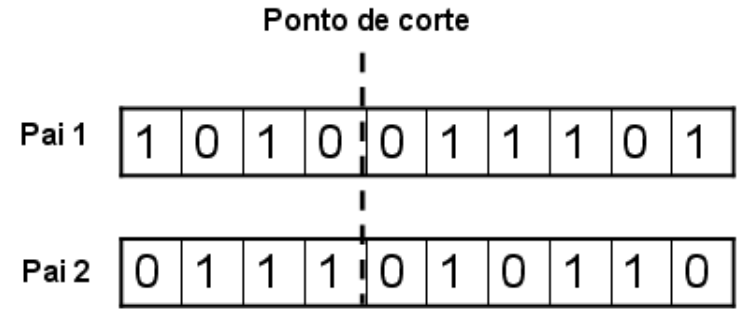
O cruzamento é o operador responsável pela recombinação de características dos ancestrais durante a reprodução, permitindo que as próximas gerações herdem essas características. Ele é considerado o operador genético predominante, por isso, é aplicado com probabilidade dada pela taxa de cruzamento, que deve ser maior que a taxa de mutação.

O operador de cruzamento mais comumente empregado é o cruzamento de um ponto. Para a aplicação deste operador, são selecionados dois indivíduos (pais) e a partir destes são gerados dois novos indivíduos (filhos). Um ponto de cruzamento é escolhido nos cromossomos dos pais, e os segmentos do cromossomo até este ponto de cruzamento são trocados. Considere, por exemplo, os indivíduos *Pai 1* e *Pai 2* e suponha o ponto de cruzamento escolhido (aleatoriamente) entre as posições 4 e 5, após o cruzamento, os indivíduos *Filho 1* e *Filho 2* são gerados, conforme Figura 3.9.

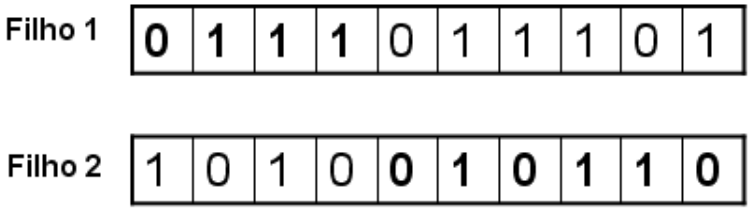
Existem outros tipos de cruzamento propostos na literatura. Uma extensão simples do cruzamento de um ponto é o cruzamento de dois pontos, onde dois pontos de cruzamento são escolhidos e o material genético são trocados entre eles. Tem-se também cruzamento uniforme [53], para cada *gen* no primeiro filho é decidido (com alguma probabilidade fixa p) qual pai vai contribuir com seu valor para aquela posição. Como o cruzamento uniforme troca *gens* ao invés de segmentos de *gens*, é possível combinar características independentemente da sua posição relativa no cromossomo.

∗ **Parâmetros para AG**

É importante também, analisar de que maneira alguns parâmetros influem no comportamento de um AG, para que se possa estabelecê-los conforme as necessidades do problema e dos recursos disponíveis. Em geral, o tamanho da população afeta o desempenho global e a eficiência. Com uma população pequena, o desempenho pode ser prejudicado, pois deste modo, a população fornece uma pequena cobertura do espaço de busca do problema. Uma grande população, geralmente, fornece uma



(a) Pais



(b) Filhos

Figura 3.9: Operador Cruzamento

cobertura representativa do domínio do problema, além de prevenir convergências prematuras para soluções locais ao invés de globais.

No entanto, grandes populações, exigem maiores recursos computacionais ou tempo de execução maior. Referente à taxa de cruzamento, quanto maior esta taxa, mais rapidamente novas estruturas serão introduzidas na população. Mas, se esta for muito alta, estruturas com bons valores de aptidão poderão ser retiradas mais rapidamente, ou seja, a maior parte da população será substituída. Com um valor baixo, o algoritmo pode tornar-se muito lento. A taxa de mutação previne que uma dada posição se torne estagnada em um valor, além de possibilitar que se atinja qualquer ponto do espaço de busca. Porém, com taxas muito altas a busca se torna essencialmente aleatória. A taxa de elitismo preserva os melhores indivíduos para a próxima geração da população.

∗ **Elementos de Planejamento em AG**

Devido às características presentes no Algoritmo Genético [39], esta técnica tem sido amplamente pesquisada e utilizada em uma grande variedade de problemas,

como por exemplo problemas de Planejamento [7]. Uma possível representação de AG em Planejamento seria através de uma lista de tamanho fixo, onde cada gen representaria uma ação do plano e o cromossomo represente um plano propriamente dito. No início da execução do AG os arquivos com a descrição do domínio do problema são lidos e uma estrutura de representação é construída na memória, contendo: o estado inicial, as constantes do problema e uma lista com todas as ações possíveis para o domínio. Qualquer seqüencia de ações desta lista seria um plano em potencial e, portanto, uma possível solução do problema. O AG então faz uma busca por uma seqüencia de ações da lista que resolva o problema de planejamento.

Lecheta apresenta em seu trabalho [32] uma representação para adaptar algoritmos genéticos em Planejamento, onde o cromossomo é uma lista de tamanho fixo, e cada gen corresponda a um valor inteiro referente a uma determinada ação. O problema de planos de tamanho variável é evitado pela definição de uma cota superior para o tamanho dos planos e a inclusão de ações NOP´s (ações sem efeito sobre o estado do mundo e que podem ser aplicáveis a qualquer instante num plano), até que todos tenham o mesmo tamanho, resultando num AG com cromossomos de tamanho fixo. Exemplo da representação de um cromossomo para um plano de 20 ações pode ser visto na Figura 3.10.

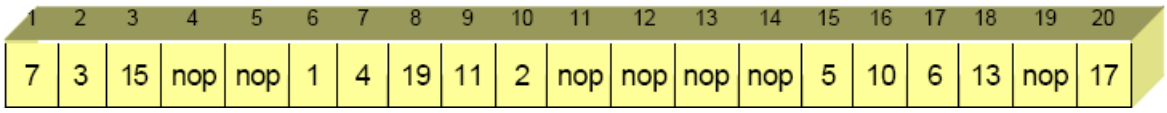


Figura 3.10: Representação de um cromossomo para um plano de 20 ações

Quanto à questão da geração da população inicial, Lecheta considerou para a codificação, a geração aleatória e os valores dos gens escolhidos, aleatoriamente, entre zero e o número de ações possíveis para o problema. Utilizou-se operador padrão de cruzamento de dois pontos e no operador de mutação escolheu-se aleatoriamente um valor para o gen selecionado uniformemente. Na análise da execução das ações de um plano representado pelo cromossomo de inteiros, tem-se um procedimento que cria um cromossomo paralelo de binários, onde o *gen* "1"representa uma ação

[7] A aplicação de Computação Evolutiva em Planejamento é comumente conhecido como Planejador Genético

executável e o *gen* "0"uma ação que não é executável nas suas respectivas posições, eliminando as ações nulas "NOP". Na avaliação do indivíduo, algumas métricas são consideradas, como por exemplo, o número da Primeira Ação Executável do plano (PAE), o número da Última Ação Executável do plano (UAE), o número de Ações Executáveis (AE), o número de Ações Consecutivas Executáveis (ACE), o número de Ações Válidas (AV), o número de Objetivos Alcançados (OA), o número Total de Objetivos (TO), a representação binária do cromossomo em relação a executabilidade excluidos os NOPs. O trabalho de Lecheta apesar de não se mostrar competitivo em relação ao tempo de processamento, mostrou uma contribuição na integração das técnicas de Planejamento e AG.

3.3.2 Planejamento com Programação Genética

Outra técnica da Computação Evolutiva é a Programação Genética (PG). PG é uma técnica automática de programação desenvolvida por Koza [31], baseado nos trabalhos de John Holland [25] em AG's. Desta maneira, um algoritmo de PG tem a mesma estrutura do Algoritmo 1. A principal diferença entre AG e PG são as estruturas para representação dos cromossomos. Assim, enquanto um AG convencional manipula cadeias de cromossomos que codificam soluções de problemas, PG manipula programas de computador que são candidatos à solução de um determinado problema.

Cada programa é um indivíduo armazenado em uma representação hierárquica. A função de aptidão é usada para descrever a qualidade dos programas. Os programas produzem uma saída desejada baseada em uma entrada e a diferença entre o esperado e o produzido na saída pode ser usada para avaliar a aptidão dos candidatos em questão.

O mecanismo de busca da Programação Genética pode também ser visto como um ciclo "criar-testar-modificar"(Figura 3.11), muito similar à forma com que os humanos desenvolvem seus programas. Inicialmente, programas são criados baseados no conhecimento sobre o domínio do problema. Em seguida, são testados para verificar sua funcionalidade. Se os resultados não forem satisfatórios, modificações são feitas para melhorá-los. Este ciclo é repetido até que uma solução satisfatória seja encontrada ou um determinado critério seja satisfeito [63].

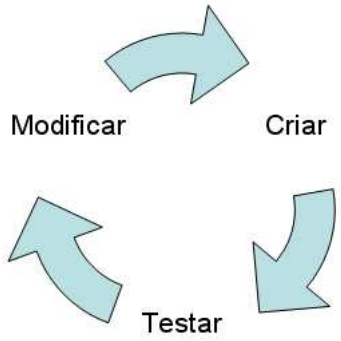


Figura 3.11: Ciclo criar-testar-modificar

Por manipular programas diretamente, PG lida com uma estrutura relativamente complexa e variável. Tradicionalmente, esta estrutura é uma árvore de sintaxe abstrata composta por funções nos nós internos e terminais nos nós-folha. A especificação do domínio do problema é feita simplesmente pela definição dos conjuntos de funções e terminais [31].

Parte-se do conjunto de funções F e do conjunto de terminais T. F pode conter operadores aritméticos (+, -, * etc.), funções matemáticas (seno, log etc.), operadores lógicos (AND, OR etc.) dentre outros. Cada função $f \in F$ tem associada uma aridade (número de argumentos) superior a zero. O conjunto T é composto pelas variáveis, constantes e funções de aridade zero (sem argumentos). Por exemplo, considerando $F = \{+, -, *, /\}$ e $T = \{x, 2\}$, expressões matemáticas simples tais como $(x * x) + 2$ podem ser produzidas. A representação é feita por uma árvore de sintaxe abstrata como mostrado na Figura 3.12

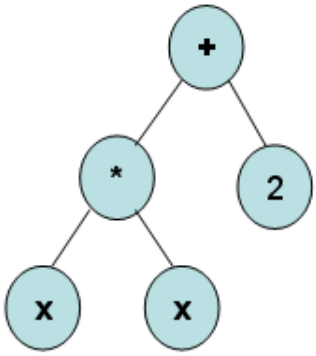


Figura 3.12: Representação indivíduo através de árvore

O primeiro passo do algoritmo consiste na criação de uma população aleatória

de programas. A população inicial é composta por árvores geradas aleatoriamente a partir dos conjuntos F e T. Inicialmente, escolhe aleatoriamente uma função $f \in F$. Para cada um dos argumentos de f, escolhe-se um elemento de $\mathrm{F} \cup \mathrm{T}$. O processo prossegue até que se tenha apenas terminais como nós-folha da árvore. Geralmente, especifica-se um limite máximo para a profundidade da árvore.

Cada programa é executado, e então recebe um valor de função de aptidão. A avaliação de aptidão depende do domínio do problema e pode ser medida de diversas formas, tanto direta quanto indiretamente. Em geral, para se proceder à avaliação de aptidão, é fornecido um conjunto de casos de treinamento, denominados *casos de aptidão*, contendo valores de entrada e saída a serem aprendidos. A cada programa é fornecido os valores de entrada e confronta-se a sua resposta ao valor esperado de saída. Quanto mais próxima a resposta do programa estiver do valor de saída, melhor é o programa.

O operador genético de cruzamento consiste na troca de fragmento entre dois programas selecionados na população. Dois programas são selecionados e são recombinados para gerar outros dois programas. Um ponto aleatório de cruzamento é escolhido em cada programa-pai e as árvores abaixo destes pontos são trocadas. Um exemplo de cruzamento pode ser visto na Figura 3.13. Neste exemplo, foram escolhidos os programas: $((2 * (x + x)) + 1)$ e $(((x + 1) * x) - 2)$. Foram escolhidos aleatoriamente um nó em cada árvore. As árvores são então trocadas, gerando os novos programas: $((x + 1) + 1)$ e $(2 * ((x + x) * x) - 2$.

Sistemas de PG produzem programas e Sistemas de Planejamento em IA tradicionais produzem planos. Planos e programas apresentam similaridades, pois ambos podem ser considerados como uma ordenação de um conjunto de instruções. Planejamento com Computação Evolutiva é conhecido como Planejamento Genético. Neste caso, Planejamento Genético utiliza planos seguindo o mesmo algoritmo usado em PG. Planejamento Genético pode evoluir uma população de planos para alcançar um conjunto de metas. O sistema de planejamento genético pode realizar simulações para averiguar a aptidão dos indivíduos, avaliando, por exemplo, a utilidade de seqüencias de ações [21], analisando declarações de estruturas que descrevem os efeitos dos operadores, podendo verificar também a quantidade de metas e submetas alcançadas.

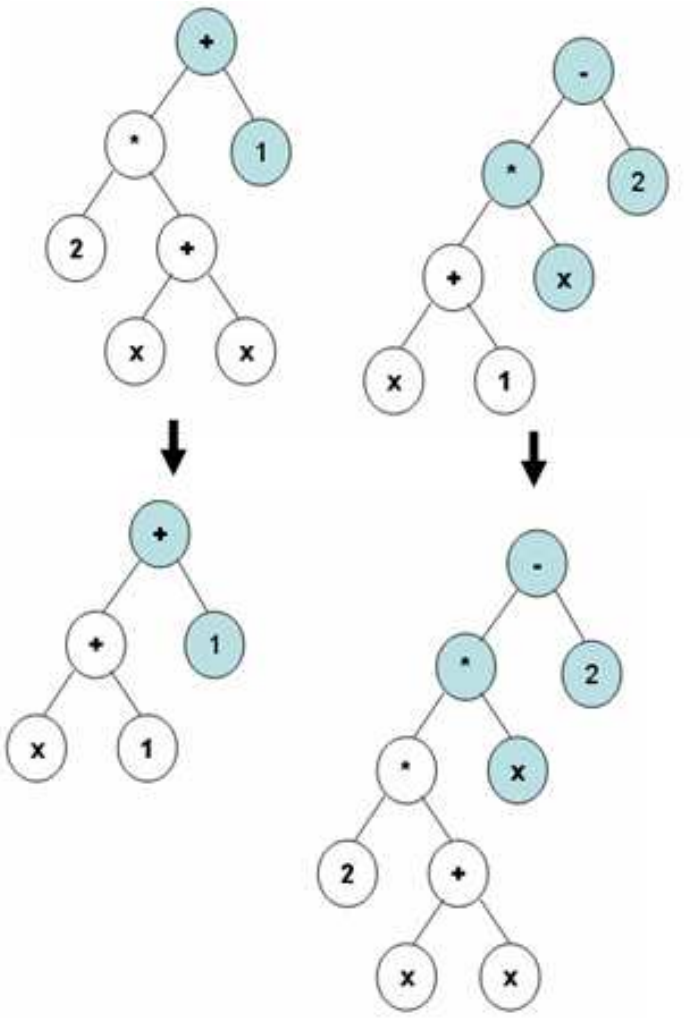


Figura 3.13: Operação genética de Cruzamento PG

A técnica de PG vem sendo aplicada a problemas de planejamento em IA, por exemplo, em Levine et. al [59] é apresentada a ferramenta *Genplan*, que faz uma combinação de programação genética e planejamento aplicados em problemas clássicos da literatura. Aler et. al [2] usa programação genética para evoluir heurística para o planejador (Prodigy). Muslea [42] apresenta SINERGY um planejador linear baseado em programação genética para gerar planos lineares que resolvem metas conjuntivas. Spector [48] descreve uma séries de ilustrações e experimentos na aplicação de PG ao tradicional domínio do "mundo dos blocos".

Uma das representações exemplificadas por Spector para o domínio específico do mundo dos blocos será descrita a seguir. O conjunto terminal é constituído por $CS(sensors)$, que dinamicamente especifica o bloco no topo da pilha; TB (*Top*

Correct Block), que especifica o bloco mais alto na pilha tal que este e todos os blocos abaixo estão na ordem correta; *NN* (Next Needed), que especifica o bloco que deveria estar no topo de *TB* no final da pilha.

As funções são constituídas por, *MS* (*Move to the Stack*), que pega e move um bloco da mesa para a pilha e retorna True, caso contrário NIL; *MT* (*Move to the Table*) pega e move um bloco no topo da pilha para a mesa retorna True, caso contrário NIL; *DU* (Do Until) uma estrutura de controle da linguagem LISP e *EQ* predicado de igualdade em LISP.

Em um estado inicial deste problema onde o bloco C está sobre o bloco A, e os blocos A e B estão sobre a mesa, uma meta poderia ser um estado em que o bloco A está sobre o bloco B, este está sobre o bloco C, que por sua vez está sobre a mesa. A representação gráfica dos estados inicial e final deste exemplo pode ser vista na Figura 3.14. Uma representação de um programa exemplificadas por Spector em LISP que representa o domínio para este problema poderia ser *(EQ (DU (MT CS) (NOT CS)) (DU (MS NN) (NOT NN)))*. A Figura 3.15 mostra a representação do programa graficamente.

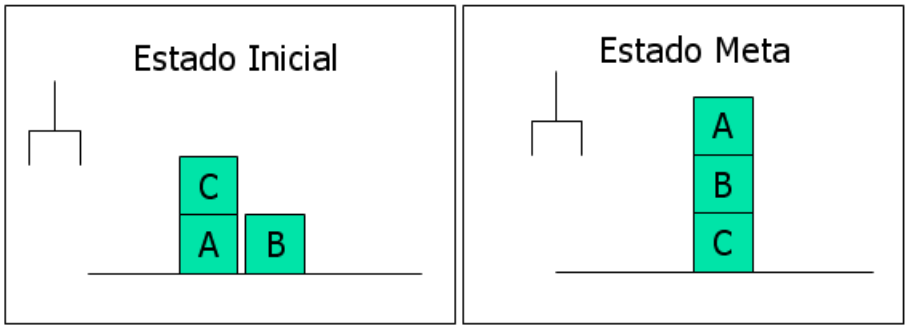


Figura 3.14: Exemplo representação do problema do mundo dos blocos

3.4 Considerações finais

A vantagem mais significativa da Computação Evolutiva está na possibilidade de resolver problemas pela simples descrição matemática da solução, não havendo necessidade de se indicar explicitamente os passos até o resultado, que certamente seriam específicos para cada caso. É lógico que os algoritmos evolutivos correspondem a uma seqüência de passos até a solução, mas estes passos são os mesmos para uma ampla gama de problemas, fornecendo robustez e flexibilidade.

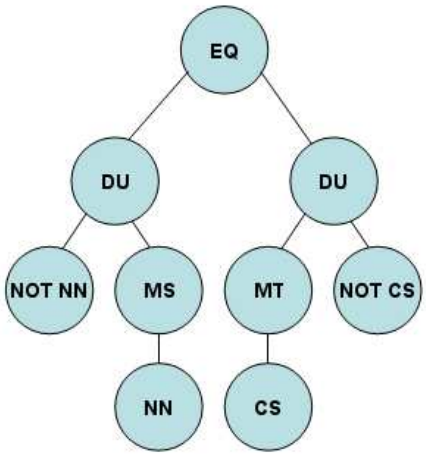


Figura 3.15: Representação gráfica programa do domínio do mundo dos blocos em PG

Sendo assim, a computação evolutiva deve ser entendida como um conjunto de técnicas e procedimentos genéricos e adaptáveis, a serem aplicados na solução de problemas complexos, para os quais outras técnicas conhecidas são ineficazes ou nem sequer são aplicáveis. A grande diversidade de domínios de problemas existentes amplia o grau de complexidade dos sistemas.

O uso de técnicas Planejamento Genético se mostra bastante interessante no cenário de workflow, pois apresenta características satisfatórias para minimizar os efeitos da complexibilidade e dos grandes espaços de busca. Outro fator relevante é o relaxamento da restrição de otimização da solução, presente em vários planejadores atuais, o que facilita a busca pela solução em técnicas genéticas.

Capítulo 4

Geração automática de processos em Workflow

Como visto anteriormente, uma característica importante a ser considerada em problemas de planejamento é a complexidade reconhecidamente difícil, tendo sido provada ser PSPACE-completo [54]. O espaço de busca, além de ser exponencial, geralmente apresenta um crescimento extremamente rápido, o que torna a sua aplicação em domínio de Workflow bastante onerosa devido à diversidade ao tamanho das instâncias destes problemas.

A associação de técnicas de Planejamento IA e AG se mostra adequada para problemas de Workflow. A primeira técnica tem muitas semelhanças com workflow e a segunda trata de forma satisfatória problemas com grande espaço de busca. Outro fator do tratamento de situações genéricas de Planejamento a partir de AG é o relaxamento de restrições do problema de otimização, presentes em vários planejadores atuais, o que facilita a busca pela solução do AG.

Este trabalho apresenta características da proposta de Aler et. al. [40], entretanto nele utiliza-se um planejador genético. Planejadores genéticos se caracterizam por realizar uma busca global no espaço de soluções, diferentemente dos planejadores clássicos que realizam esta busca localmente. Esta característica permite aos planejadores genéticos manipularem problemas de grandes dimensões. Outra importante característica é a flexibilidade de representação que os AG's oferecem, permitindo considerar aspectos tal como fluxo condicional e paralelo entre atividades, que não são muito explorados em trabalhos anteriores e são importantes quando se considera

modelagem de processos.

Assim, este capítulo apresenta uma arquitetura denominada "Workflow Genético", utilizada para a geração automática de modelos de processo em sistemas de workflow. A geração está baseada em um planejador genético que identifica e gera automaticamente modelos de processos de acordo com as atividades disponíveis. A seção 4.1 apresenta um mapeamento entre Planejamento em IA e Workflow, descrevendo características da união entre estas duas técnicas, problemas apresentados e possíveis soluções propostas. A arquitetura do Workflow Genético é apresentada na seção 4.2. Os módulos desta arquitetura, Planejador Genético e Analisador, responsáveis pela geração automática de modelos de processos são descritas nas seções 4.3 e 4.4, respectivamente. Na seção 4.5 é realizado uma avaliação da extração de modelos entre o Workflow Genético e o planejador POND, medindo fluxos com ações seqüencial e diferentes ações condicionais.

4.1 Modelagem de Workflow através de Planejamento

A aplicação de Planejamento ao domínio de workflow toma como ponto de partida o modelo do planejamento clássico. Planejamento em IA busca descobrir uma sucessão de ações para atingir a meta, ou seja, o próprio plano [46]. As definições de planejamento em IA apresentadas no relatório do PLANET [45] complementam sutilmente esta definição, introduzindo o conceito de processo. O processo seria uma descrição de um conjunto de atividades que são executadas para a realização de um trabalho. Já o plano, seria a descrição de uma seqüência de atividades para atingir um dado objetivo, isto é, um processo instanciado. Estas definições são complementares e introduzem o processo como o conjunto de todos os planos válidos.

4.1.1 Mapeamento Workflow-Planejamento

As correspondências entre Planejamento e Workflow podem ser vistas na tabela 4.1, na qual nota-se que uma instância de processo sendo executada (caso) é considerada um plano em Planejamento. A execução de uma atividade em Workflow é vista como uma ação no Planejamento. Um recurso de Workflow (pessoas, aplicações, máquinas) é visto como um agente executor em Planejamento. Esse mapeamento

permitiu a aplicação das técnicas Planejamento a determinadas etapas do sistema de Workflow, guiando esforço de pesquisas subseqüentes.

Workflow	Planejamento
Caso	Plano
Atividade	Ação
Recurso	agente execução

Tabela 4.1: Workflow e Planejamento

Dadas as semelhanças entre o planejamento e a modelagem de workflow, pode-se buscar caminhos para aplicar as estratégias do primeiro na execução do segundo. Tais semelhanças repousam no sequenciamento de atividades e em suas execuções. Ambas as abordagens produzem uma seqüencia de passos para a solução genérica de instâncias de um problema particular dentro do mesmo domínio. Esta seqüencia de passos é uma lista de atividades para o workflow e uma seqüência de ações para o planejamento. Ambos tentam conduzir à meta.

Recentemente, alguns trabalhos foram apresentados com a perspectiva de integração das técnicas de Workflow e Planejamento. Myers et. al. [43], por exemplo, apresentam as correspondências entre os dois campos e as contribuições possíveis trazidas pela adoção das técnicas. Berry et. al. [8] apresenta o esboço de um sistema de Workflow com controle reativo baseado em IA, chamado SWIM, o qual busca extender o paradigma de Workflow para responder a ambientes dinâmicos e com incertezas. Moreno [41] apresenta um estudo de caso que explora os ganhos que um sistema de gerenciamento de Workflow pode obter incorporando técnicas de Planejamento contingente utilizando o planejador Cassandra. Aler et. al. [1] apresentam o SHAMASH, uma ferramenta para re-engenharia de processos de negócio, cujas funcionalidades incluem a definição e uso de padrões organizacionais e simulação e otimização dos modelos.

4.1.2 Características da união de Workflow e Planejamento

Embora o uso da técnica de planejamento permita ao usuário economizar muito tempo na modelagem de processo em sistema de workflow, há ainda muitas melhorias que podem ser feitas [40]. Algumas características levantadas são:

- Falta de uma flexibilidade de representação dos operadores de planejamento em linguagens de definição de processos, assim como suas regras de produção.
- Aspectos como fluxo paralelo e condicional não são muito explorados em trabalhos prévios e que são importantes quando consideramos modelagem de processos em workflow.
- O espaço de busca da solução apresenta um crescimento extremamente rápido, além de ser exponencial, devido principalmente ao tamanho das instâncias dos problemas de workflow e a sua diversidade;

Em seguida são detalhadas algumas das características levantadas acima com suas respectivas propostas de soluções no presente trabalho.

◦ **Representação - Atividades versus Operadores**

Em planejamento, os operadores representam as ações que poderão ser tomadas pelos executores. Os operadores especificam pré-condições e efeitos para cada ação. Na modelagem de processos de workflow estes conceitos não estão explicitamente presentes na linguagem, por isso é proposto o uso de atributos de extensão para representar estas regras. São propostos dois atributos um descrevendo as pré-condições e os efeitos. O uso destes atributos assemelha-se às regras de produção, onde as pré-condições são o antecedente; e os efeitos o conseqüente. Isto possibilitará ao Modelador do processo, no momento de cadastrar as atividades, informar as condições para sua execução, bem como, os efeitos causados pela mesma. Cabe ao desenhista do processo informar ao sistema todas as atividades possíveis de execução no domínio, e as regras de execução das mesmas.

Na linguagem de definição de processos adotada (XPDL), as informações dos atributos da regra de produção serão informados através do campo $< Extended\ Attribute >$ de cada atividade correspondente, como pode ser visto através da representação da atividade na Figura 4.1. O campo $< ExtendedAttribute >$ permite uma flexibilidade de representação podendo ser aplicado em diferentes domínios do conhecimento. A representação correspondente em PDDL da atividade na Figura 4.1 é mostrada na Figura 4.2.

$< ActivityId = "coletar" Name = "$**Coletar**$»$

$< Description >$

$Coletar\ informações\ reclamação\ telefônica < /Description >$

$< Performer > System < /Performer >$

$< ExtendedAttributes >$

$< ExtendedAttribute >$

$<$ ***Regra Produção*** $>$

$<$ ***Pré-Condições and(departamento_Contactada)(cliente_Contactado)*** $>$

$<$ ***Pós-Condições=(coleta_Realizada)*** $>$

$\vdots$

$< /ExtendedAttribute >$

$< /ExtendedAttributes >$

$< /Activity >$

Figura 4.1: Atividades XPDL

(action: Coletar

precondition: and (departamento_Contactada)(cliente_Contactado)

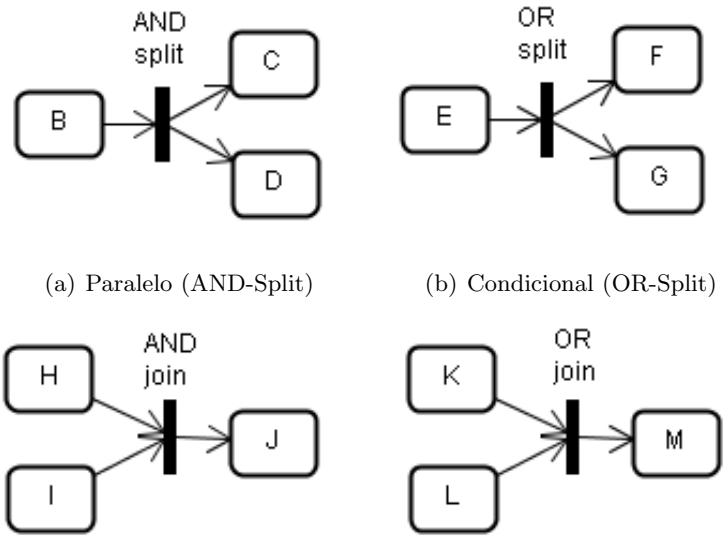
effect : (coleta_Realizada))

Figura 4.2: A ação PDDL

◦ **Conexões entre Atividades**

Considerando o planejamento clássico, as ações em um plano conectam-se entre si de forma seqüencial. Na modelagem de processos de workflow, uma atividade geralmente conecta-se a outra seqüencialmente. Mas existem situações onde, podem ocorrer tomadas de decisão ou paralelismo.

As atividades envolvendo roteamentos podem ser de dois tipos: separação ou junção (Split/Join). Os roteamentos de separação (Split) estão na saída da atividade corrente e podem direcionar para uma execução paralela ou condicional (AND / OR). Na execução paralela, duas ou mais atividades podem ser habilitadas ao mesmo tempo pela máquina de workflow. O roteamento condicional obriga a escolha de um caminho para a execução. Os roteamentos de junção (Join) são de forma semelhante, mas uma entrada do tipo OR necessita apenas ser atingida por um dos ramos, enquanto as do tipo AND precisam ser atingidas por todos os ramos para que a atividade seja habilitada.



(a) Paralelo (AND-Split) (b) Condicional (OR-Split)

(c) União total (AND-Join) (d) União interativa ou Paralelo com seleção (OR-Join)

Figura 4.3: Modelo de roteamento de atividades

A Figura 4.3 mostra exemplos dos tipos de roteamento de atividades presentes no contexto deste trabalho. Em 4.3(a), o fluxo paralelo permite que duas ou mais atividades podem ser executadas simultaneamente e independentemente, são disparadas em determinado ponto do processo e resincronizadas mais adiante. Temos que, tanto as atividades C e D são ativadas simultaneamente após o término da atividade B. Em 4.3(b), o fluxo condicional habilita apenas uma das atividades subseqüentes. Apenas F ou G são ativadas após o término de E. Em 4.3(c), é realizado um sincronismo para a atividade do tipo AND-Join. A atividade J só pode ser executado após a finalização das atividades H e I. Em 4.3(d), é realizado um sincronismo para a atividade do tipo OR-Join. A atividade M pode ser executado após a finalização de K ou L.

4.2 Arquitetura do Workflow Genético

A arquitetura do Workflow Genético é esboçada na Figura 4.4. A primeira etapa *mapeamento atividade(atividade/ação)* (2) consiste em receber as informações das atividades disponíveis em um repositório (1), e que estão no formato XPDL, e gerar

o equivalente em PDDL.

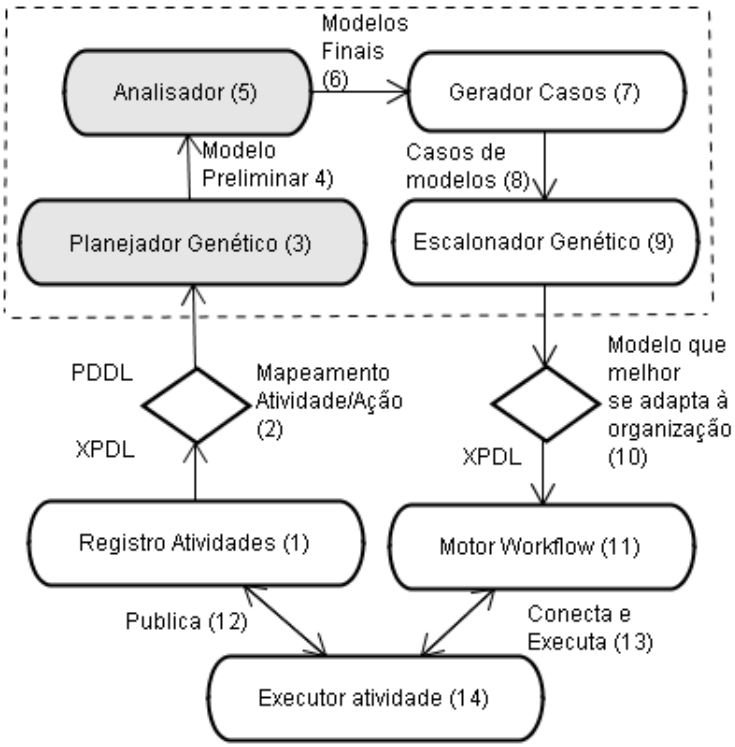


Figura 4.4: Arquitetura do Workflow Genético

O módulo Planejador Genético (3) gera um modelo preliminar contendo uma representação unificada de possíveis combinações de modelos (4) de processos. Na geração destas combinações todas as atividades unificáveis, ou seja, todas atividades cujas pré-condições e pós-condições são satisfeitas em determinado estágio do modelo são consideradas na representação. O Planejador Genético utiliza técnicas de planejamento combinados com AGs para gerar as possíveis combinações de modelos de processo.

O Planejador Genético permite evoluir representações de modelos de processos preocupando-se em garantir a corretude do modelo, considerando não somente roteamento seqüencial, mas o uso de roteamentos condicionais e paralelos. Todos estes roteamentos são tratados, devido ao uso do paradigma de algoritmo genético e um planejador de objetivo geral, possibilitando considerar problemas de vários domínios do mundo real.

Atividades ou blocos de atividades que executam o mesmo papel definem pontos de conflito e indicam que é possível ter mais de um modelo de processo para o

problema. Assim, o módulo *Analisador* (5) tem a responsabilidade de extrair os modelos finais, ou seja, os possíveis modelos de processos válidos. O *Analisador* extrai os modelos finais considerando atividades condicionais (OR), paralelas (AND) e seqüenciais (SEQ).

Cada modelo de processo válido (6) juntamente com as informações dos recursos associados a cada atividade e seus respectivos tempos de execução podem ser encaminhados para um *Simulador*, que consiste nos módulos *Gerador de Casos* e *Escalonador Genético*.

A funcionalidade do *Gerador de Casos* (7) é gerar os possíveis casos existentes, ou seja, as possíveis rotas do modelo de processo. Na situação onde o *Gerador de Casos* encontrar um roteamento do tipo OR, instâncias relativas para cada um das alternativas rotas possíveis serão geradas. Para roteamento do tipo AND, uma serialização das atividade é executada e instância que alternem a execução dessas atividades são geradas. Para os outros casos onde as atividades do modelo de processo são seqüencias, o mapeamento é feito diretamente.

Os casos são enviados (8) para o módulo *Escalonador Genético* (9) que, juntamente com as informações obtidas dos recursos associados para cada atividade e seus respectivos tempos de execução, analisam os modelos que melhor se adaptam à organização. Esta análise utiliza técnicas de escalonamento baseado também em AG. O *Escalonador Genético* simula a execução de todas as instâncias de processos geradas pelo módulo *Gerador de Casos*, atribuindo às atividades de cada instância de processo os recursos aptos para executá-las. Os modelos gerados (10) são também convertidos para formato XDPL, visando sua execução (11) em um motor de Workflow que contemple tal representação.

O Workflow Genético apresenta uma flexibilidade de representação possibilitando a aplicação da arquitetura no domínio de Web Service. Esta flexibilidade será apresentada através de estudo de casos no próximo capítulo. Na aplicação no cenário Web, o módulo *Repositório de Atividades* da arquitetura 4.4 exerce um papel de um registro de serviços UDDI. A publicação (12) da descrição dos serviços no repositório (1) é executada pelo respectivo *Provedor de Serviços* (14). A publicação no sistema pode ser feita manualmente ou por qualquer ferramenta de edição que contemple a representação adotada (XPDL). Por sua vez, o motor de Workflow realiza a conexão

e execução (13) com o executor (14) do respectivo serviço do modelo.

Os módulos da arquitetura, responsáveis pela geração dos modelos de processos, *Planejador Genético* e *Analisador*, que são o foco deste trabalho, serão detalhados nas seções seguintes. Maiores detalhes do módulo *Simulador*, *Gerador de Casos* e *Escalonador Genético* estão descritos em Alves et. al [5] e Guimarães et. al. [23].

4.3 Módulo Planejador Genético

O Algoritmo 2 implementa o planejador genético para a geração automática de modelos de processos, supondo um conjunto R contendo as atividades envolvidas no domínio do problema. Baseado em R, gera-se a população inicial P, composta de indivíduos que representam modelos de processos, coerentes e criados realizando variações de transições entre as atividades envolvidas. Além disso, R contém duas atividades genéricas *Begin* (A) e *End* (Z), que marcam o início e o fim de cada modelo de processo. Dessa maneira, todos os fluxos do processo iniciam na atividade genérica *Begin* e terminam na atividade genérica *End*.

Os indivíduos são analisados utilizando-se técnicas de planejamento. Uma análise nas transições das atividades do modelo é executada para averiguar se as pré-condições da atividade de origem são satisfeitas com as pós-condições da atividade destino. Na construção, todas as atividade cujas pré-condições da atividade de origem são satisfeitas com as pós-condições da atividade destino são consideradas. O resultado do algoritmo é um modelo preliminar M contendo uma representação de modelo de processos.

A evolução da população é realizada através da aplicação de operadores genéticos. Os operadores genéticos são aplicados mediante taxas de utilização ($\sharp$*Taxa_Cruzamento*, $\sharp$*Taxa_Mutação*). O algoritmo evolui a população inicial até um limite de $\sharp$*Número_Gerações* gerações.

4.3.1 Elementos do Planejador Genético

Na definição no Algoritmo 2, a representação do indivíduo, que descreve a estrutura das soluções, a população inicial, que representa os modelos de processos que serão evoluídos, os operadores genéticos, que determinam a evolução da população e a

função de adaptação, que avalia a qualidade da solução, são requeridos.

Algoritmo 2 Planejador Genético de Processos em Workflow

Entrada: R: conjunto de atividades do repositório

1: $P \leftarrow$ ***Geração_População_Inicial****(R)*
2: **para todo** indivíduo $Ind \in P$ **faça**
3: avalia *aptidão*(Ind)
4: **fim para**
5: **repeat**
6: **para** $k = 1$ **até** ($\sharp$*Taxa_Cruzamento* x $\sharp$*Tamanho_População*) **faça**
7: $pai_1 \leftarrow$ seleção *torneio(P)*
8: $pai_2 \leftarrow$ seleção *torneio(P)*
9: indivíduos filhos $\{filho_1, filho_2\} \leftarrow$ ***Cruzamento***(pai_1, pai_2)
10: avalia aptidão $\{filho_1, filho_2\}$
11: adiciona os indivíduos filhos $\{filho_1, filho_2\}$ na população intermediária PI
12: **fim para**
13: **para** $j = 1$ **até** $\sharp$*Taxa_Mutação* x $\sharp$*Tamanho_População*) **faça**
14: $Ind_m \leftarrow$ ***Mutação***(P, R)
15: avalia aptidão Ind_m
16: adiciona o indivíduo Ind_m na população intermediária PI
17: **fim para**
18: **Reproduz** ($\sharp$*Taxa_Elitismo* x $\sharp$*Tamanho_População*) melhores individuos $Ind_e \in P$
19: adiciona os indivíduos Ind_e na população intermediária PI
20: $P \leftarrow PI$
21: **until** $\sharp$*Número_Gerações*
22: **retorna** M: modelo preliminar de todos os processos válidos

4.3.2 Representação da solução

O uso de AG através da utilização de uma representação matricial dos cromossomos se torna bastante atrativa, uma vez que podemos tratar os principais tipos de roteamentos essenciais em workflow como disjunção e conjunção de rotas de fluxos, o que é inviável ou bastante onerosa se optar por uma outra forma de representação, como por exemplo, uma lista que apresenta um roteamento seqüencial ou uma árvore. A

representação em árvore, a qual PG trabalha, além de ser uma estrutura relativamente complexa, representa apenas roteamento de disjunção.

A representação do indivíduo Ind (uma possível solução) é uma matriz de tamanho fixo $n \times 3$, onde $n = |R|$. Cada linha k da matriz corresponde a um gen G_k do indivíduo representado por uma tripla (E_k, a_k, S_k), onde a_k representa uma atividade do processo, E_k representa o conjunto de atividades que precedem a atividade a_k e S_k representa o conjunto de atividades que sucedem a atividade a_k.

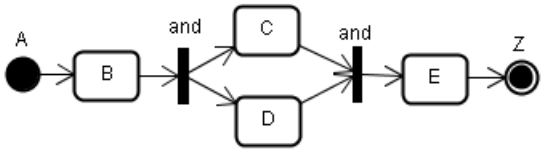
Tabela 4.2: Matriz Representação Indivíduo

Conjunto E_k	Atividade a_k	Conjunto S_k	Adaptação μ_k
E_1, ε_1	a_1	S_1, σ_1	μ_1
E_2, ε_2	a_2	S_2, σ_2	μ_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
E_n, σ_n	a_n	S_n, σ_n	μ_n

Se uma atividade $a_j \in E_k$ $(a_j \in S_k)$ então $a_j \notin S_k$ $(a_j \notin E_k)$ e $a_j \neq a_k$, onde $j = 1 \ldots n$. A tabela 4.2 mostra a matriz de representação de um indivíduo genérico. Em cada gen G_k, se as pós-condições de a_k satisfazem as pré-condições em distintas atividade de S_k, uma conjunção é verificada e um roteamento paralela (AND) ocorre. Porém, se as pós-condições satisfazem as pré-condições em distintas atividades de S_k, mas somente uma delas pode ser escolhida, então há uma escolha exclusiva (disjunção) e um roteamento seletivo (OR) ocorre. Em outras situações, se as pós-condições satisfazem somente uma atividade em S_k, um roteamento seqüencial (SEQ) ocorre. O conjunto E_k é avaliado de modo similar. Os roteamentos são representadas por ε_k em E_k e σ_k em S_k. Além disto, observa-se nesta tabela a coluna 4 referente a contribuição de cada gen G_k para a função de adaptação e armazenada em μ_k. Entretanto, esta coluna não compõe a representação do indivíduo, está incluído nesta tabela apenas para facilitar a representação da função de adaptação. Um exemplo de um indivíduo pode ser visto na Figura 4.5, assim como o respectivo modelo de processo.

E_{-k} Sets	Activity a_{-k}	S_{-k} Sets
	A	(B) seq
(A) seq	B	(C,D) and
(B) seq	C	(E) seq
(B) seq	D	(E) seq
(C,D) and	E	(Z) seq
(E) seq	Z	

(a) Representação Matricial



(b) Representação Gráfica

Figura 4.5: Exemplo representação indivíduo

4.3.3 Função de Adaptação

Conforme visto, cada individuo é uma representação de modelo de processo; entretanto, sob o ponto de vista de planejamento, cada indivíduo é um conjunto de planos. Assim, o processo de avaliação dos indivíduos leva em consideração aspectos de planejamento que neste caso são as regras de produção associadas a cada atividade. O processo de avaliação do individuo Ind consiste em avaliar cada gen G_k do indivíduo e verificando se as pré-condições de a_k são satisfeitas pelas pós-condições das atividades presentes em E_k. As pós-condições de a_k em relação as atividades em S_k são analisadas semelhantemente. Assim, pode-se observar a porcentagem de erros e acertos na satisfação das pré-condições e pós-condições da atividade a_k e armazenadas em μ_k. Uma atividade em E_k ou S_k é válida se satisfaz as pré-condições ou as pós-condições de a_k.

A função de adaptação do indivíduo Ind é dada pela equação 4.1, onde $n = |R|$ e também o tamanho de cada indivíduo Ind e $\mu_k = (\alpha(G_k)+\beta(G_k)+\gamma(G_k)+\varphi(G_k))/4$ é a média dos seguintes percentuais, sendo $\alpha(G_k)$ o percentual de atividades válidas em E_k, $\beta(G_k)$ o percentual de atividades válidas em S_k, $\gamma(G_k)$ o percentual de pré-condições satisfeitas da atividade a_k e $\varphi(G_k)$ o percentual de pós-condições satisfeitas da atividade a_k.

$$f(Ind) = \sum_{k=1}^{n} \mu_k \tag{4.1}$$

Um exemplo de avaliação do indivíduo pode ser visto na Figura 4.6, com as respectivas pré-condições e pós-condições de cada atividade a_k, além dos conjuntos E_k e

S_k. Considere a avaliação do gen G_2, cujo $a_2 = B$. Pode-se notar que, as atividades C e E do conjunto S_2 são válidas, porque as suas respectivas pré-condições b_1 e b_2 unificam com pós-condições da atividade B, assim como, as atividades G e F não são válidas em S_2 porque suas respectivas pré-condições não unificam com pós-condições da atividade B. A porcentagem de atividades válidas em S_2 é armazenada em $\beta(G_2)$, assim como, a porcentagem de pós-condições unificadas de B é armazenada em $\varphi(G_2)$. De modo similar é feito a análise das pré-condições de B em relação ao conjunto E_2 e atribuídas as porcentagens de unificações em $\alpha(G_2)$ e $\gamma(G_2)$. Por fim, é calculada a média das porcentagens do gen G_2 e armazenada em μ_2.

Conjunto E_k	Pré-Condição a_k	Atividade a_k	Pós-Condição a_k	Conjunto S_k
		A	a_1	(B)
(A) $\alpha(G_2) = 100\%$	a1 Unificável $\gamma(G_2) = 100\%$	B $\mu_2 = 79{,}25\%$	and(b_1 , ~~b_2~~ , b_3) Unificável $\varphi(G_2) = 67\%$	~~(C, G,~~ E, ~~F)~~ $\beta(G_2) = 50\%$
(B)	b_1	C	c_1	(D)
(F, C, G, E)	b_3	D	d_1	(Z)
(B)	b_2	E	e_1	(D)
(B)	g_1	F	f_1	(D)
(B)	and(c_1 , e_1)	G	g_1	
(D)	f_1	Z		

Figura 4.6: Exemplo Avaliação Indivíduo

Durante a evolução das populações, podem existir tanto nos conjuntos E_k quanto nos conjuntos S_k atividades não unificáveis. Sendo assim, caso $\gamma(G_k)$ ou $\varphi(G_k)$ tenham percentuais iguais a 100%, as atividades não unificáveis são eliminadas. O Algoritmo 3 então verifica se as pré-condições e/ou pós-condições das atividades de cada gen já foram totalmente unificadas, eliminado atividades desnecessárias nas entradas E_k ou saídas S_k do respectivo gen, assim como as respectivas referências.

4.3.4 Geração da População Inicial

A partir das atividades R do repositório cria-se a população inicial P que é constituída de indivíduos que representam possíveis modelos de processos que serão evoluídos no algoritmo do planejador genético. Estes modelos de processos são

Algoritmo 3 Verificador de Unificação

Entrada: *ind*: indivíduo

1: **para** $k = 0$ to $tamanho(ind)$ **faça**
2: **se** percentual $\varphi(G_k)$ de pós-condições unificadas da atividade a_k é igual a 100%, onde a_k é a atividade relativa do gen G_k **então**
3: **para** $j = 1$ **até** $|S_k|$ **faça**
4: **se** atividade b_j não é unificável, *onde* $b_j \in S_k$ **então**
5: $S_k = S_k - \{b_j\}$
6: **para** $i = 1$ **até** $tamanho(ind)$ **faça**
7: **se** $a_i = b_j$ /∗ onde a_i *é a atividade relativa do gen* G_i **então**
8: $E_i = E_i - \{a_k\}$
9: **fim se**
10: **fim para**
11: **fim se**
12: **fim para**
13: **fim se**
14: **se** percentual $\gamma(G_k)$ de pré-condições unificadas da atividade a_k é igual a 100%, onde a_k é a atividade relativa do gen G_k **então**
15: **para** $j = 1$ **até** $|E_k|$ **faça**
16: **se** atividade b_j não é unificável, *onde* $b_j \in E_k$ **então**
17: $E_k = E_k - \{b_j\}$
18: **para** $i = 1$ **até** $tamanho(ind)$ **faça**
19: **se** $a_i = b_j$ **então**
20: $S_i = S_i - \{a_k\}$
21: **fim se**
22: **fim para**
23: **fim se**
24: **fim para**
25: **fim se**
26: **fim para**
27: referencia saídas S_k vazias para o gen final G_F, onde $1 \leq k \leq tamanho(ind)$
28: referencia entradas E_k vazias para o gen inicial G_I, onde $1 \leq k \leq tamanho(ind)$
29: **retorna** *ind*: indivíduo verificado

coerentes, ou seja, todos os fluxos de cada modelo de processo iniciam na atividade genérica *Begin* (A) e terminam na atividade genérica *End* (Z). Os fluxos que compõem os modelos são aleatoriamente criados através de permutações aleatórias entre as atividades constituintes.

A geração dos indivíduos da população inicial é apresentada no Algoritmo 4. Inicialmente, cria-se o gen G_I e o gen G_F contendo as atividades genéricas *Begin* (A) e *End* (Z), respectivamente, correspondendo ao início e fim dos fluxos do processo e também os gens G_1 e G_n de cada indivíduo. Em seguida, são selecionados aleatoriamente atividades em L (Lista auxiliar, que contem as atividade de R) para compor o conjunto S_I de G_I, excluídas as atividades A e Z de L.

O próximo passo, consiste em verificar S_k de G_k do individuo Ind a partir de G_I, onde $1 \leq k \leq tamanho(Ind)$. Se S_k é vazio, então é realizado uma nova seleção de atividades em L para compor S_k; eliminando as atividades selecionadas de L. Para cada atividade $b_j \in S_k$, onde $1 \leq j \leq tamanho(S_k)$, se existe algum gen i, $1 \leq i \leq tamanho(Ind)$, tal que $a_i = b_j$, então o conjunto de entrada E_i de G_i torna-se $E_i \cup \{a_k\}$. Caso contrário, então não existe G_i tal que $a_i = b_j$, sendo assim, deve-se criar G_{k+1}, onde $E_{k+1} = E_{k+1} \cup \{a_k\}$ e $a_{k+1} = b_j$. O conjunto S_{k+1} é construído de forma similar. Note que os conjuntos de entrada E_k, $1 \leq k \leq tamanho(Ind)$, são criados em função das saídas S_k. Por último, referencia-se os S_k nulos para G_F.

Um exemplo da geração de um indivíduo pode ser visto na Figura 4.7. No passo 1 tem-se a criação dos gens G_1 e G_n através da representação das atividades genéricas A e Z, respectivamente. A lista auxiliar L contém as atividades *B, C, D, E, F* do repositório R, excluído as atividades genéricas A e Z. Em seguida, é realizada no passo 2 a seleção aleatória de atividades em L para compor S_1; considere a seleção das atividades *E, B, C*. As atividades selecionadas são eliminadas da lista L.

No passo 3 é verificado a atividade E na saída S_1 do gen G_1, como não existe um gen G_i, $1 \leq i \leq tamanho(Ind)$, tal que $a_i = E$, então cria-se o novo gen G_2, onde $E_2 = A$ e $a_2 = E$. Em seguida, é selecionado aleatoriamente atividades em L para compor S_2; considere a seleção da atividade F. Nos passos 4 e 5 são realizados as verificações das outras atividades B e C de S_1 de forma similar. Após a verificação das atividades de S_1, realiza-se a remoção das atividades selecionadas aleatórias em L.

Algoritmo 4 Geração_População_Inicial

Entrada: R: conjunto de atividades do repositório

1: **para** $l = 0$ to $\sharp$*Tamanho da População* **faça**
2: lista auxiliar de atividades $L \leftarrow R$
3: criar gen inicial G_I e adicionar em a_I a atividade genérica inicial A
4: criar gen final G_F e adicionar em a_F atividade genérica final Z
5: remover atividades A e Z da lista L
6: seleção aleatória atividades em L para compor S_I de G_I
7: adicionar G_I no individuo Ind
8: **para** $k = 1$ **até** $tamanho(Ind)$ **faça**
9: **se** $S_k = \emptyset$ **então**
10: seleção aleatória atividades em L para compor S_k de G_k
11: remove atividade selecionadas de L
12: **fim se**
13: **para** $j = 1$ **até** $tamanho(S_k)$ **faça**
14: **se** $\exists$ algum gen i, $1 \leq i \leq tamanho(Ind)$, tal que a_i é igual atividade b_j de S_k **então**
15: $E_i = E_i \cup \{a_k\}$ /* *onde* a_k *é a atividade relativa do gen* G_k
16: **senão**
17: criar um novo gen G_{k+1}
18: $a_{k+1} = b_j$
19: $E_{k+1} = E_{k+1} \cup \{a_k\}$
20: seleção aleatória atividades em L para compor S_{k+1} de G_{k+1}
21: adicionar G_{k+1} em Ind
22: **fim se**
23: **fim para**
24: remover atividades selecionadas da lista L
25: **fim para**
26: referencia saídas S_k vazias para o gen final G_F, onde $1 \leq k \leq tamanho(Ind)$
27: referencia entradas E_k vazias para o gen inicial G_I, onde $1 \leq k \leq tamanho(Ind)$
28: $P \leftarrow P \cup Ind$
29: **fim para**
30: **retorna** P : população inicial

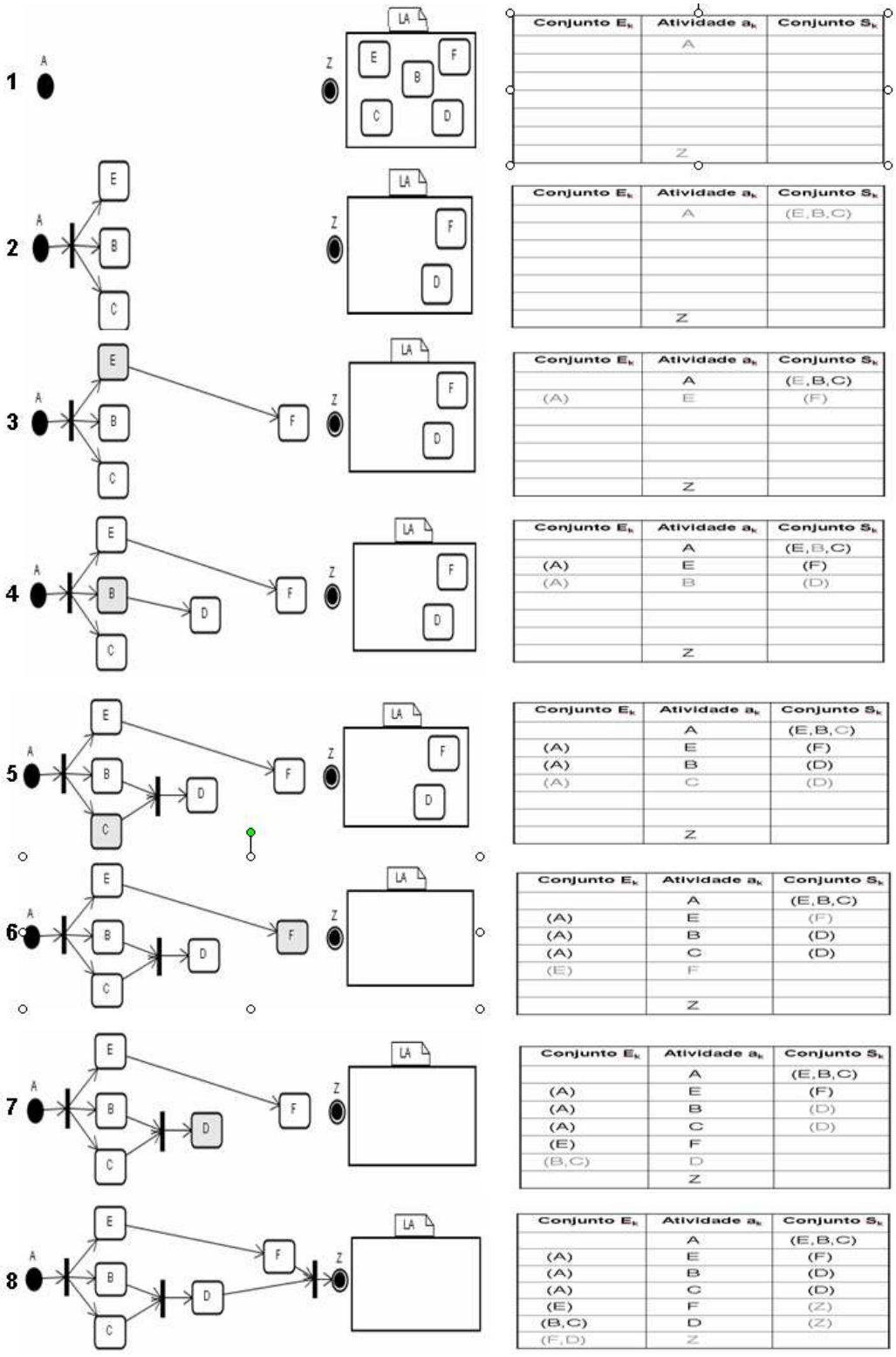


Figura 4.7: Geração de um Indivíduo

De modo similar a análise da saída S_1, os passos 6 e 7 verificam as atividades das demais saídas S_k, $1 \leq k \leq tamanho(Ind)$. No passo 6 é realizado a análise da atividade F em S_2, criando o gen G_5, onde $E_5 = E$ e $a_2 = F$. No passo 7, é realizado a análise das saídas dos gens G_3 e G_4, respectivamente. Note-se que, a lista L está vazia neste passo, logo não são geradas saídas para os respectivos gens. No passo 8, referencia-se as saídas S_k vazias para o gen final G_n.

4.3.5 Operadores Genéticos

A evolução da população é realizada através da aplicação de operadores genéticos. O Algoritmo 2 utiliza os operadores de Cruzamento, Mutação e Reprodução.

Na Reprodução um percentual ($\sharp$*Taxa_Elitismo*) dos melhores indivíduos em P são copiados, da população corrente para a próxima. Os operadores de Cruzamento e Mutação são detalhados a seguir.

◦ Operador de Cruzamento

O operador de Cruzamento realiza a permutação entre atividades dos conjuntos E_k e/ou S_k de gens de indivíduos pais previamente selecionados, gerando novos indivíduos filhos para a próxima geração da população. O processo de seleção de indivíduos para cruzamento é realizado através de torneio, que consiste em selecionar o indivíduo Ind com maior valor de função de adaptação entre três indivíduos aleatoriamente selecionados.

Assim, após a seleção dos indivíduos pais (*pai1* e *pai2*), o ponto de cruzamento será aleatoriamente selecionado entre os gens G_k, $1 \leq k \leq tamanho(Ind)$ de cada indivíduo. Os indivíduos filhos (*filho1* e *filho2*) resultam da permutação de atividades entre os conjuntos E_k e S_k dos gens selecionados. O operador de cruzamento nos conjuntos de entrada E_k é apresentado no Algoritmo 5, o cruzamento na saída S_k é realizado de modo similar, apenas substituindo E_k por S_k.

Considere por exemplo os indivíduos, *pai*1 e *pai*2, da Figura 4.8. Considere também os gens da atividade D selecionados. Aplicando o operador de cruzamento, os conjuntos de entrada E_k e saída S_k dos gens selecionados serão permutados. Primeiramente, as atividades de referências dos conjuntos E_k e S_k são eliminadas, como podemos ver no passo 2. Em seguida, no passo 3 ocorre a permutação entre

as atividades dos conjuntos E_k e S_k, assim como as respectivas referências.

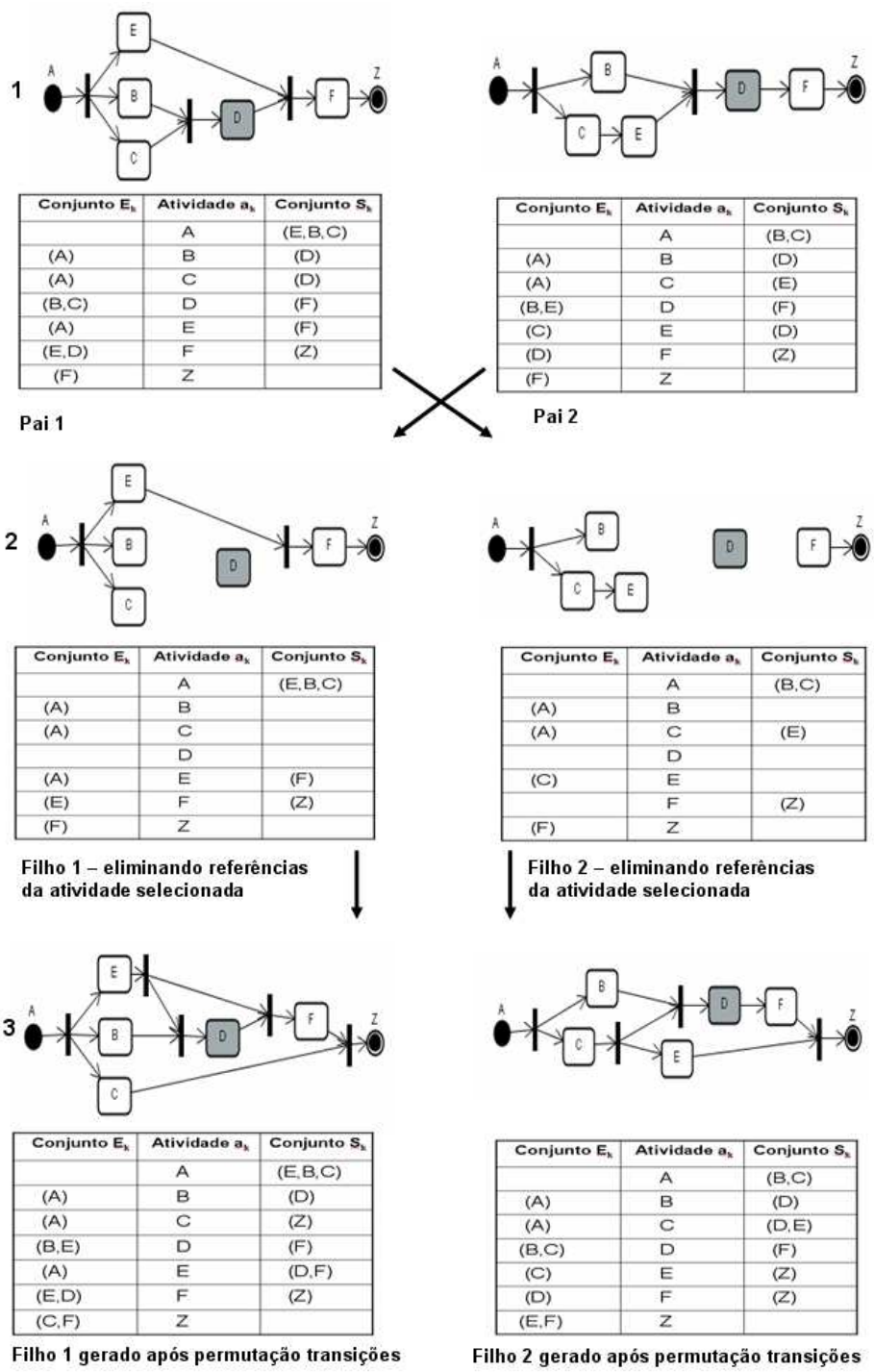


Figura 4.8: Operação de cruzamento entre dois indivíduos pais

Algoritmo 5 Cruzamento Entrada

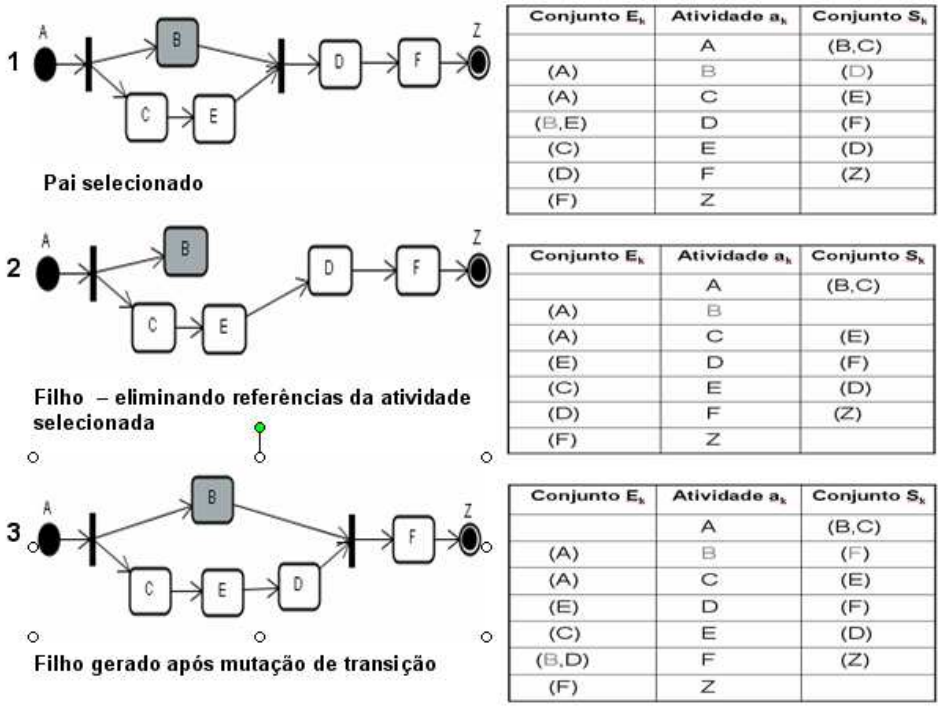
Entrada: *pai1,pai2*: Pais

1: $G_{pai1} \leftarrow$ seleciona aleatoriamente um gen de troca no pai_1
2: $G_{pai2} \leftarrow$ seleciona aleatoriamente um gen de troca no pai_2
3: $Eaux_{pai1} \leftarrow E_{pai1}$ /* *conjunto auxiliar de entradas do gen de troca no* pai_1
4: $Eaux_{pai2} \leftarrow E_{pai2}$ /* *conjunto auxiliar de entradas do gen de troca no* pai_2
5: **para** $j = 1$ **até** $|E_{pai1}|$ **faça**
6: **para** $i = 1$ **até** $tamanho(pai_1)$ **faça**
7: **se** $a_i = b_j$ **então** $S_i = S_i - \{a_{pai1}\}$
8: /* *onde* $b_j \in E_{pai1}$, a_{pai1} é a atividade relativa do gen G_{pai1} e a_i *é a atividade relativa do gen i*
9: **fim para**
10: **fim para**
11: **para** $j = 1$ **até** $|E_{pai2}|$ **faça**
12: **para** $i = 1$ **até** $tamanho(pai_2)$ **faça**
13: **se** $a_i = b_j$ **então** $S_i = S_i - \{a_{pai2}\}$
14: **fim para**
15: **fim para**
16: Remove E_{pai1} e E_{pai2}
17: $E_{pai1} \leftarrow Eaux_{pai2}$ e $E_{pai2} \leftarrow Eaux_{pai1}$
18: **para** $j = 1$ **até** $|E_{pai1}|$ **faça**
19: **para** $i = 1$ **até** $tamanho(pai_1)$ **faça**
20: **se** $a_i = b_j$ **então** $S_i = S_i \cup \{a_{pai1}\}$
21: **fim para**
22: **fim para**
23: **para** $j = 1$ **até** $|E_{pai2}|$ **faça**
24: **para** $i = 1$ **até** $tamanho(pai_2)$ **faça**
25: **se** $a_i = b_j$ **então** $S_i = S_i \cup \{a_{pai2}\}$
26: **fim para**
27: **fim para**
28: referencia saídas S_i vazias para o gen final G_F, onde $1 \leq i \leq tamanho(pai1)$
29: referencia saídas S_j vazias para o gen final G_F, onde $1 \leq j \leq tamanho(pai2)$
30: referencia entradas E_i vazias para o gen inicial G_I, onde $1 \leq i \leq tamanho(pai1)$
31: referencia entradas E_j vazias para o gen inicial G_I, onde $1 \leq j \leq tamanho(pai2)$
32: **retorna** $filho1 \leftarrow pai1$ e $filho2 \leftarrow pai2$

◦ Operador de Mutação

O operador de Mutação substitui atividades não unificáveis em E_k ou S_k, por novas atividades não pertencentes a estes conjuntos e também diferente de a_k, gerando um novo indivíduo filho para a próxima geração da população. Para tanto, seleciona-se aleatoriamente um indivíduo Ind em P. Um gen G_k, $1 \leq k \leq n$, é selecionado de acordo com a sua contribuição para o valor da função de adaptação, aquele com menor μ_k apresenta maior probabilidade de seleção no método da roleta.

O operador de mutação no conjunto de saída S_k é apresentado no Algoritmo 6, a mutação na entrada E_k é realizado de modo similar, apenas substituindo S_k por E_k. Para exemplificar, considere o indivíduo *pai* da Figura 4.9. Considere que o gen G_2 seja selecionado. Aplicando o operador de mutação no conjunto de saída S_2, a atividade D é eliminada, considerando que suas pré-condições não unificam com as pós-condições da atividade $a_2 = B$. Temos também que, B deve ser eliminado no conjunto E_4, pois corresponde a atividade que precede D. Em seguida, no passo 3 uma nova atividade é selecionada aleatoriamente no repositório R para o conjunto S_2. Supondo que essa atividade seja F, então ela é incluída em S_2. Note que, F corresponde a atividade a_6 do gen G_6, sendo assim, B deve ser incluída em E_6.



Conjunto E_k	Atividade a_k	Conjunto S_k
	A	(B,C)
(A)	B	(D)
(A)	C	(E)
(B,E)	D	(F)
(C)	E	(D)
(D)	F	(Z)
(F)	Z	

Conjunto E_k	Atividade a_k	Conjunto S_k
	A	(B,C)
(A)	B	
(A)	C	(E)
(E)	D	(F)
(C)	E	(D)
(D)	F	(Z)
(F)	Z	

Conjunto E_k	Atividade a_k	Conjunto S_k
	A	(B,C)
(A)	B	(F)
(A)	C	(E)
(E)	D	(F)
(C)	E	(D)
(B,D)	F	(Z)
(F)	Z	

Figura 4.9: Operação de Mutação Saída

Algoritmo 6 Mutação Saída

Entrada: *P,R*: População, atividades do repositório

1: *pai* ← roleta(P) /∗ *indivíduos com menor valor de aptidão apresentam um percentual maior na roleta*
2: G_{pai} ← seleciona aleatoriamente um gen de troca no *pai*
3: **para** $j = 1$ **até** $|S_{pai}|$ **faça**
4: **se** atividade b_j não é unificável, *onde* $b_j \in S_{pai}$ /∗ *ou seja, as pré-condições da atividade de saída* b_j *não satisfazem as pós-condições da atividade* a_{pai} *do gen de troca* **então**
5: $S_{pai} = S_{pai} - \{b_j\}$
6: **para** $i = 1$ **até** $tamanho(pai)$ **faça**
7: **se** $a_i = b_j$ /∗ onde a_i *é a atividade relativa do gen* i **então**
8: $E_i = E_i - \{a_{pai}\}$
9: **fim se**
10: **fim para**
11: seleciona nova atividade aleatória a_r em R obedecendo restrições de inserção
12: **se** restrições de inserção satisfeitas **então**
13: $S_{pai} = S_{pai} \cup a_r$
14: **para** $k = 1$ **até** $tamanho(pai)$ **faça**
15: **se** $a_k = a_r$ **então**
16: $E_k = E_k \cup \{a_{pai}\}$
17: **fim se**
18: **fim para**
19: **fim se**
20: **fim se**
21: **fim para**
22: referencia saídas S_k vazias para o gen final G_F, onde $1 \leq k \leq tamanho(pai)$
23: referencia entradas E_k vazias para o gen inicial G_I, onde $1 \leq k \leq tamanho(pai)$
24: **retorna** $filho \leftarrow pai$

4.4 Módulo Analisador

O resultado do Planejador Genético referente ao Algoritmo 2 é um modelo preliminar M contendo uma representação de processos válidos. O módulo Analisador, implementado no Algoritmo 7 simplesmente verifica em M a possibilidade de existir mais de um modelo de processos, ou seja, verifica atividades que executam o mesmo papel.

Algoritmo 7 Módulo Analisador

Entrada: *ind*: indivíduo representando o modelo M

1: **para** $k = 1$ to $tamanho(ind)$ **faça**
2: verifica a unificação das pré-condições das atividades na saída S_k em relação as pós-condições da atividade a_k do gen G_k
3: **se** distintas atividades em S_k unificam a mesma condição em a_k **então**
4: seta as referidas atividades em S_k como ponto de conflito, assim como suas respectivas referências. Cada ponto de conflito corresponde a um possível modelo de processo, representado fluxos alternativos.
5: **fim se**
6: **fim para**
7: $\{F\} \leftarrow$ percorre o individuo extraindo modelos nos pontos de conflito.
8: **retorna** $\{F\}$: conjunto de modelos finais

Para exemplificar, considere por exemplo, as atividade da tabela 4.3 como as atividades R do repositório. Com a aplicação do Planejador Genético, a representação do modelo M descrito na Figura 4.10 é obtida. Após o retorno do modelo preliminar M pelo módulo Planejador Genético o módulo Analisador da arquitetura da Figura 4.4 percorre a respectiva representação verificando e analisando os pontos de conflito e extraindo modelos finais de processos válidos.

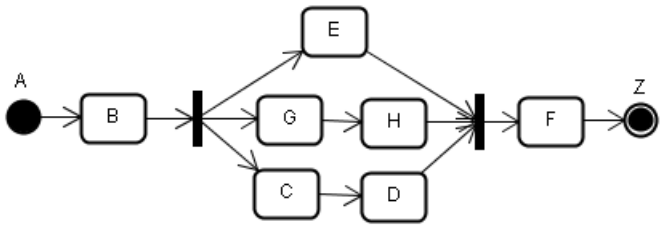
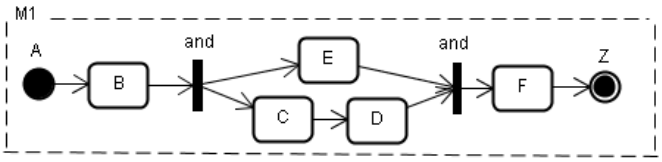


Figura 4.10: Modelo preliminar gerado pelo Planejador Genético

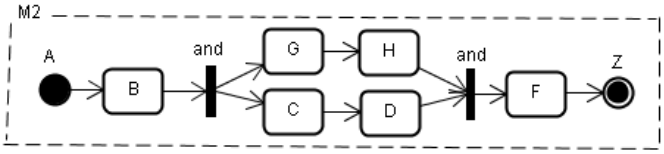
Tabela 4.3: Especificação Atividades do Repositório

Atividades	**Pre_Condições**	**Pos_Condições**
(A)		a_1
(B)	a_1	b_1 e b_2
(C)	b_1	c_1
(D)	c_1	d_1
(E)	b_2	x_1
(G)	b_2	g_1
(H)	g_1	x_1
(F)	d_1 e x_1	f_1
(Z)	f_1	

Baseado no modelo M da Figura 4.10, pode-se notar que as pré-condições da atividade E e G são similares, assim como as pós-condições da E e H também são similares. Logo, o Analisador constata que os blocos de atividades G e H executam o mesmo papel que a atividade E. Portanto, tem-se mais de uma representação de modelo de processo. Após a aplicação do Analisador, os modelos de processos $M1$ e $M2$ presentes na Figura 4.11 são extraídos, considerando também a representação dos roteamentos OR-Split, Or-Join, AND-Split e AND-Join.



(a) M1



(b) M2

Figura 4.11: Modelos finais extraidos pelo Analisador

4.5 Resultados e Discussões

A arquitetura proposta alcançou o objetivo principal de extrair automaticamente modelos de processos baseado nas atividades disponíveis, considerando roteamentos paralelos, condicionais e seqüenciais, tornando-se uma ferramenta adequada de auxílio na modelagem de processos em Sistemas de Workflow. Quando novas atividades estão disponíveis, basta que sejam publicadas no repositório para que o sistema encontre e gere novos modelos de processos.

O tempo desta geração também foi considerado satisfatório, e como não foi encontrado durante a pesquisa benchmark de comparação, escolheu-se o planejador condicional POND [11] (versão 1.1), com o intuito de realizar comparação com um planejador *estado-da-arte* desta área. O objetivo é avaliar a eficiência da extração dos modelos; no caso específico medindo somente fluxos com ações seqüencias e diferentes ações condicionais.

4.5.1 Planejador Condicional POND versus Workflow Genético

No Planejador POND, a declaração *observe(X)* especifica que o efeito não é determinístico, como pode ser visto na Figura 4.12, onde uma ação B que pode produzir dois efeitos diferentes (representados por *in b1* e *in b2*) de forma não-determinística. As ações C e D são satisfeitas pelos efeitos da ação B respectivamente.

Normalmente em domínio de Planejador condicional contempla-se apenas a ramificação do tipo OR-Split, porém, para manter o padrão com modelos de processos em workflow, foi considerado no primeiro experimento roteamentos do tipo OR-Split juntamente com seus roteamentos de convergência OR-Join. O tipo OR-Join é essencial em processos de workflow, pois representam o ponto de convergência dos fluxos condicionais.

Um exemplo de ações com roteamentos OR-Split e OR-Join para o planejador POND pode ser visto na representação PDDL na Figura 4.14, com a definição do problema descrita na Figura 4.13. A ação B produz dois efeitos diferentes OR-Split, representados pela satisfação dos predicados *(inB negativo)* e *(inB positivo)* . As ações C e D são satisfeitas pelos efeitos da ação B, produzindo os efeitos *c1* e *d1*, respectivamente. Para a execução da atividade E é necessário que os predicados *c1* ou *d1* sejam satisfeitos. Note-se que apenas um dos predicados *c1* ou *d1* precisam

ser satisfeitos, tendo uma convergência do tipo OR-Join.

```
(:action B
  :precondition (a1)
  :effect (and (observe (in b1))
    (observe (in b2))))
(:action C
  :parameters ()
  :precondition (in b1)
  :effect (c1))
(:action D
  :parameters ()
  :precondition (in b2)
  :effect (d1))
```

Figura 4.12: Representação de ação condicional no planejador POND

```
(define (problem ReclamacaoTelefonica) (:domain condicional)
  (:requirements :strips :equality :typing :conditional-effects
  :disjunctive-preconditions) (:objects positivo negativo - status)
    (:init
      (begin)
      (oneof (inB positivo) (inB negativo) ) ) )
  (:goal (end)) )
```

Figura 4.13: Definição Problema no planejador POND

Os experimentos a seguir foram realizados em um micro computador Pentium 4 3.0 GHz e com memória RAM de 992Mb e Sistema Operacional Linux. As probabilidades dos operadores genéticos são dadas por $\sharp Taxa\_Cruzamento = 0.70$, $\sharp Taxa\_Mutação = 0.20$, $\sharp Taxa\_Elitismo = 0.10$ e o tamanho da população é de 100 indivíduos. O Algoritmo evolui até encontrar um indivíduo com porcentagem de adaptação igual a 100%.

```
(define (domain condicional) (:requirements :strips :equality :typing :conditional-effects
:negative-preconditions :disjunctive-preconditions)
(:types status) (:constants )
(:predicates (inB ?s - status) (begin) (B1) (C1) (end) (positivo) (negativo) )
(:action B
  :parameters (?s - status)
  :precondition (begin)
  :effect (and (observe (inB ?s))
    (observe (not (inB ?s)))))
(:action C
  :parameters ()
  :precondition (inB negativo)
  :effect (c1))
(:action D
  :parameters ()
  :precondition (inB positivo)
  :effect (d1))
(:action E
  :parameters (?s - status)
  :precondition (or (c1) (d1))
  :effect (end)))
```

Figura 4.14: Representação PDDL de ação condicional no planejador POND

No primeiro experimento, foram feitos testes com diferentes quantidades de atividades que representam roteamentos do tipo OR-Split, juntamente com atividades que representam convergência do tipo OR-Join. Duas possibilidades de roteamento para cada atividade OR-Split foram consideradas, exemplificando o caso mais comum em modelos de workflow, em que um determinado ponto do processo, o fluxo segue uma determinada rota se alguma condição for positivo ou segue outra rota, em caso contrário. A Figura 4.15 mostra um exemplo de modelo de processo com roteamentos OR-Split e OR-Join, realizado nos testes de comparação.

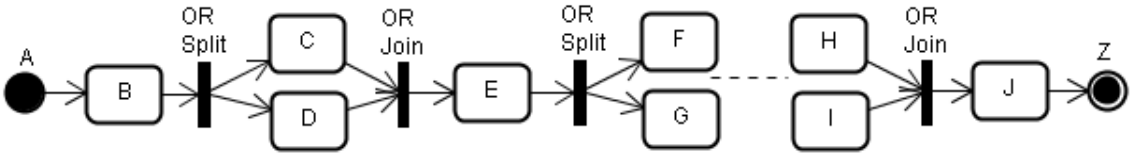


Figura 4.15: Modelo de processos com roteamento OR-Split e OR-Join

A Figura 4.16 mostra o tempo de execução versus quantidade de atividades OR-Split e OR-Join na execução do planejador POND. A Figura 4.17 mostra o tempo de execução versus quantidade de atividades OR-Split e OR-Join na execução do Workflow Genético. O tempo de execução do Workflow Genético é uma média entre duas execuções, uma vez que este tempo depende do comportamento da evolução, que é influenciado, por exemplo, pela diversidade da população inicial criada aleatoriamente. A Figura 4.18 mostra o tempo entre duas execuções do Workflow Genético.

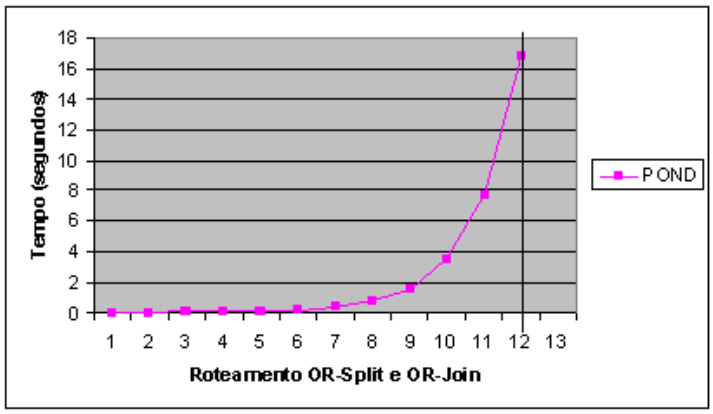


Figura 4.16: Tempo para atividades OR-Split e OR-Join - POND

Apesar do planejador POND ter um desempenho rápido em relação ao Workflow Genético até o limite de 12 atividades do tipo OR-Split juntamente com 12 atividades do tipo OR-Join, como pode ser visto através da Figura 4.19, tem-se que, após este limite o planejador POND não consegue tratar todos os processos recursivos em execução, conforme pode ser observado na mensagem de erro da Figura A.1 da seção A.2 do apêndice A. Como os domínios de problemas em workflow do mundo real podem envolver situações similares ao exemplo, o Workflow Genético se torna mais viável. O Workflow Genético superou o limite de situações que o POND conseguiu tratar. Alguns tempos adicionais de testes do Workflow Genético para esse experimento podem ser vistos na Tabela 4.4.

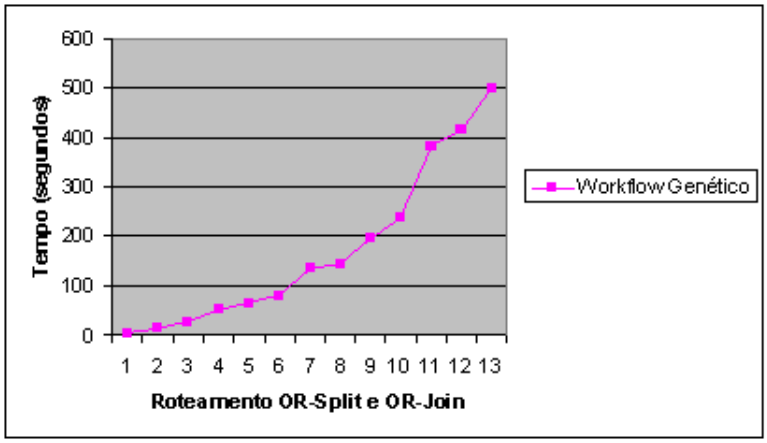


Figura 4.17: Tempo para atividades OR-Split e OR-Join - Workflow Genético

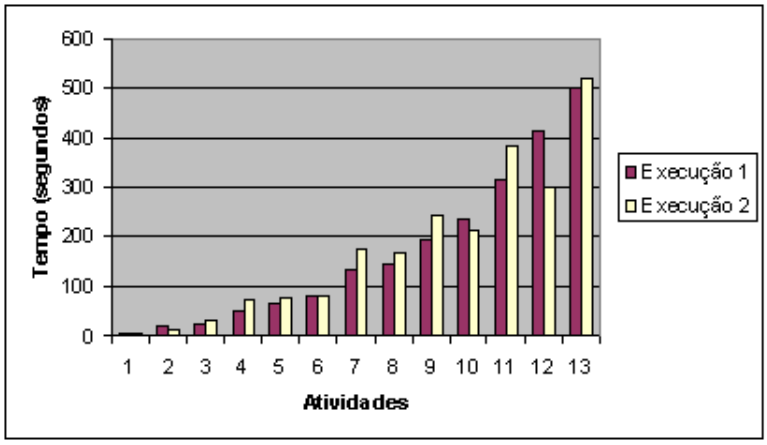


Figura 4.18: Tempo de execução Workflow Genético

Além disso, o planejador POND não se mostrou eficiente em problemas onde um dos ramos conduzia à meta e outro não, entrando em um laço infinito e não produzindo nenhum resultado. Para a modelagem de workflow isto significa que apenas os caminhos gerados são significativos. Entretanto, o modelo pode representar estes ramos como indefinidos, permitindo que o desenhista de processos reveja seu levantamento de atividades. No Workflow Genético, esta situação não foi identificada como um problema, uma vez que soluções ótimas ou não são retornadas.

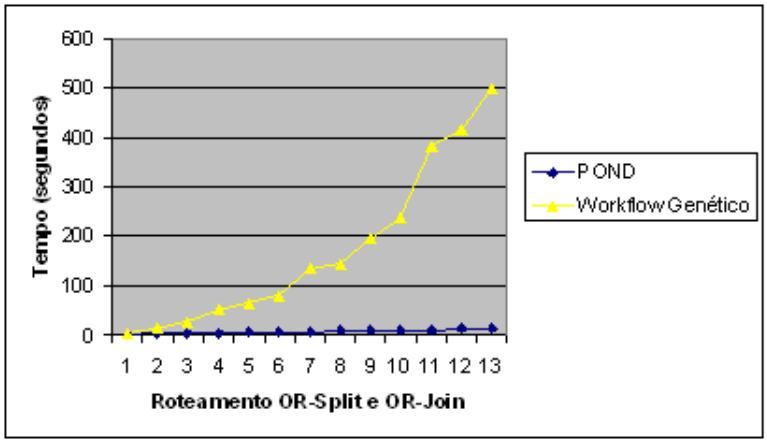


Figura 4.19: Comparação de atividades OR-Split e OR-Join - POND x Workflow Genético

Tabela 4.4: Execuções Workflow Genético

Atividades OR-Split e OR-Join	**Total geral atividades**	**Tempo médio (segundos)**
12	36	500
13	39	700
15	45	2400
15	50	5400

Um segundo experimento foi realizado considerando apenas quantidade de roteamentos do tipo OR-Split. Neste experimento foram feitos testes com diferentes quantidades de roteamentos OR-Split para uma mesma atividade. Este experimento contempla em domínio de workflow a possibilidade de seguir por várias rotas a partir de um determinado ponto do processo. A Figura 4.20 mostra um exemplo de modelo de processo com múltiplos roteamentos OR-Split realizados nos testes deste experimento, onde n é o número de atividades no roteamento OR-Split. O planejador POND abortou (finalizou) a execução e não retornou resultados quando atribuído um limite de $n = 20$. No Workflow Genético, a execução ocorreu em aproximadamente 2400 segundos, retornando a resposta esperada.

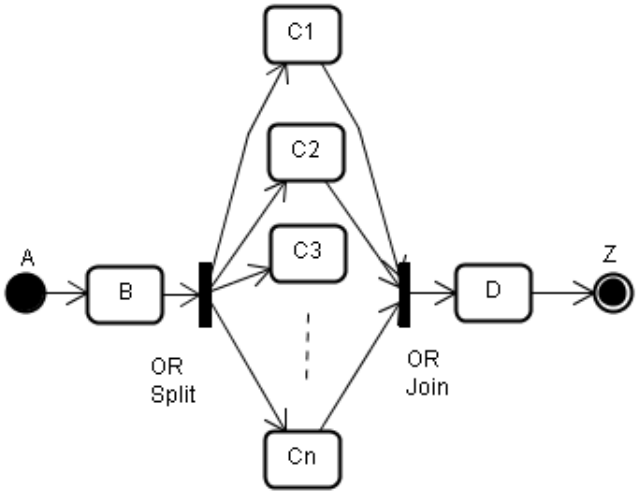


Figura 4.20: Modelo de Processo com roteamento OR-Spli

A definição de um modelo particular pode envolver eventualmente a execução de atividades em paralelo. Um plano possível, produzido pelo planejador POND seria P=B,C,D,F, entretanto, não é possível identificar as atividades que estão em paralelo e a ordem de execução. Estes aspectos são tratados pelo Workflow Genético, como pode ser visto na Figura 4.21. Além das características mencionadas, para utilizar o resultado do POND em um motor de workflow necessitaria de um conversor para a linguagem que o motor utiliza. Como o Workflow Genético trabalha com a evolução e conversão de processos para XPDL, a integração com o motor de workflow é automática.

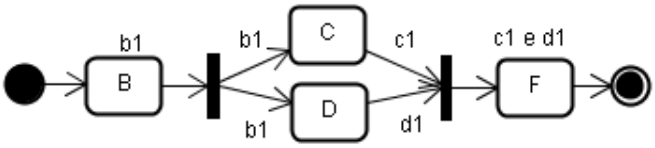


Figura 4.21: Modelo com atividades em paralelo

Do ponto de vista de trabalhos relacionados, realizou-se uma comparação com o planejador PRODIGY, o qual foi utilizado na ferramenta SHAMASH em Aler et. al. [1]. A arquitetura Workflow Genético é inspirada neste trabalho, entretanto não foi possível compará-las, pois a ferramenta SHAMASH não foi encontrada. Prodigy em trabalho anterior realiza somente busca sequencial. Em relação ao tempo de

processamento de atividades seqüenciais o planejador Prodigy tem um desempenho melhor que o Workflow Genético. O diferencial apresentado pelo Workflow Genético é a capacidade de considerar os roteamentos OR-Split, OR-Join, AND-Split e AND-Join, que são essenciais para o tratamento de problemas de workflow no mundo real.

Esta seção mostrou a viabilidade da aplicação da arquitetura do Workflow Genético na modelagem de processos de Workflow. Outra característica a ser considerada é a flexibilidade de representação do Workflow Genético possibilitando a aplicação em diferentes domínios do mundo real. No próximo capítulo será apresentado, através de estudo de casos, a possibilidade de aplicação da arquitetura no domínio Web.

Capítulo 5

Aplicação em Workflow de Web Services - Estudo de Casos

A arquitetura proposta no capítulo anterior apresenta uma grande flexibilidade de representação, podendo ser utilizada em diferentes domínios de conhecimento, como por exemplo o domínio Web, através da utilização de serviços disponíveis em uma rede de computadores (Web Services). A experimentação neste domínio demonstra esta flexibilidade, para isto, é apresentado neste capítulo estudo de casos de desenvolvimento de modelos de processo de negócio formado por uma composição de serviços que estejam disponíveis em uma rede de computadores.

5.1 Modelagem de processo em Workflow de Web Services

A capacidade de negócios pode ser definida como um conjunto de serviços que trabalham de maneira coerente em busca de um propósito de negócio definido. Tal conceito ajuda na mudança de uma visão funcional de uma empresa fechada, tradicional, para uma visão orientada a serviço de uma empresa aberta, capacitada para negócios eletrônicos. O que permite que haja uma evolução de aplicações projetadas de forma fortemente acopladas, inflexíveis, para aplicações fracamente acopladas, com conjunto flexível de interfaces que facilita o compartilhamento de capacidades de negócios [3].

A tecnologia de workflow é utilizada para coordenar as iterações entre os Web Services, os quais de uma forma composta visam alcançar um objetivo comum dentro

da organização. E os Web Services representam os passos lógicos que compõem o Workflow, ou seja, os passos que devem ser executados para que se alcance um objetivo de negócio dentro da organização.

Conforme Aversano et al. [7], uma grande necessidade de organizações virtuais é o suporte automático a processos de negócios que ultrapassam as fronteiras de uma empresa. As necessidades da globalização emergente e o impacto propulsivo da Internet estão mudando as estratégias das empresas. Modelos de processos implementados devem garantir um alto grau de flexibilidade e liberdade para os atores envolvidos nas atividades do processo.

Conforme afirma Wohed et al. [62], a composição de Web Services é um paradigma emergente que possibilita a integração tanto de aplicações internas, como de aplicações que transpõem as fronteiras organizacionais. Desse modo, linguagens e técnicas para composição de Web Service têm surgido e estão sendo continuamente aprimoradas, porém um mecanismo de composição automática de serviços em uma rede de computadores é ainda um assunto não encontrado na literatura durante a pesquisa. Sendo que, a arquitetura proposta no capítulo anterior pode ser facilmente adaptada para auxiliar nesta composição.

Os Web Services, nesta pesquisa, representam as atividades do Workflow, ou seja, são os passos a serem seguidos para que o objetivo do processo seja alcançado. Tais serviços devem estar disponíveis numa rede de computadores, que pode ser uma rede interna de uma organização, ou mesmo a própria Internet. Isto é definido pela organização levando em consideração, por exemplo, o quanto deseja expor processos de negócio através de Web Services.

Existem na literatura várias linguagens de representação de modelagem de fluxos de processos, inclusive aplicados à composição de Web Services. No entanto, a arquitetura proposta utiliza a linguagem de definição de processos XPDL que além de contemplar as principais características de workflow fornece através do campo atributos estendidos $< ExtendedAttribute >$ uma flexibilidade de representação. No campo $< ExtendedAttribute >$ podemos representar além das informações das regras de produção das atividades como descritas no capítulo 4, informações complementares para aplicar no domínio de Web Service. Um exemplo da descrição de uma atividade com a respectiva regra de produção e informações complementares

podem ser vistas na Figura 5.1 e esboçados na Figura 5.2.

< *ActivityId* = "*WebServiceColetar*" *Name* = "**WS Coletar**»
  < *Description* >
    Coletar informações reclamação telefônica < /*Description* >
  < *Performer* > *System* < /*Performer* >
  < *ExtendedAttributes* >
    < *ExtendedAttribute* >
      < ***Regra Produção*** >
        < ***Pre-Condições and(Depto-Contactado)(Cliente-Contactado)*** >
        < ***Pos-Condições=(Coleta-ok)*** >
      < ***Url=(http://localhost/Mestrado/WebService/Coletar.asmx")*** >
      < ***Porta=(8080)*** >
      < ***Categoria=Reclamação-Telefônica*** >
      < ***Tempo-Estimado=10*** >
      ⋮
    < /*ExtendedAttribute* >
  < /*ExtendedAttributes* >
< /*Activity* >

Figura 5.1: Representação Web Service através de Atividade XPDL

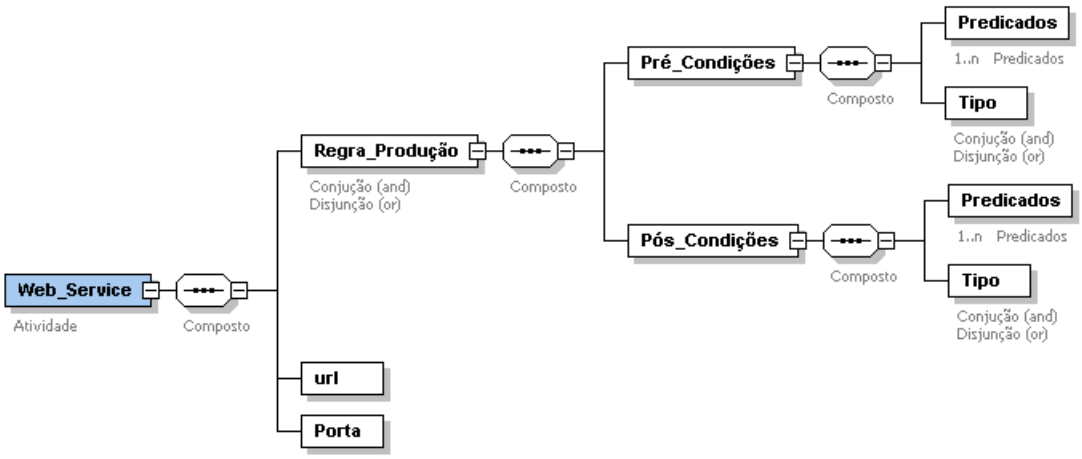


Figura 5.2: Esboço estrutura Web Service

As informações complementares das atividades podem ser obtidas da descrição WSDL de cada Web Service ou de suas descrições no registro UDDI. Informações

para localização e execução dos serviços, como URL, nome serviço, porta devem estar disponíveis, além de informações auxiliares como categoria do problema, e tempo estimado de execução. A modelagem de processos é executada por meio das informações das regras de produção (pré-condições e pós-condições) dos Web Services publicados no registro UDDI. Informações relacionadas a categoria é usada para agrupar os problemas específicos. Informações relacionadas ao tempo estimado são usadas pelo Escalonador Genético, que pode ser visto em Alves et. al [5] e Guimarães et. al. [23]. As outras informações tal como URL e porta são úteis para localização e execução dos Web Services.

Existem ambientes de testes UDDI na Web no qual registros podem ser publicados, como por exemplo, projetos da IBM [26] e da Microsoft [36], onde o projetista pode buscar informações sobre os Web Services e, então, definir quais deles serão utilizados nas atividades do workflow. No entanto, algumas características como pré-condições e pós-condições essenciais para o mapeamento em técnicas de planejamento não estão geralmente explícitas. Para realizar estes tratamentos e ao mesmo tempo ter um ambiente com liberdade de criação e modificação foi criado um módulo próprio de publicação destes registros.

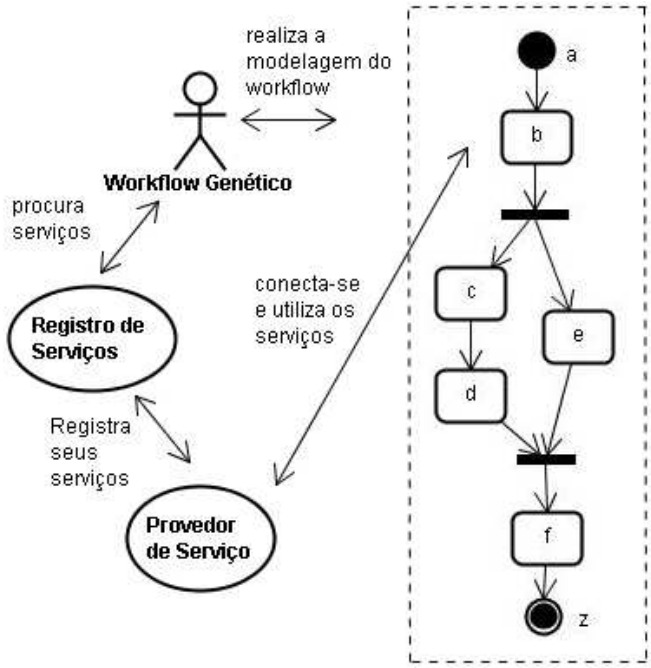


Figura 5.3: Iteração entre o Registro de Serviço, Provedor Serviço e Workflow Genético e o Workflow

A Figura 5.3 apresenta uma visão geral da relação existente entre o registro de serviços , o provedor de serviço, Workflow Genético e o processo de workflow. O Workflow Genético busca os registros de serviços disponíveis, realizando a modelagem dos processos. Essas atividades conectam, em tempo de execução, com os respectivos provedores de serviços.

5.2 Estudo de casos

Nesta seção, os conceitos e ferramentas apresentados em capítulos e seções anteriores são utilizados para desenvolvimento de estudo de casos, descritos através de uma breve descrição do contexto de aplicação, seguida pelo modelo de processo gerado. Os exemplos propostos não têm por objetivo serem completos, de modo a atender todas as possibilidades que podem existir no contexto do caso exposto. Mas sim, de ilustrar a utilização do ambiente da arquitetura proposta para execução da modelagem de processos para Workflow de Web Services. Na seção 5.2.1 será descrito o estudo de caso referente ao domínio Reclamação Telefônica. Na seção 5.2.2 será descrito o estudo de caso de submissão de artigo.

5.2.1 Estudo de Caso: Reclamação Telefônica

Este estudo de caso trata um cenário de processos de reclamação telefônica através da Internet. A modelagem do processo é realizada mediante as informações dos Web Services (atividades) publicadas no repositório UDDI. Os Web Services necessários para modelagem de processos de negócio são pesquisados no registro UDDI baseados na categoria do problema a qual pertencem. Os parâmetros de execução (pré-condições e pós-condições) e as informações necessárias para localização e execução dos Web Services, como por exemplo, URL, porta são informados na interface de publicação UDDI ou retiradas da própria descrição WSDL de cada Web Service.

A Figura 5.4 mostra uma tela comum da linguagem HTML para realizar entrada de informações do Web Service, podendo também anexar o arquivo WSDL correspondente. A descrição WSDL do Web Service é descrita na seção A.3 do apêndice. Nesta Figura estão destacadas as partes em que se obtém as informações de localização do Web Service. A informação para o *método* está destacada na

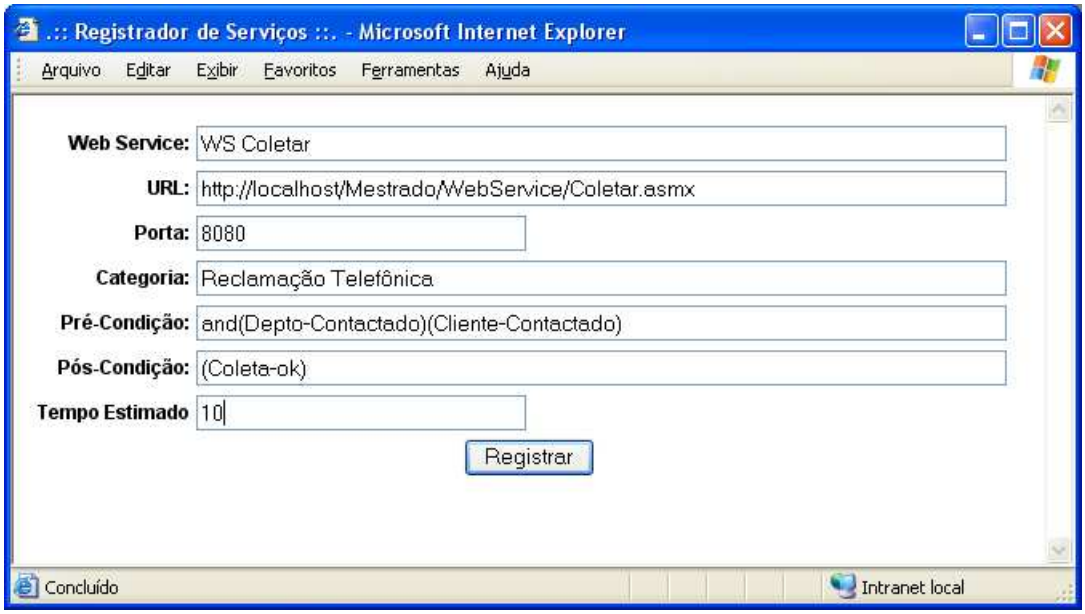


Figura 5.4: Formulário UDDI de publicação informações Web Service

marcação $< operationname = "Coletar»$, onde é definido o nome da operação que o serviço disponibiliza para cada tipo de porta. Neste exemplo é definido apenas com o serviço **Coletar**, podendo ter mais de um método para cada Web Service. Na tag $< servicename = "Sistema»$ é definido o **nome** do serviço, ou seja o executor dos métodos, seguido pelo nome da porta $< portname = "SistemaSoap"$ e localização (URL) $< soap : addresslocation = "http : //localhost/\ Mestrado/\ WebService/\ Coletar.asmx"/ >$

As informações das pré-condições e pós-condições serão informadas no módulo da interface UDDI da Figura 5.4, assim como tempo médio de execução. Note que não é obrigatório o uso de registro UDDI, porém deve-se fornecer um outro meio de divulgação das informações dos Web Services disponíveis na rede. A Tabela 5.1 descreve as atividades publicadas com as respectivas pré-condições e pós-condições de execução.

As probabilidade dos operadores genéticos são dadas por $\sharp Taxa\_Cruzamento = 0.75$, $\sharp Taxa\_Mutação = 0.15$ e $\sharp Taxa\_Elitismo = 0.10$. O tamanho da população é de 100 indivíduos e o número de gerações é $\sharp Nro\_Gerações = 10$ para obter o modelo de processo.

A Figura 5.5 ilustra a representação gráfica do processo criado pela arquitetura proposta para o domínio *Reclamação Telefônica*, fazendo uso dos Web Services

Tabela 5.1: Especificação das atividades do repositório

Atividades (Web Services)	**Pre-Condições**	**Pos-Condições**
Begin(A)		*início*
WS_Registar(B)	*início*	*Msg_Cliente e Msg_Depto*
WS_Contactar_Cliente(C)	*Msg_Cliente*	*Cliente_Contactado*
WS_Contactar_Departamento(D)	*Msg_Depto*	*Depto_Contactado*
WS_Coletar(E)	*Depto_Contactado e Cliente_Contactado*	*Coleta_OK*
WS_Avaliar(F)	*Coleta_OK*	*Positivo ou Negativo*
WS_Enviar_Carta(G)	*Negativo*	*Carta_Enviada*
WS_Pagar(H)	*Positivo*	*Pago*
WS_Arquivar(I)	*Carta_Enviada ou Pago*	*fim*
End(Z)	*fim*	

apropriados. Analisando os passos dos fluxos do processo verifica-se que o efeito da atividade genérica *Begin* é satisfeito pela pós-condição da atividade *WS_Registrar*, pois correspondem ao mesmo predicado *início*. Por sua vez, os efeitos *Msg_Cliente* e *Msg_Depto* da atividade *WS_Registrar* são satisfeitos pela pré-condição das atividades *WS_Contactar_Cliente* e *WS_Contactar_Departamento*, respectivamente, correspondendo a uma ramificação AND-Split. A atividade *WS_Contactar_Cliente* e a atividade *WS_Contactar_Departamento* tem como efeitos *Cliente_Contactado* e *Depto_Contactado*, que correspondem às pré-condições da atividade *WS_Coletar*.

Note que a atividade *WS_Coletar* necessita dos dois predicados *Cliente_Contactado* e *Depto_Contactado* para ser completamente satisfeita, correspondendo a uma ramificação AND-Join. A pré-condição da atividade *WS_Avaliar* é satisfeita pelos efeitos da atividade *WS_Coletar*. Os efeitos da atividade *WS_Avaliar*, corresponde a uma ramificação OR-Split pois podem ser *Positivo* ou *Negativo*, satisfazendo as atividades *WS_Enviar_Carta* e *WS_Pagar* respectivamente. A atividade *WS_Arquivar* tem como pre-condições *Carta_Enviada* ou *Pago*, satisfeitas pelos efeitos das atividades *WS_Enviar_Carta* e *WS_Pagar* respectivamente, correspondendo a uma ramificação OR-Join. Por fim, a atividade genérica *End* é satisfeita

pelos efeitos da atividade *WS_Arquivar*.

Este modelo, obtido a partir do Workflow Genético, quando confrontado com o diagrama de atividades da Figura 5.1 mostra-se correto, contemplando os roteamentos seqüenciais, paralelo e condicional previstos. Desta forma, o sistema mostra-se eficaz na geração da modelagem de processos. O modelo gerado é então, convertido para a linguagem XPDL e submetido a uma máquina de workflow. O modelo criado escrito na linguagem de definição de processo XPDL adotada se encontra na seção A.1 do apêndice A.

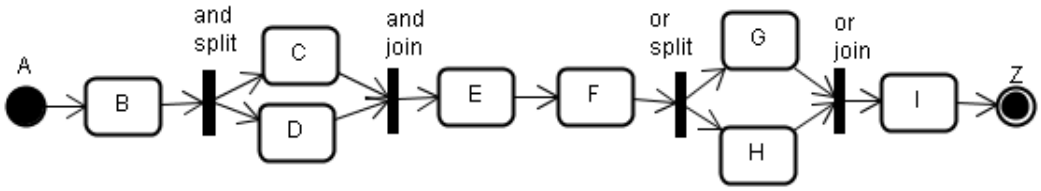


Figura 5.5: Processo Reclamação Telefônica

5.2.2 Estudo de Caso: Submissão de artigo

Este estudo de caso trata de um processo de submissão de artigos através da Web. O usuário deve acessar a página de submissão de artigos, preencher um formulário com informações sobre o artigo, como títulos, autores, instituição e endereço eletrônico para contato, dentre outras. Além de preencher tais informações, o usuário deve enviar o arquivo num formato específico para o servidor. Depois de enviar estas informações e o arquivo do artigo, verifica-se se a data de submissão não expirou. Caso a data limite já tenha sido ultrapassada, uma mensagem eletrônica é enviada para o e-mail de contato fornecido, notificando que a submissão foi rejeitada e o processo deve ser cancelado. Caso a submissão tenha sido realizada dentro do prazo, as informações do artigo são efetivamente cadastradas e em seguida uma mensagem é enviada para o e-mail de contato, notificando que a submissão do artigo foi recebida. O próximo passo é a seleção de revisores para o artigo submetido. Esta seleção é feita levando-se em consideração a área mais específica na qual o artigo se enquadra, cuja informação é passada pelo usuário na página de submissão. Os revisores também podem ser divididos por áreas, de modo que os revisores selecionados serão aqueles da mesma área do artigo. Essa seleção deve ser informada aos revisores via e-mail.

Então, deve-se aguardar até a data limite estipulada para os revisores cadastrarem suas respostas. Após a qual verifica-se os mesmos responderam. Em caso negativo, uma mensagem notificando o atraso é enviada para cada revisor que ainda não respondeu e se aguarda mais algum tempo. Isto deve ser realizado até que a resposta seja devidamente cadastrada por cada revisor. Por fim, após todos os revisores terem cadastrado suas respostas em relação à submissão do artigo, essas respostas são recuperadas e são enviadas para o e-mail de contato do autor, informando do parecer final de sua submissão. A Tabela 5.2 descreve as atividades publicadas para o problema, com as respectivas pré-condições e pós-condições de execução.

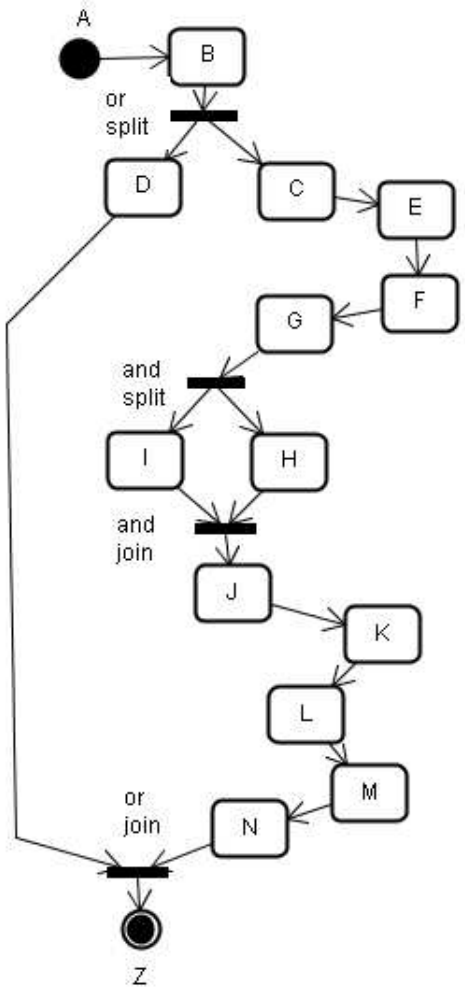


Figura 5.6: Processo Submissão de Artigo

A Figura 5.6 ilustra a representação gráfica do processo criado pelo Workflow Genético para o domínio "Submissão de Artigo", fazendo uso dos Web Services apro-

Tabela 5.2: Especificação das atividades do repositório para o domínio Submissão de Artigos

Atividades (Web Services)	**Pre-Condições**	**Pos-Condições**
Begin(A)		*início*
WS_Verificar_data_submissão(B)	*início*	*dentro_prazo ou fora_prazo*
WS_Cadastrar_Artigo(C)	*dentro_prazo*	*artigo_cadastrado*
WS_Enviar_msg_fora_prazo(D)	*fora_prazo*	*fim*
WS_Enviar_Confirmacao(E)	*artigo_cadastrado*	*msg_confirmacao*
WS_Selecionar_Revisores(F)	*msg_confirmacao*	*revisores_selecionados*
WS_Recuperar_msg_Revisores(G)	*revisores_selecionados*	*msg_revisores e msg_pagina*
WS_Enviar_Notificacao_Revisores(H)	*msg_revisores*	*notacao_revisor*
WS_Atualizar_Pagina_Artigos(I)	*msg_pagina*	*pagina_atualizada*
WS_Verificar_Limite_Resposta(J)	*notacao_revisor e pagina_atualizada*	*limite_verificado*
WS_Aguardar_Tempo(K)	*limite_verificado*	*tempo_esgotado*
WS_Verifica_Resposta_Revisores(L)	*tempo_esgotado*	*resposta_verificada*
WS_Recuperar_Resposta_Artigo(M)	*resposta_verificada*	*resposta_recuperada*
WS_Enviar_Resposta_Submissao(N)	*resposta_recuperada*	*fim*
End(Z)	*fim*	

priados. Uma observação no referido processo é o Web Service *WS_Aguardar_Tempo(K)* que funciona como *sleep*, ou seja, uma atividade que tem como função aguardar um determinado tempo para continuar o processo, ou poderia ser tratado como uma iteração para análise de determinado dados no processo.

5.3 Considerações finais

A simulação da execução dos modelos gerados nos experimentos foram realizados em uma rede local onde os serviços foram disponibilizados em diferente máquinas (provedores) e publicados em um registro UDDI.

Para desenvolvimento, teste e simulação dos estudo de casos foram utilizadas as seguintes ferramentas:

- Tomcat: Servidor Web desenvolvido pelo projeto Jakarta.
- Eclipse: IDE, desenvolvida pela IBM.
- AXIS: API para criação/utilização de Web Service criado pelo Jakarta.
- WSDL2Java: Plugin para eclipse que auxilia na criação de clientes para consumir o Web Service.
- Microsoft Visual Studio .NET 2003: Ferramenta utilizada para criação / utilização de Web Service criada pela Microsoft.

Também foi implementado um motor de workflow simples em Java que, a cada passo da execução do modelo de processo gerado, conecta e utiliza os serviços.

A análise do Workflow Genético através da execução no domínio Web apresenta grandes vantagens e os estudos de casos apresentados servem para ter uma visão de sua aplicação. Dentre as vantagens, temos por exemplo, que decisões de projeto que estejam encapsuladas dentro dos Web Services que compõem o processo podem ser revistas e alteradas sem afetar outras partes do processo. A flexibilidade de representação do Workflow Genético fornece suporte a qualquer domínio de aplicação, sejam aquelas relacionadas aos sistemas de produção, à entrega de um serviço ou à geração de conhecimento, podendo integrar serviços internos e serviços que transpõem as fronteiras organizacionais, bastando as implementações e publicações dos Web Services que executam o trabalho específico para um determinado domínio. Temos também que toda pessoa pode descobrir e publicar dinamicamente novos serviços através do repositório UDDI. Os Web Service fornecem todas as informações necessárias de localização e execução, não precisando ter conhecimento prévio de como foram implementados ou a plataforma em que estão sendo executados, fornecendo flexibilidade e adaptação para modelagem de processos.

Capítulo 6

Conclusão e Trabalhos Futuros

Em geral, o processo de modelagem de processos é realizado manualmente por especialistas em processos, não garantindo a adequação desejada, podendo estar sujeita a falhas e definições ineficientes. Além disso, a modelagem de processos complexos exige muito tempo e uma análise detalhada, devido ao grande número de atividades envolvidas.

Neste trabalho apresentou-se uma arquitetura para a geração automática de modelos de processos em sistema de workflow, integrando técnicas de Planejamento em IA e AG, com resultados promissores. Foi apresentada uma revisão dos principais conceitos da área de workflow e de planejamento em IA, com o objetivo de traçar uma estratégia para conexão entre as duas áreas. A semelhança entre processos de workflow e planos estão apoiadas no encadeamento de atividades e ações que buscam atender um objetivo específico.

Como os domínios de problemas em workflow do mundo real podem envolver inúmeras atividades de roteamento condicional, juntamente com atividades que representam convergência, o Workflow Genético mostrou mais adequado que o planejador POND nesta situação, uma vez que, o POND não conseguiu tratar situações com número elevado deste tipo de roteamento.

O planejador POND somente retornou fluxos que conduziram à meta, entrando em um laço infinito e não produzindo nenhum resultado em situações em que determinado fluxo conduz à meta e outro não, significando que apenas os fluxos que atingem a atividade final prevista são significativos. Entretanto, o modelo pode representar fluxos que não atingem a atividade final, permitindo que o Modelador

de processos reveja seu levantamento de atividades. No Workflow Genético, esta situação não foi identificada como um problema, desde que sempre se obtém uma solução. A ordem de execução de atividades em paralelo no resultado do fluxo do POND não é explícita, enquanto estas atividades ficam claramente expressas na representação do Workflow Genético.

A proposta é inovadora em relação a importantes aspectos não muito explorados, tais como, roteamentos OR-Split, OR-Join, AND-Split e AND-Join, que são essenciais para o tratamento de problemas de workflow no mundo real. A ferramenta SHAMASH, que corresponda um trabalho relacionado, realiza somente busca e geração de modelo seqüencial.

A integração dos modelos de processos com motores de workflow é automática, uma vez que estes já são gerados em uma representação XPDL, um padrão deste tipo de ferramenta. Esta característica é relevante para o desenvolvimento de modelos de processos.

A arquitetura pode ser aplicada a diferentes domínios do conhecimento, podendo ser considerada uma ferramenta adaptável para a geração automática de modelos de processos otimizados. A aplicação no domínio de Web Service, através da utilização de serviços disponíveis em uma rede de computadores, possibilita a integração tanto de aplicações internas, como de aplicações que transpõem as fronteiras organizacionais. Serviços podem ser facilmente localizados e modificados através de registros UDDI. Decisões de projeto que estejam encapsuladas em Web Services que compõem o processo podem ser revistas e alteradas sem afetar outras partes do processo, tornando os processos mais flexíveis e com uma maior liberdade para os recursos envolvidos nas atividades do processo.

A utilização de heurísticas para especializar a etapa de planejamento de acordo com o domínio, definição e aplicação de novos operadores genéticos, uso de outras representações para os indivíduos e introdução de técnicas de escalonamento durante o planejamento, são alguns aspectos a serem considerados em trabalhos futuros.

Referências Bibliográficas

[1] Aler, R., Borrajo, D., Camacho, D., and Alonso, A. S. A Knowledge-based Approach for Business Process Reengineering, SHAMASH. *Knowledge Based Systems 15*, 8 (2002), 473–483.

[2] Aler, R., Borrajo, D., and Isasi, P. Learning to Solve Planning Problems Efficiently by Means of Genetic Programming. *Evol. Comput. 9*, 4 (2001), 387–420.

[3] Allen, P. *Realizing e-business with components*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[4] Allen, R. *Workflow handbook 2001*. Future Strategies Inc., 2001, ch. Workflow: An Introduction.

[5] Alves, F. S., Guimaraes, K. F., and Fernandes, M. A. Modeling Workflow Systems with Genetic Planner and Scheduler. *ICTAI 0* (2006), 381–388.

[6] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., and Weerawarana, S. Business Process Execution Language for Web Services, 2003. Disponível em: *http://www-128.ibm.com/developerworks/library/specification/ws-bpel/* Acesso em: *Outubro 2006*.

[7] Aversano, L., and Canfora, G. Introducing eServices in Business Process Models. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering* (2002), ACM Press, pp. 481–488.

[8] Berry, P., and Drabble, B. Swim: An ai-based system for workflow enabled reactive control. In *proceedings of the IJCAI Workshop on Workflow and Process Management held as part of IJCAI-99* (1999), IJCAI, Stockholm, Sweden.

[9] Blum, A., and Furst, M. Fast Planning through Planning Graph Analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)* (1995), pp. 1636–1642.

[10] Bonet, B., and Geffner, H. Planning as Heuristic Search. *Artificial Intelligence 129*, 1-2 (2001), 5–33.

[11] Bryce, D., and Kambhampati, S. Cost Sensitive Reachability Heuristics for Handling State Uncertainty. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)* (Arlington, Virginia, 2005), AUAI Press, pp. 60–68.

[12] Casati, F., Ceri, S., Pernici, B., and Pozzi, G. Conceptual Modeling of Workflows. In *9th International Conference* (Gold Cost, Australia, 1995).

[13] Chiu, D. K. W., Karlapalem, K., and Li, Q. E-ADOME: Enacting Composite E-services in an Advanced Workflow Environment. In *COMPSAC '01: Proceedings of the 25th International Computer Software and Applications Conference on Invigorating Software Development* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 311–316.

[14] Edelkamp, S., and Hoffmann, J. Pddl2.2: The language for the Classical Part of the 4th International Planning Competition, institution = Albert-Ludwigs-Universitäit, Institut für Informatik Freiburg Germany, year = 2004, type = Technical Report, number = 195, address = Freiburg, German,. Tech. rep.

[15] Ernst, G. W., and Newell, A. GPS: A Case Study in Generality and Problem Solving. *Academic Press* (1969).

[16] Fábio Zschornack, N. E. Uma Linguagem para Representação de Workflow em XML com Suporte a Evolução e Versionamento de Esquemas. In *CLEI* -

XXVIII Conferencia Latino americana de Informática (Montevideo, Uruguai, 2002).

[17] Fikes, R. E., and Nilsson, N. J. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2 (1971), 189–208.

[18] Fox, M., and Long, D. Pddl+ : Planning with time and metric resources. Technical report, University of Durham, UK, 2001.

[19] Fox, M., and Long, D. Pddl2.1: An extension to pddl for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR) 20* (2003), 61–124.

[20] Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. PDDL - The Planning Domain Definition Language. Tech. Rep. CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University, 1998.

[21] Ginsberg, M. L. Computational Considerations in Reasoning about Action. In *KR* (1991), pp. 250–261.

[22] Gottschalk, K. D., Graham, S., Kreger, H., and Snell, J. Introduction to Web Services Architecture. *IBM Systems Journal 41*, 2 (2002), 170–177.

[23] Guimarães, K., and Fernandes, M. An Approach for Flexible Job-Shop Scheduling with Separable Sequence-Dependent Setup Time. In *IEEE International Conference on Systems, Man, and Cybernetics* (Taipei, Taiwan, 2006), pp. 3727–3731.

[24] Hoffmann, J., and Nebel, B. The FF Planning System: Fast Plan Generation through Heuristic Search. Tech. rep., AIPS Planning Competition, 2000.

[25] Holland, J. H. *Adaptation in natural and artificial systems.* University of Michigan Press, 1975.

[26] IBM. Web Services - UDDI. Disponível em: http://www-306.ibm.com/software/solutions/webservices/uddi/ Acesso em: Maio 2005.

[27] James Allen, J. H., and Tate, A. *Readings in Planning.* San Mateo, CA, 1990.

[28] Kautz, H., and Selman, B. Blackbox: A New Approach to the Application of Theorem Proving to Problem Solving. In *Workshop Planning as Combinatorial Search, AIPS-98* (Pittsburgh, PA, 1998).

[29] Kautz, H. A., and Selman, B. Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)* (1992), pp. 359–363.

[30] Knorr, K. WWW Workflows based on Petri Nets. In *Proceedings of the Ninth International Conference on Information Systems Development* (Kristiansand, Norway, 2000).

[31] Koza, J. R. *Genetic Programming II - Automatic Discovery of Reusable Programs.* Cambridge MA : Harvard University Press, 1994.

[32] Lecheta, E. M. Algoritmos Genéticos para Planejamento em Inteligência Artificial. Master's thesis, Universidade Federal do Paraná, 2004.

[33] Leymann F., Roller D., S. M. T. Web Services and Business Process Management. *IBM Systems Journal 41*, 2 (2002), 198–211.

[34] mendes de Araujo, R., and da Silva Borges, M. R. Sistemas de workflow. XX Jornada de Atualização em Informática, 2001. Congresso da Sociedade Brasileira de Computação,Fortaleza, Ceará, Brasil.

[35] Michalewicz, Z. *Genetic algorithms + data structures = evolution programs -3rd ed.* Springer-Verlag, London, UK, 1996.

[36] Microsoft. Web Services - UDDI. Disponível em: *http://test.uddi.microsoft.com/* Acesso em: *Setembro 2005.*

[37] Minton, S., Bresina, J. L., and Drummond, M. Total-Order and Partial-Order Planning: A Comparative Analysis. *Journal of Artificial Intelligence Research 2* (1994), 227–262.

[38] Minton, S., Knoblock, C., Kuokka, D. R., Gil, Y., Joseph, R. L., and Carbonell, J. G. Prodigy 2.0: The Manual and Tutorial. Tech. Rep. CMU-CS-89-146, School of Computer Science, Carnegie Mellon University, 1989.

[39] Mitchell, M. *An introduction to genetic algorithms.* MIT Press, Cambridge, MA, USA, 1996.

[40] Moreno, M. D. R., Borrajo, D., and Meziat, D. Process Modeling and AI Planning Techniques: A new approach. In *Second International Workshop on Information Integration and Web-based Applications and Services (IIWAS2000)* (Yogyakarta (Indonesia), 2000).

[41] Moreno, M. D. R., Kearney, P., and Meziat, D. A case study - Using Workflow and Planners. In *Nineteenth Workshop of the UK Planning and Scheduling Special Inteest Group- PLANSIG2000* (UK, 2000), pp. 175–184.

[42] Muslea, I. Sinergy: A linear planner based on genetic programming. In *ECP '97: Proceedings of the 4th European Conference on Planning* (London, UK, 1997), Springer-Verlag, pp. 312–324.

[43] Myers, K. L., and Berry, P. M. Workflow Management Systems: An AI Perspective. Tech. rep., Artificial Intelligence Center, SRI International, Menlo Park, CA, 1998.

[44] Pednault, E. Adl: Exploring the Middle Ground between STRIPS and the Situation Calculus. Tech. Rep. pp.324-332, International Conference on Principles of Knowledge Representation and Reasoning, Toronto, 1989.

[45] Planet. PLANET Workflow Management R&D RoadMap. Disponível em: *www.planet-noe.org* Acesso em: *Outubro 2006.*

[46] Russell, S., and Norvig, P. *Artificial Intelligence:A Modern Approach.* Prentice-Hall, Englewood Cliffs, NJ, 1995.

[47] Schoppers, M. Universal Plans for Reactive Robots in Unpredictable Environments. In *IJCAI* (1987), pp. 1039–1046.

[48] Spector, L. Genetic Programming and AI Planning Systems. In *AAAI* (1994), pp. 1329–1334.

[49] Sun. The Java Web Services Tutorial - For Java Web Services Developer's Pack. Disponível em: *http:// java.sun.com /webservices/ docs /1.6/ tutorial /doc /index.html* Acesso em: *Outubro 2006*.

[50] Sun. Extensible markup language *xml* 1.0, 2004. Disponível em: *http://www.w3.org/TR/REC-xml/* Acesso em: *Outubro 2006*.

[51] Sun. Soap version 1.2 part 0: Primer, 2004. Disponível em: *http://www.w3.org/TR/soap12-part0/* Acesso em: *Outubro 2006*.

[52] Sutton, M. J. D. *Document Management for the Enterprise : Principles, Techniques, and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[53] Syswerda, G. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms* (San Francisco, CA, USA, 1989), Morgan Kaufmann Publishers Inc., pp. 2–9.

[54] T.Bylander. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence 69*, 1-2 (1994), 165–204.

[55] Tramontina, G. B. Análise de Problemas de Escalonamento de Processos em Workflow, school = Instituto de Computação, Universidade de Campinas (UNICAMP), year = 2004, address = Campinas, SP, Brazil. Master's thesis.

[56] UDDI, O. Advancing Web Services Discovery Standard. Disponível em: *http://www.uddi.org/* Acesso em: *Junho 2006*.

[57] van der Aalst, W., and van Hee, K. *Workflow Management: Models, Methods and Systems*. The MIT Press, 2002.

[58] W3C. Web Services Description Language (wsdl). Disponível em: *http://www.w3.org/TR/wsdl* Acesso em: *Julho 2006*.

[59] Westerberg, C. H., and Levine, J. Genplan: Combining genetic programming and planning, 2000.

[60] WfMC. The workflow reference model. Tech. Rep. WfMC-TC00-1003, Workflow Management Coalition, Hampshire, Reino Unido, Janeiro 1995.

[61] WfMC. Terminology and Glossary. Tech. Rep. WfMC-TC-1011, Workflow Management Coalition, Hampshire, UK, 1999.

[62] Wohed, P., van der Aalst, W. M., Dumas, M., and ter Hofstede, A. H. Pattern Based Analysis of BPEL4WS, 2002.

[63] YU, T. Structure Abstraction and Genetic Programming. *Congress on Evolutionary Computation (CEC99)* (1999), 652–659.

[64] Zeng, D. D., and Zhao, J. L. Effective Role Resolution in Workflow Management. *INFORMS journal on computing 17*, 3 (2005), 374–387. ISSN: 1091-9856.

[65] zur Muehlen, M. Resource Modeling in Workflow Applications. In *Proceedings of the 1999 Workflow Management Conference* (novembro 1999), pp. 137–153.

Apêndice A

Anexo arquivos

A.1 Arquivo XPDL - estudo de caso Reclamação Telefônica

Esta seção apresenta o modelo de processo XPDL gerado pela arquitetura no estudo de caso "Reclamação Telefônica"do capítulo 5 da seção 5.2.1.

```
<?xml version="1.0" encoding="UTF-8"?> <Package
Id="ReclamacaoTelefonica" Name="ReclamaÃ§Ã£o TelefÃ´nica"
xmlns="http://www.wfmc.org/2002/XPDL1.0"
xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0
http://wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd">
    <PackageHeader>
        <XPDLVersion>1.0</XPDLVersion>
        <Vendor>Together</Vendor>
        <Created>2005-09-30 08:58:27</Created>
    </PackageHeader>
    <RedefinableHeader PublicationStatus="UNDER_TEST"/>
    <ConformanceClass GraphConformance="NON_BLOCKED"/>
    <WorkflowProcesses>
        <WorkflowProcess AccessLevel="PUBLIC" Id="Operacoes" Name="Operacoes">
            <ProcessHeader DurationUnit="D">
                <Created>2005-09-30 08:58:56</Created>
```

```
</ProcessHeader>
<RedefinableHeader PublicationStatus="UNDER_TEST"/>
<Participants>
    <Participant Id="Secretaria" Name="WS Secretaria">
        <ParticipantType Type="ROLE"/>
    </Participant>
    <Participant Id="Sistema" Name="WS Sistema">
        <ParticipantType Type="ROLE"/>
    </Participant>
    <Participant Id="FuncionarioReclamacoes"
                Name="WS Funcionario Reclamacoes">
        <ParticipantType Type="ROLE"/>
    </Participant>
    <Participant Id="AnalistaReclamacoes"
                Name="WS Analista Reclamacoes">
        <ParticipantType Type="ROLE"/>
    </Participant>
    <Participant Id="FuncionarioFinancas"
                Name="WS FuncionÃ<rio Financas">
        <ParticipantType Type="ROLE"/>
    </Participant>
</Participants>
<Activities>
    <Activity Id="registrar" Name="registrar">
        <Implementation>
            <No/>
        </Implementation>
        <Performer>Secretaria</Performer>
        <StartMode>
            <Automatic/>
        </StartMode>
        <FinishMode>
```

```
                    <Automatic/>
                </FinishMode>
                <SimulationInformation>
                    <Cost>Alto</Cost>
                    <TimeEstimation>
                        <Duration>10</Duration>
                    </TimeEstimation>
                </SimulationInformation>
                <TransitionRestrictions>
                    <TransitionRestriction>
                        <Split Type="AND">
                            <TransitionRefs>
                                <TransitionRef Id="Operacoes_tra2"/>
                                <TransitionRef Id="Operacoes_tra1"/>
                            </TransitionRefs>
                        </Split>
                    </TransitionRestriction>
                </TransitionRestrictions>
                <ExtendedAttributes>
                    <ExtendedAttribute Name="Condicao" Value="?x">
                        "IF" start
                        "THEN"novareclamacao</ExtendedAttribute>
                </ExtendedAttributes>
            </Activity>
            <Activity Id="contactarcepto" Name="contactar depto">
                <Implementation>
                    <No/>
                </Implementation>
                <Performer>FuncionarioReclamacoes</Performer>
                <StartMode>
                    <Automatic/>
                </StartMode>
```

```
                <FinishMode>
                    <Automatic/>
                </FinishMode>
                <SimulationInformation>
                    <Cost>Alto</Cost>
                    <TimeEstimation>
                        <Duration>10</Duration>
                    </TimeEstimation>
                </SimulationInformation>
                <ExtendedAttributes>
                    <ExtendedAttribute Name="Condicao" Value="?x">
                        "IF" novareclamacao
                        "THEN" deptocontactato</ExtendedAttribute>
                </ExtendedAttributes>
            </Activity>
            <Activity Id="contactarcliente" Name="contactar cliente">
                <Implementation>
                    <No/>
                </Implementation>
                <Performer>FuncionarioReclamacoes</Performer>
                <StartMode>
                    <Automatic/>
                </StartMode>
                <FinishMode>
                    <Automatic/>
                </FinishMode>
                <SimulationInformation>
                    <Cost>Alto</Cost>
                    <TimeEstimation>
                        <Duration>10</Duration>
                    </TimeEstimation>
                </SimulationInformation>
```

```
            <ExtendedAttributes>
                <ExtendedAttribute Name="Condicao" Value="?x">
                   "IF" novareclamacao
                   "THEN" clientecontactado</ExtendedAttribute>
            </ExtendedAttributes>
        </Activity>
        <Activity Id="enviarcarta" Name="enviar carta">
            <Implementation>
                <No/>
            </Implementation>
            <Performer>FuncionarioReclamacoes</Performer>
            <StartMode>
                <Automatic/>
            </StartMode>
            <FinishMode>
                <Automatic/>
            </FinishMode>
            <SimulationInformation>
                <Cost>Alto</Cost>
                <TimeEstimation>
                    <Duration>10</Duration>
                </TimeEstimation>
            </SimulationInformation>
            <ExtendedAttributes>
                <ExtendedAttribute Name="Condicao" Value="?x">
                    "IF" negativo
                    "THEN" cartaenviada</ExtendedAttribute>
            </ExtendedAttributes>
        </Activity>
        <Activity Id="coletar" Name="coletar">
            <Implementation>
                <No/>
```

```
            </Implementation>
            <Performer>Sistema</Performer>
            <StartMode>
                <Automatic/>
            </StartMode>
            <FinishMode>
                <Automatic/>
            </FinishMode>
            <SimulationInformation>
                <Cost>Alto</Cost>
                <TimeEstimation>
                    <Duration>10</Duration>
                </TimeEstimation>
            </SimulationInformation>
            <TransitionRestrictions>
                <TransitionRestriction>
                    <Join Type="AND"/>
                </TransitionRestriction>
            </TransitionRestrictions>
            <ExtendedAttributes>
                <ExtendedAttribute Name="Condicao" Value="?x">
                   "IF" deptocontactado "^" clientecontactado
                   "THEN" coletaok</ExtendedAttribute>
            </ExtendedAttributes>
        </Activity>
        <Activity Id="arquivar" Name="arquivar">
            <Implementation>
                <No/>
            </Implementation>
            <Performer>Sistema</Performer>
            <StartMode>
                <Automatic/>
```

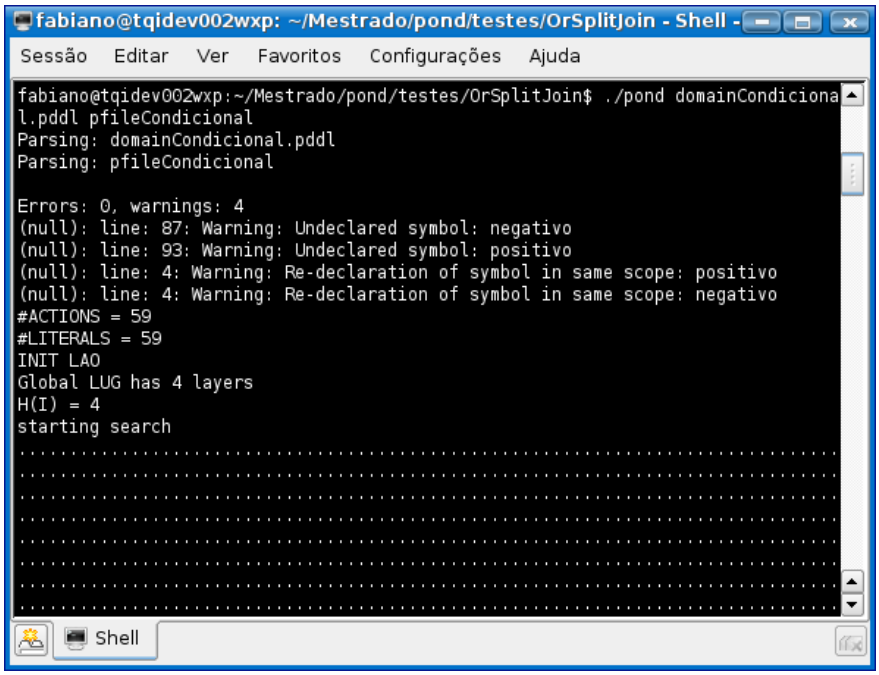
```
            </StartMode>
            <FinishMode>
                <Automatic/>
            </FinishMode>
            <SimulationInformation>
                <Cost>Alto</Cost>
                <TimeEstimation>
                    <Duration>10</Duration>
                </TimeEstimation>
            </SimulationInformation>
            <TransitionRestrictions>
                <TransitionRestriction>
                    <Join Type="XOR"/>
                </TransitionRestriction>
            </TransitionRestrictions>
            <ExtendedAttributes>
                <ExtendedAttribute Name="Condicao" Value="?x">
                  "IF" cartaenviada "|" pago
                  "THEN" fim</ExtendedAttribute>
            </ExtendedAttributes>
        </Activity>
        <Activity Id="avaliar" Name="avaliar">
            <Implementation>
                <No/>
            </Implementation>
            <Performer>AnalistaReclamacoes</Performer>
            <StartMode>
                <Automatic/>
            </StartMode>
            <FinishMode>
                <Automatic/>
            </FinishMode>
```

```
                <SimulationInformation>
                    <Cost>Alto</Cost>
                    <TimeEstimation>
                        <Duration>10</Duration>
                    </TimeEstimation>
                </SimulationInformation>
                <TransitionRestrictions>
                    <TransitionRestriction>
                        <Split Type="XOR">
                            <TransitionRefs>
                                <TransitionRef Id="Operacoes_tra6"/>
                                <TransitionRef Id="Operacoes_tra7"/>
                            </TransitionRefs>
                        </Split>
                    </TransitionRestriction>
                </TransitionRestrictions>
                <ExtendedAttributes>
                    <ExtendedAttribute Name="Condicao" Value="?x">
                    "IF" coletaok
                    "THEN" negativo "|" positivo</ExtendedAttribute>
                </ExtendedAttributes>
            </Activity>
            <Activity Id="pagar" Name="pagar">
                <Implementation>
                    <No/>
                </Implementation>
                <Performer>FuncionarioFinancas</Performer>
                <StartMode>
                    <Automatic/>
                </StartMode>
                <FinishMode>
                    <Automatic/>
```

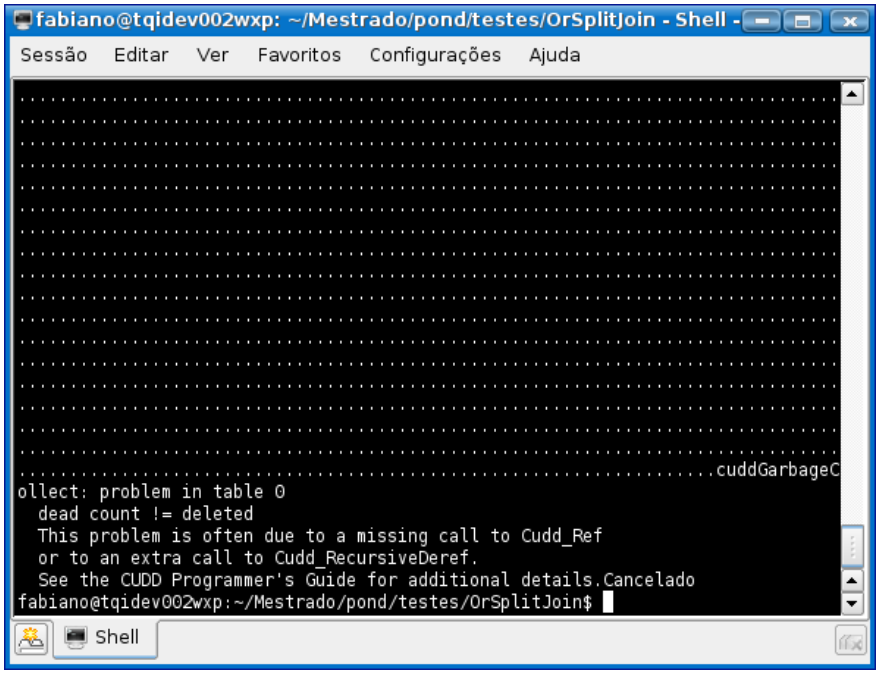
```
                </FinishMode>
                <SimulationInformation>
                    <Cost>Alto</Cost>
                    <TimeEstimation>
                        <Duration>10</Duration>
                    </TimeEstimation>
                </SimulationInformation>
                <ExtendedAttributes>
                    <ExtendedAttribute Name="Condicao" Value="?x">
                    "IF" positivo
                    "THEN" pago</ExtendedAttribute>
                </ExtendedAttributes>
            </Activity>
        </Activities>
        <Transitions>
            <Transition From="registrar" Id="Operacoes_tra1"
                        To="contactarcliente"/>
            <Transition From="registrar" Id="Operacoes_tra2"
                        To="contactarcepto"/>
            <Transition From="contactarcepto" Id="Operacoes_tra3"
                        To="coletar"/>
            <Transition From="contactarcliente" Id="Operacoes_tra4"
                        To="coletar"/>
            <Transition From="coletar" Id="Operacoes_tra5"
                        To="avaliar"/>
            <Transition From="avaliar" Id="Operacoes_tra6"
                        To="enviarcarta"/>
            <Transition From="avaliar" Id="Operacoes_tra7"
                        To="pagar"/>
            <Transition From="pagar" Id="Operacoes_tra8"
                        To="arquivar"/>
            <Transition From="enviarcarta" Id="Operacoes_tra9"
```

```
                                To="arquivar"/>
            </Transitions>
        </WorkflowProcess>
    </WorkflowProcesses>
</Package>
```

A.2 Mensagem erro na execução do planejador POND



(a)



(b)

Figura A.1: Mensagem de erro no processamento de execução do planejador POND

A.3 Descrição WSDL referente ao web service Coletar

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://tempuri.org/WebService/Sistema"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://tempuri.org/WebService/Sistema"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/WebService/Sistema">
      - <s:element name="Coletar">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="Depto_Contactado" type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="Cliente_Contactado" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="ColetarResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="ColetarResult" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
      </s:schema>
    </types>
  - <message name="ColetarSoapIn">
      <part name="parameters" element="s0:Coletar" />
    </message>
  - <message name="ColetarSoapOut">
      <part name="parameters" element="s0:ColetarResponse" />
    </message>
  - <portType name="SistemaSoap">
    - <operation name="Coletar">
        <input message="s0:ColetarSoapIn" />
        <output message="s0:ColetarSoapOut" />
      </operation>
    </portType>
  - <binding name="SistemaSoap" type="s0:SistemaSoap">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
    - <operation name="Coletar">
        <soap:operation soapAction="http://tempuri.org/WebService/Sistema/Coletar" style="document" />
      - <input>
          <soap:body use="literal" />
        </input>
      - <output>
          <soap:body use="literal" />
        </output>
      </operation>
    </binding>
  - <service name="Sistema">
    - <port name="SistemaSoap" binding="s0:SistemaSoap">
        <soap:address location="http://localhost/Mestrado/WebService/Coletar.asmx" />
      </port>
    </service>
</definitions>
```

Figura A.2: Descrição WSDL referente ao Web Service Coletar