UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

WORKFLOW COM TÉCNICAS DE PLANEJAMENTO APOIADO EM INTELIGÊNCIA ARTIFICIAL

Jony Teixeira de Melo

Setembro

2005

Jony Teixeira de Melo

Workflow com Técnicas de Planejamento Apoiado em Inteligência Artificial

Faculdade de Computação

Universidade Federal de Uberlândia

2005

Jony Teixeira de Melo

Workflow com Técnicas de Planejamento Apoiado em Inteligência Artificial

Dissertação apresentada como parte dos requisitos exigidos para a obtenção do grau de Mestre em Ciência da Computação junto à Universidade Federal de Uberlândia, Minas Gerais.

© Todos os direitos reservados

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Workflow com Técnicas de Planejamento Apoiado em Inteligência Artificial**" por **Jony Teixeira de Melo** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 19 de Setembro de 2005

Orientador: _____
Prof. Dr. Carlos Roberto Lopes
Universidade Federal de Uberlândia UFU/MG

Banca Examinadora: _____
Prof. Dr. Jacques Wainer
Universidade Estadual de Campinas
UNICAMP/SP

_____
Prof. Dr. Stéphane Julia
Universidade Federal de Uberlândia UFU/MG

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Data: Setembro de 2005

Autor: **Jony Teixeira de Melo**

Título: **Workflow com técnicas de Planejamento apoiado em Inteligência Artificial**

Faculdade: **Faculdade de Computação**

Grau: **Mestre** Convocação: **Setembro** Ano: **2005**

A Universidade Federal de Uberlândia possui permissão para distribuir e ter cópias desse documento para propósitos exclusivamente acadêmicos, desde que a autoria seja devidamente divulgada.

_____

Autor

O AUTOR RESERVA OS DIREITOS DE PUBLICAÇÃO E ESSE DOCUMENTO NÃO PODE SER IMPRESSO OU REPRODUZIDO DE OUTRA FORMA, SEJA NA TOTALIDADE OU EM PARTES SEM A PERMISSÃO ESCRITA DO AUTOR.

À minha esposa Cláudia, pelo apoio e carinho em todos os momentos.

Agradecimentos

Agradeço ao Prof. Carlos pelo apoio dado durante todo este trabalho e pela oportunidade de executá-lo.

Ao meu grande amigo Umberto pela amizade sempre presente, pelo incentivo e companheirismo nesta etapa que vencemos juntos, e nos diversos projetos que realizamos durante este período.

Agradeço também aos amigos e colegas da UFU com os quais tive o privilégio de conviver nestes anos. Aos Professores do programa de Mestrado, pelo conhecimento e experiência transferidos.

Aos amigos da CTBC que sempre apoiaram esta caminhada. Em especial ao incentivo do Adriano, Lena, Germano, Reginaldo, Melissa, Maiymi e Darlene.

Ao meu professor de inglês John Joe O'Connell, pelas aulas e apoio.

Agradeço à minha família e amigos pela compreensão da minha ausência em diversos momentos. À minha mãe, Natalina. Também à Jany, Diony, Matheus, Were, Paulinho, D.Dina, Mariley, Pedro, Renato e Juninho.

Agradeço a Deus por mais esta realização.

Resumo

Este trabalho apresenta uma nova abordagem para o desenvolvimento de Sistemas de Workflow baseada em técnicas de planejamento apoiadas em Inteligência Artificial. A abordagem proposta concentra-se no campo de modelagem de processos. A ferramenta de planejamento é utilizada para gerar automaticamente modelos de processos consistentes. Os processos modelados geralmente possuem atividades que ramificam o fluxo em mais de um caminho possível. A modelagem de tais processos via planejamento exige a obtenção de planos condicionais. Este trabalho apresenta algoritmos que permitem obter planos condicionais a partir de planejadores clássicos. Finalmente, são realizados experimentos para medir a eficiência e eficácia destes algoritmos, e os resultados são comparados com um planejador contingente estado-da-arte.

Palavras-chaves: Workflow, Modelagem, Inteligência Artificial, Planejamento.

Abstract

This work presents a new approach that uses Artificial Intelligence Planning techniques in the development of a Workflow Modelling System. This approach concentrates on the field of process modelling. The planning tool is used to automatically generate models of consistent processes. The modeled processes usually have activities that split the flow into more than one possible branch. The use of planner to perform this modelling process, exacts conditional plans. This work presents algorithms give rise to conditional plans when using classic planners. Finally, experiments are carried out to measure the efficiency and effectiveness of these algorithms, and the results are compared to those of a contingent *state-of-the-art* planner.

Keywords: Workflow, Modeling, Artificial Intelligence, Planning.

Conteúdo

1 **Introdução** **1**

2 **Workflow** **4**

2.1 Introdução 4

2.2 Workflows e Processos de Negócios 5

2.3 Sistema de Gerenciamento de Workflow 7

2.4 Terminologia 8

2.5 Arquitetura de um WfMS 9

2.6 Funcionalidades de um Sistema de Workflow 13

2.6.1 Modelagem de Processos 13

2.6.2 Execução 17

2.6.3 Acompanhamento e Gerenciamento 17

2.7 Tipos de Workflow 18

2.7.1 Classificação Quanto à Abordagem de Comunicação 20

2.7.2 Classificação Quanto à Orientação 21

2.8 Considerações Finais 22

3 **Planejamento Apoiado em IA** **23**

3.1 Conceitos de Planejamento 23

3.1.1 Mundo dos Blocos 25

3.1.2 A Linguagem dos Problemas de Planejamento 27

3.2 Abordagens de Planejamento 32

3.2.1 STRIPS 32

3.2.2 Busca no Espaço de Estados 34

3.2.3 Planejamento de Ordem Parcial (POP) 40
3.2.4 Planejamento por Grafos 45
3.2.5 Planejamento por Satisfabilidade 49
3.3 Planejamento em Ambientes Dinâmicos 50
3.3.1 Monitoramento e Replanejamento 50
3.3.2 Planejamento Reativo 51
3.3.3 Planejamento Contingente 53
3.3.4 Planejamento Conformante 54
3.4 Considerações Finais 55

4 **Usando o Planejamento para Modelagem de Workflow 56**
4.1 Infra-estrutura da Nova Abordagem 57
4.2 Formalização do Planejamento 59
4.3 Atividades versus Operadores 60
4.4 Conexões entre Atividades 61
4.5 Mapeamento de Atividades em Operadores 63
4.6 Planejando o Processo de Workflow 65
4.6.1 Fluxo Paralelo 66
4.6.2 Fluxo Condicional 69
4.7 Implementação dos Algoritmos 70
4.7.1 Discussão dos Resultados 72
4.8 Trabalhos Relacionados 75
4.9 Considerações Finais 77

5 **Conclusão 78**

Lista de Figuras

Figura 1: Estrutura genérica de sistemas de workflow (fonte WfMC). 10
Figura 2: Arquitetura / Modelo de referência (fonte WfMC) 11
Figura 3: Definição básica de processo (fonte WfMC). 14
Figura 4: Itens da representação gráfica de workflow 16
Figura 5: Tipos de encadeamento de atividades. 17
Figura 6: Workflow do tipo a*d-hoc* – revisão de artigos. 19
Figura 7: Workflow do tipo Administrativo – revisão de artigos. 19
Figura 8: Relacionamento entre tipos de workflow. 21
Figura 9: Exemplo de um plano para a solução do problema do pneu furado..... 24
Figura 10: Operadores no domínio do mundo dos blocos. 25
Figura 11: Problema do mundo dos blocos para empilhar três blocos (A / B / C). 26
Figura 12: Anomalia de Sussman. 34
Figura 13: Ilustração das duas abordagens de busca..... 35
Figura 14: Algoritmo de um planejador regressivo. 37
Figura 15: Problema de planejamento descrito em ADL..... 41
Figura 16:Plano de ordem-parcial e suas linearizações para o problema das meias e sapatos..... 42
Figura 17:Estado final do planejamento do problema das meias e sapatos. 43
Figura 18: Representação de um grafo de três camadas para um problema do mundo dos blocos. 46
Figura 19: Relações *mutex* em um grafo de planejamento. 47
Figura 20: Algoritmo *GraphPlan* (RUSSELL; NORVIG, 2003). 48
Figura 21: Arquitetura do planejador SATplan. 50
Figura 22: Esquema de um planejador reativo primitivo. 52

Figura 23: Esquema de integração de AI Planning com o workflow. 58
Figura 24: Tela de mapeamento de processos do JAWE..... 59
Figura 25: Comparação entre atividade em XPDL e ação em PDDL..... 61
Figura 26: Modelo de ramificação de atividades..... 62
Figura 27: Desdobramento de atividades *OR-Split*..... 63
Figura 28: Algoritmo: Desdobramento de atividades..... 64
Figura 29: Exemplo de um plano com atividades em paralelo..... 66
Figura 30: Algoritmo: Extração do modelo do plano..... 67
Figura 31: Algoritmo: Extração de plano condicional..... 69
Figura 32: Tela do Sistema de Modelagem Automática de workflow - *SisMAP*. 71
Figura 33: Diagrama de Atividades para o "Serviço de Reclamações"..... 71
Figura 34: Segmento da definição de um problema de planejamento condicional em PDDL utilizado pelo POND. 73
Figura 35: Tempo de execução versus quantidade de atividades *OR-SPLIT*. 74
Figura 36: Tempo de execução versus quantidade de operadores..... 75

Lista de Tabelas

Tabela 1 - Comparação entre as linguagens STRIPS e ADL 30
Tabela 2 : Especificação das atividades do processo..... 72

Lista de Acrônimos

ADL – *Action Description Language.*

AI – Artificial Intelligence

APIs – *Application Programming Interfaces*

ASCII – *American Standard Code for Information Interchange*

BDDs – *Binary Decision Diagrams*

BPR – *Business Process Re-engineering*

COSMOSS – *Customer Orientated System for the Management Of Special Services*

CRM – *Customer Relationship Management*

CSCW – *Computer Supported Cooperative Work*

DNA – *DeoxyriboNucleic Acid*

EAI – *Enterprise Application Integration*

EBNF – *Extended Backus Normal Form*

ERP – *Enterprise Resource Planning*

FF – *Fast-Foward planner*

FNC – Forma Normal Conjuntiva

FND – Forma Normal Disjuntiva

GSat – *Greedy local search procedure*

HSP – *Heuristic Search Planner*

IA – Inteligência Artificial

LPO – Lógica de Primeira Ordem

LUG – *Labelled Uncertainty Graph*

MDPs – *Markov Decision Processe*

PDDL – *Planning Domain Definition Language*

PDL – *Planning Description Language*

POND – *Partially Observable Non-Deterministic planner*

POP – *Planejamento de Ordem Parcial*

SAT – *SATisfiability*

SCR – *Situated Control Rules*

SGBD – Sistema Gerenciador de Banco de Dados

STRIPS – ST*tanford* R*esearch* I*nstitute* P*roblem* S*olver*

TCU – *Techinical Coordination Unit*

WfMS – *Workflow Management System*

WPDL – *Workflow Process Definition Language*

XML – *Extensible Markup Language*

XPDL – *XML Processing Description Language*

Capítulo 1

Introdução

A execução de uma atividade complexa, seja ela a produção de um bem, a entrega de um serviço ou geração de conhecimento, exige geralmente a participação de mais de um indivíduo para atingir um objetivo. Isto é um fato comum dentro de nossa sociedade, nas empresas, escolas, hospitais etc. Estas organizações distribuem suas atividades em uma seqüência de tarefas, cujo conjunto pode ser visto como um processo. Em especial nas empresas temos os processos de negócio que são conjuntos de tarefas que cooperam para atingir um resultado. Estes processos podem envolver inúmeras áreas internas à organização, além de clientes, fornecedores e parceiros de negócio.

No estágio atual de desenvolvimento da sociedade humana, as pessoas estão exigindo um nível cada vez maior de qualidade, além de agilidade e confiabilidade na execução de serviços. Isto faz surgir uma necessidade de tratar o trabalho em equipe, o gerenciamento e reuso da informação e o aperfeiçoamento constante dos processos de produção. Neste contexto as ferramentas de *workflow*[1] têm um papel importante, pois estão associadas à automação de processos. Tais ferramentas tiveram sua origem nas pesquisas de automação de escritórios da década de 70 quando o principal objetivo era oferecer soluções para gerar, distribuir e compartilhar documentos dentro de uma organização (SUTTON, 1996). Na década de 80, sofreram a influência das pesquisas de *groupware* e trabalho cooperativo apoiado em computadores, conhecido pela sigla CSCW (do inglês *Computer Supported Cooperative Work*), que tornou as ferramentas de *workflow* um instrumento de coordenação

[1] O termo *workflow* é usado na maioria das vezes significando processos de negócio, porém esta expressão não traduz todos os significados do termo original. Por isso fez-se a opção por manter o termo em inglês para maior clareza. O seu conceito será definido no capítulo 2.

de trabalho em equipe. Na década de 90, continuou sua evolução tirando vantagem do rápido crescimento das redes de computadores (*Internet*/*Intranet*).

Recentemente, tem crescido o interesse em aplicar técnicas de inteligência artificial (IA) em sistemas tradicionais, para solucionar problemas comuns. Existe um interesse especial nas técnicas que envolvem planejamento[2], as quais já são bastante utilizadas em áreas especializadas como: missões espaciais, robótica, planejamento de missões militares, controle de elevadores etc (PLANET, 2003; RUSSELL; NORVIG, 2003). Mas muitas outras áreas podem tirar proveito da possibilidade de automação oferecida pela aplicação de uma tecnologia de *AI Planning*. Nesta linha encontra-se o PLANET[3], cujo objetivo é fomentar o desenvolvimento e integração desta tecnologia com diversas áreas. Esta entidade foi organizada em unidades menores, chamadas Unidades de Coordenação Técnica - *TCU*[4], responsáveis por conduzir o desenvolvimento em cada área. Uma das linhas de trabalho promove a aplicação efetiva das técnicas de planejamento e escalonamento aos sistemas de *workflow*. Este trabalho baseia-se nas diretrizes apresentadas no roteiro que orienta as pesquisas desta área (PLANET, 2003), e visa integração entre a modelagem de processos de *workflow* e ferramentas de planejamento.

As empresas são umas das principais áreas de uso de ferramentas de workflow e atuam em um ambiente marcado por incertezas e constantes mudanças. Estas mudanças são impostas pelo mercado; causadas por exigências dos clientes, ações da concorrência, ou por alterações regulatórias em decorrência de novas leis ou políticas governamentais. Isto gera uma demanda por dois tipos de ferramentas: Uma que permita automaticamente modelar, modificar, simular e otimizar processos existentes de acordo com as regras de negócio da empresa. E outra para executar, monitorar e adaptar-se dinamicamente as modificações dos processos gerados pela primeira. Estes dois tipos de ferramentas constituem as estruturas principais de uma aplicação de workflow. As técnicas de *AI Planning* ajudam a tornar o processo de modelagem ágil, preciso e otimizado; bem como tornar a execução adaptável.

Este trabalho apresenta uma contribuição para o aperfeiçoamento da integração entre sistema de workflow e planejamento. O texto está estruturado como segue. No capítulo 2,

---

[2] Planejamento neste contexto refere-se a uma subárea de Inteligência Artificial (do Inglês *Artificial Intelligence Planning*).

[3] O PLANET é uma das "redes de excelência" financiada pela IST, Information Society Technologies, programa da União Européia. Criado em 1998, o qual reúne atualmente aproximadamente 200 pesquisadores e profissionais de universidades, centros de pesquisa e indústrias; em mais de 18 países.

[4] TCU do inglês *Techinical Coordination Unit*.

primeiro é feita uma apresentação dos conceitos de workflow, a terminologia usada, a arquitetura subjacente a estas aplicações e sua classificação. O capítulo 3 apresenta os conceitos de planejamento apoiado em IA e discute as linguagens e formalismos do planejamento clássico e dinâmico. O capítulo 4 apresenta a arquitetura sobre a qual é construída a proposta de integração de planejamento e workflow. A proposta deste trabalho também é detalhada no capítulo 4, onde são apresentados os algoritmos propostos e resultados da sua aplicação, bem como os trabalhos relacionados. Por fim, no capítulo 5 é apresentada a conclusão e trabalhos futuros.

Capítulo 2

Workflow

Este capítulo apresenta os conceitos básicos de workflow, terminologias e características. Também são mostrados os tipos de workflow mais comuns e discute-se o processo de modelagem.

2.1 Introdução

Uma introdução interessante aos conceitos de workflow é feita por Plesums (2002), na qual o autor visualiza uma cena na idade média com monges sentados em suas mesas, cuidadosamente copiando escrituras. Um monge mais graduado deveria dirigir os trabalhos, dividindo e atribuindo atividades aos demais, provavelmente designando a primeira página, a que possuía ilustrações e maior nível de detalhes, ao monge com maior habilidade artística. Talvez designasse as atividades de revisão a um monge mais idoso e sábio, mas com mãos trêmulas.

De fato, encontram-se exemplos desta organização de trabalho em todos os períodos da história. Supervisores designando tarefas baseados em critérios como habilidades, agilidade, qualidade ou experiência para vários recursos. Com o passar do tempo, estes recursos deixaram de ser apenas pessoas para incluir ferramentas, como por exemplo: máquinas de escrever, máquinas de calcular e formulários impressos. Algumas das ferramentas possibilitaram a automação de parte das tarefas, especialmente no início da década de 50, com o surgimento dos computadores comerciais. Porém, ainda que parte das tarefas tenha sido automatizada, o gerenciamento do trabalho pouco mudou. Ainda persistia a figura do supervisor designando as tarefas e monitorando seu andamento. Atividades eram

passadas de área a área, de pessoa a pessoa, seguindo regras ou orientações dos supervisores. Conforme crescia a complexidade, surgiam mecanismos para rastrear estas atividades – tais como listas, formulários e protocolos – o que permitia medir a produtividade e localizar trabalhos extraviados. Ainda hoje, várias empresas e órgãos públicos fazem gestão de algumas de suas atividades desta forma.

Mas, nos últimos 20 anos, com a evolução das ferramentas de *groupware,* surgiram mecanismos para automatizar o gerenciamento e distribuição das tarefas. Foram os precursores dos sistemas de *workflow* modernos. Definidos formalmente como sistemas computacionais, estas ferramentas eram programas de computador que designavam tarefas, encaminhava-as para execução e monitorava seu progresso.

Contudo, ainda não há consenso sobre o que é de fato Workflow (GEORGAKOPOULOS *et al*, 1995), ou quais funcionalidades um sistema de gerência de workflow deva possuir. O termo workflow é aplicado frequentemente para designar processos de negócios, especificação de um processo, sistemas que executam e automatizam processos ou aplicações que simplesmente suportam a colaboração e coordenação de pessoas que executam um processo.

Uma definição de Workflow é apresentada pela Coalizão para Especificações de Gerenciamento de Workflow (WfMC[5]), uma entidade criada em 1993 por cerca de 90 empresas (entre elas: HP, IBM, Microsoft, Oracle, SAP, Siemens e Xerox) que tem por objetivo o desenvolvimento de padrões para tecnologia de workflow: *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* (WFMC, 2004).

2.2 Workflows e Processos de Negócios

Apesar da definição de workflow ser tradicionalmente ligada a processos de negócios em termos de escritório, como movimentação de documentos, processamento de pedidos ou emissão de faturas, seus princípios e ferramentas podem ser aplicados a diversas outras atividades onde a coordenação do trabalho seja exigida (PLESUMS, 2002). A distribuição das atividades pode ser feita automaticamente para pessoas, equipamentos ou sistemas

[5] WfMC - *Workflow Management Coalition Specification.*

computacionais. Workflows são ferramentas poderosas para a organização de tarefas complexas, pode ser usado desde o sequenciamento de DNA (MEIDANIS, 1996) até pesquisas científicas para comprovação de princípios da teoria da relatividade – como a detecção de ondas gravitacionais (GIL *et al*, 2004).

As empresas ainda são as que mais se beneficiam das ferramentas de reengenharia e gerenciamento de processos. Tais organizações estão inseridas em um meio competitivo e marcado por rápidas mudanças. Para manterem-se competitivas ou mesmo para acompanhar este ritmo, empresas, governos e organizações em geral, estão adotando novos modelos de gestão, reduzindo as estruturas hierárquicas e compensando com uma organização social do trabalho mais flexível e participativo. Um capital cada vez mais valorizado é a informação. Mas, para a informação ter valor ela precisa ser consistente, atualizada e corretamente representada. Esta gestão do conhecimento contempla não somente informações sobre o cliente e mercado, mas sobre os processos internos da própria organização. Mais como uma conseqüência deste cenário, do que como uma resposta consciente, estas organizações estão em constante reengenharia e melhoria de seus processos de negócio. No entanto, a reengenharia dos processos de negócio BPR[6] é um processo de mudança complexo e que afeta diretamente pessoas, sistemas e o fluxo do trabalho.

A aplicação de tais métodos tem por objetivo diminuir o tempo de processos, como processamento de pedidos e entregas, ou diminuir o tempo de desenho e produção de um novo produto, entre outros. A qualidade do atendimento ao cliente é extremamente importante, e não deve incluir apenas os clientes externos, mas os clientes internos também devem ser considerados.

O processo de negócio central de uma organização é aquele que engloba o conjunto de atividades necessárias para a entrega de um produto ou serviço ao cliente, começando com o contato inicial e concluindo com a finalização do contrato, envolvendo faturamento e arrecadação quando houver. Além do processo de negócio central, existem os processos de gerenciamento que dão suporte as atividades administrativas da empresa e que por fim garantem direta ou indiretamente a execução do processo principal.

A forma de representar estes processos e o seu nível de detalhamento varia muito de uma organização para outra. Enquanto em alguns casos os processos são implícitos e informais, em outros estão expressos na forma de um código de boas práticas, ou ainda em

[6] BPR – do ingles *Business Process Re-engineering.*

uma documentação formal de processos. Manter o conhecimento sobre os processos da organização nestes formatos leva ao risco de que fiquem desatualizados. Processos informais aparecem quando o controle é relaxado e caso o mesmo seja ostensivo compromete a flexibilidade e poder de reação sobre eventos inesperados.

2.3 Sistema de Gerenciamento de Workflow

Existe uma diferença importante entre Sistema de Workflow e Sistema de Gerenciamento de Workflow, ou WfMS[7]. O primeiro, refere-se a sistemas que suportam processos específicos, gerenciando os fluxos de trabalho e o encadeamento das atividades para cada instância do processo. Cada instância é um caso particular da execução de um processo. Um sistema de workflow, geralmente envolve um Sistema de Gerenciamento de Workflow, definições dos processos, infra-estrutura como banco de dados, comunicação e aplicativos. O segundo, refere-se à ferramenta básica que prove as características e funcionalidades para a implementação e controle do processo. Assim como sistemas de banco de dados são criados e administrados usando-se um Sistema Gerenciador de Banco de Dados (SGBD), os sistemas de workflow são criados e administrados por WfMS.

Com o uso de WfMS os processos são representados de forma explícita e tem-se a garantia de que o processo documentado corresponde exatamente ao que é executado. Um WfMS deve distribuir automaticamente as atividades através da organização garantindo seu encadeamento, ou seja, sua execução na seqüência correta. Isto é feito com base em entradas como o desenho do processo, que envolve o fluxo de trabalho, os nós de decisão, as exceções, os recursos que serão usados etc. As atividades são distribuídas tanto para pessoas quanto para agentes automatizados, que são geralmente componentes de software.

Pode-se esperar que estas pessoas façam uso de mais de um sistema nas suas atividades diárias. Por exemplo, sistemas de faturamento, sistemas de informações de clientes, de gerenciamento de solicitações, CRM[8], ERP[9] etc. Estes sistemas, em geral, provêem seus próprios mecanismos de controle de processos.

---

[7] *Workflow Management System.*

[8] CRM é a sigla da expressão em inglês *Customer Relationship Management*, e neste contexto refere-se aos sistemas que gerenciam informações dos clientes de forma integrada, com objetivo de atendê-los de forma personalizada.

[9] ERP é a sigla da expressão em inglês *Enterprise Resource Planning*, seu significado vai além da tradução literal, fazendo com que no Brasil sejam conhecidos como "Sistemas Integrados de Gestão Empresarial".

A vantagem de uma solução baseada em WfMS é a representação do processo ser explícita e separada do código do sistema. Isto garante independência dos sistemas envolvidos e que um processo possa envolver mais de um sistema. É mais natural que um sistema de workflow acione estes outros sistemas do que tê-los criando seus próprios mecanismos de gestão de processos ou acionando um sistema de workflow externo. Ou seja, ter um sistema de workflow independente guiando as aplicações, ao invés de ter uma aplicação guiando um sistema de workflow. Como a integração entre sistemas é fundamental, as ferramentas de EAI (*Enterprise Application Integration*) frequentemente estão relacionadas a workflow (PLESUMS, 2002).

Dadas estas características, um WfMS pode ser configurado rapidamente para suportar um novo negócio ou processo, ou ainda ser rapidamente reconfigurado como resposta à alteração de um processo atual.

2.4 Terminologia

A seguir são apresentados alguns conceitos fundamentais para a compreensão dos sistemas de workflow. As definições destes conceitos básicos permitirão um melhor entendimento do tema proposto. (AALST; KEES, 2002; R-MORENO, 2000):

a) **Atividade ou Tarefa:** Uma atividade é uma unidade de trabalho que é executada de uma só vez por um ator. Isto significa que uma tarefa é indivisível e não pode ser interrompida.
b) **Processo:** O conjunto de atividades interligadas e ordenadas para atingir um objetivo específico.
c) **Instância ou caso:** O objetivo principal de um sistema de workflow é tratar casos. Casos são, por exemplo: a reclamação de um cliente, a solicitação de um serviço, a revisão de um artigo etc. Cada caso é identificado unicamente no sistema e possui um tempo de vida limitado. Instanciar um processo é fazer uma cópia deste processo para um caso particular.
d) **Papel ou função:** Define-se um papel com base em um conjunto de atributos pertencentes a um determinado conjunto de atores.

e) **Ator, executor ou agente:** Também chamado recurso, trata-se de uma pessoa ou componente de sistema que executa o trabalho representado por uma instância de atividade de um workflow.

f) **Evento:** Algo que ocorre em um determinado instante de tempo específico. Exemplo: a ocorrência de um sinistro, acidente de carro etc.

g) **Gatilho ou Trigger:** Disparo de uma atividade por um evento. Pode ser visto como uma regra que é avaliada em função da ocorrência de um evento. Disparo significa dar início à execução.

h) **Item de Trabalho:** É a representação de uma atividade, ou seja, trabalho que deve ser realizado por um ator.

i) **Lista de Trabalho:** É uma lista de itens de trabalho associados a um determinado Ator.

j) **Roteamento ou encaminhamento:** Determina quais atividades podem ser executadas e em qual ordem. Como algumas atividades são opcionais, sua execução depende do caso instanciado. A ordem de execução das atividades também pode variar de caso a caso.

2.5 Arquitetura de um WfMS

Um Sistema de Gerenciamento de Workflow (WfMS) é organizado geralmente em camadas de aplicação que seguem o modelo cliente-servidor. Na camada cliente geralmente encontra-se as ferramentas para definição dos processos, interação e administração. Na camada de servidor encontra-se o mecanismo impulsionador do workflow, chamado motor ou máquina de workflow. Este último mecanismo é responsável pelo roteamento do fluxo de trabalho para os diversos agentes do sistema, sejam estes usuários ou componentes de sistemas.

A Figura 1 mostra a visão geral de uma estrutura genérica de um sistema com estas características. São três os elementos principais da estrutura. Primeiro, os componentes do sistema que englobam: uma ferramenta de definição ou modelagem de processos, a máquina de workflow que é o coração do sistema, responsável pela execução do processo e as interfaces com o usuário. O segundo componente refere-se aos dados de controle da aplicação de workflow. O terceiro, e último componente, são as aplicações e dados externos ao sistema

de workflow, mas sob os quais ele atua ou dos quais faz uso para a execução dos processos. Mais detalhes sobre estes componentes serão vistos ao longo deste trabalho.

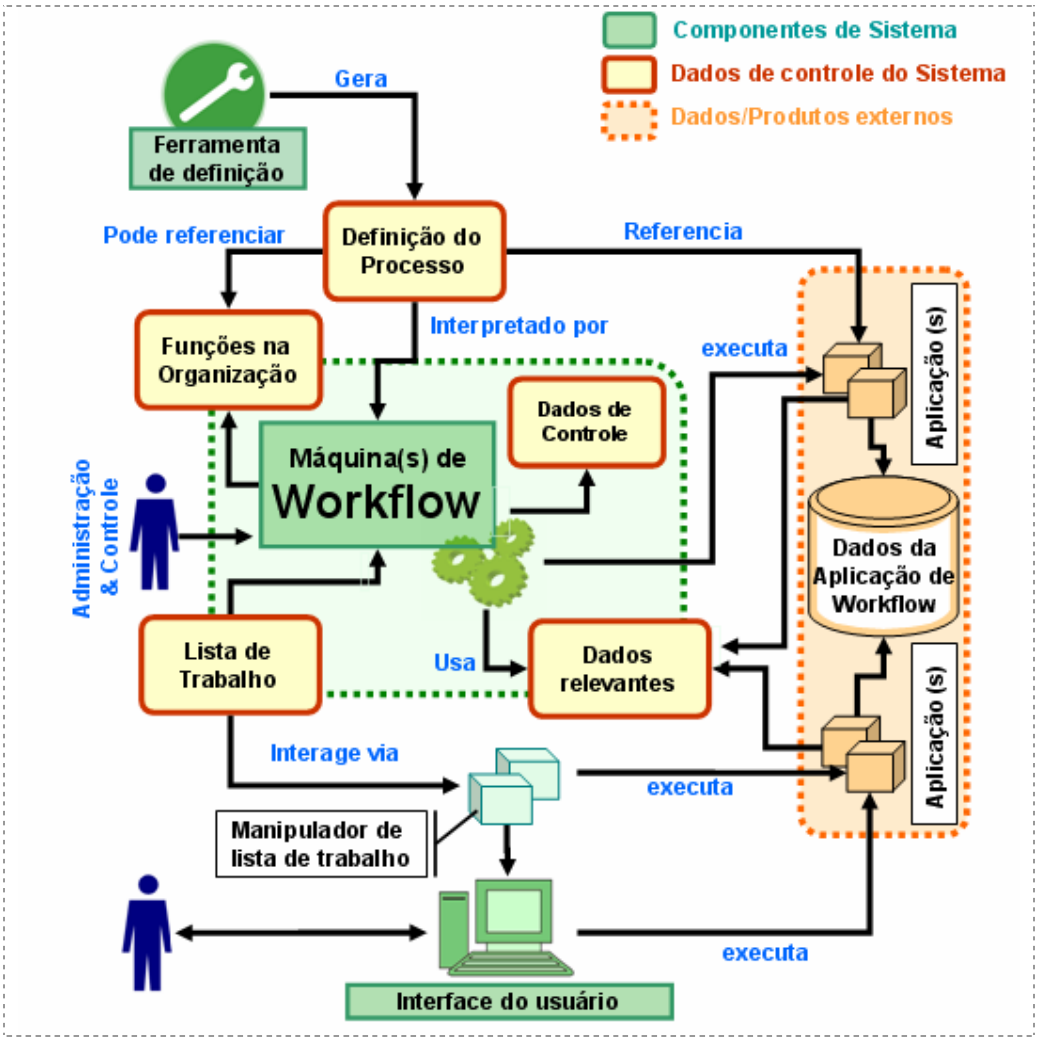


Figura 1: Estrutura genérica de sistemas de workflow (fonte WfMC).

Em 1996, o Departamento de Defesa dos Estados Unidos da América, patrocinou um estudo inicial que foi o ponto de partida do trabalho da WfMC (ALLEN , 2005). O resultado principal deste trabalho foi um Modelo de Referência de Workflow (Figura 2). Este modelo de referência constitui-se em uma arquitetura que é baseada na estrutura genérica anterior, com o objetivo de dar suporte a diversas tecnologias. Define interfaces e protocolos para a comunicação entre as ferramentas existentes, aumentando sua integração.

O ambiente de execução dos processos é isolado dos demais e ao redor deste são definidos vários atributos funcionais padronizados e distribuídos através de um conjunto de interfaces de programação, APIs (*Application Programming Interfaces*). Isto permite que

produtos desenvolvidos independentemente possam interagir adequadamente. Como exemplo, pode-se usar um software distinto para a modelagem do workflow e outro para sua execução e controle.

A seguir será feita uma breve descrição destas interfaces visando esclarecer melhor seu objetivo.

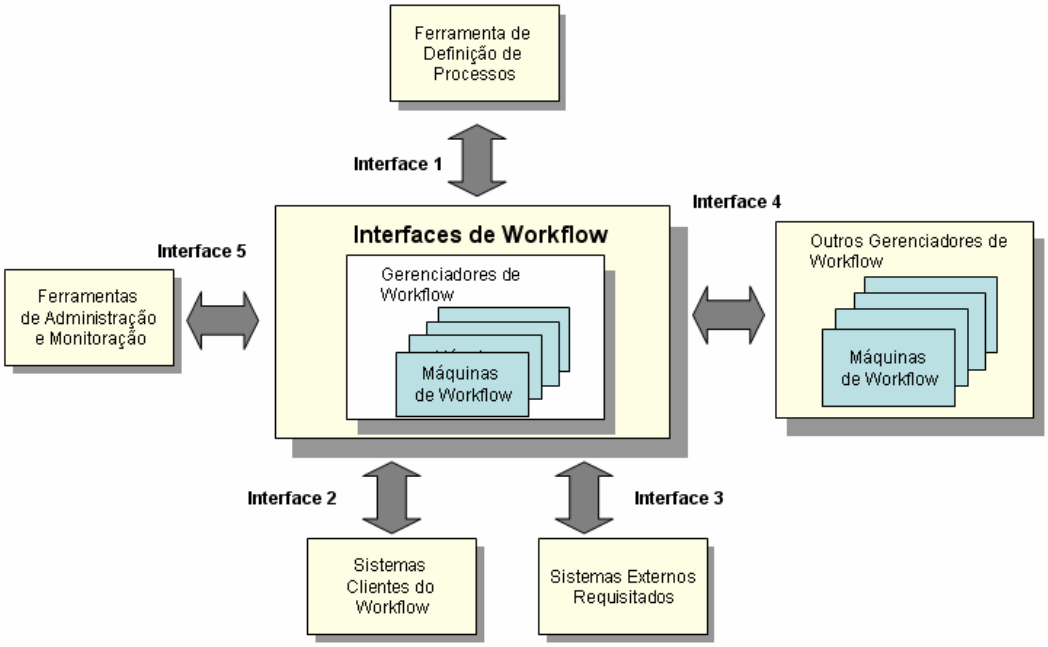


Figura 2: Arquitetura / Modelo de referência (fonte WfMC)

Motor do *workflow*: É o núcleo da arquitetura, o módulo que gerencia e controla todos os componentes existentes, a execução dos processos, a seqüência de execução das atividades, ativação de aplicativos externos. É o módulo responsável pela interligação dos outros módulos, ou seja, também é uma interface entre diferentes módulos da arquitetura de um workflow.

Interface 1 – Definição do Processo: Permite escolher entre diversas ferramentas de modelagem de processos. Definida em função da importação/exportação de descrições de processos. São ferramentas que possibilitam a construção do workflow através da inserção de processos, subprocessos, tarefas etc, possibilitando a verificação de inconsistências nos processos. Algumas ferramentas de definição do processo possuem o recurso de criação gráfica do fluxo.

Sendo uma interface comum para troca de definições, está baseada em uma linguagem padrão para definição de processos. Esta linguagem, conhecida com WPDL (*Workflow Process Definition Language*), é a representação de um processo em um formato que permita

manipulação automática, como modelagem ou execução por um Sistema de Gerenciamento de Workflow (ALLEN, 2005). O processo de definição consiste de uma rede de atividades e seus relacionamentos, critérios que indicam o início e término do processo e informações sobre cada atividade como participantes, aplicativos associados, dados etc.

Esta linguagem é baseada em um conjunto ASCII. Sua gramática é dada na forma EBNF (*Extended Backus Normal Form*), e contempla o uso de chaves para especificação de objetos, atributos, relacionamentos e variáveis para especificar nomes e valores. Mais recentemente surgiu a linguagem de definição de processos XPDL[10], que é baseada na WPDL, mas utilizando-se do formato XML para a descrição dos processos.

Interface 2 - Aplicações Cliente: Projetada para facilitar a integração entre aplicações clientes e diferentes sistemas de worklow, garantindo a portabilidade desta aplicação. Permitir centralizar em uma única aplicação, listas de trabalho provenientes de vários sistemas de workflow, independente dos gerenciadores de workflow utilizados. Isto possibilita combinar na mesma visualização do usuário, ou seja, na sua interface operacional, serviços de sistemas distintos, passando a sensação de ser uma única máquina de workflow.

Interface 3 - Integração com Aplicações Externas: Permitir a integração com ferramentas ou aplicações externas ao sistema de gerenciamento do workflow, possibilitando que possam atuar como agentes dos processos. Um exemplo é a interligação com um sistema de envio de e-mails ou mensagens instantâneas via celular ou *web*.

Interface 4 - Interoperabilidade entre Máquinas de Workflow: Permitir o uso de diferentes produtos de execução de workflow, simultaneamente e coordenados uns com os outros. Possibilita que Sistemas de Workflow heterogêneos interajam entre si.

Interface 5 - Administração e Controle: Permitem o uso de ferramentas distintas para o gerenciamento, controle e auditoria dos sistemas de workflow. Monitora além do fluxo de trabalho, os componentes internos do workflow, e também quais aplicações estão sendo executadas.

A proposta apresentada neste trabalho propõe a extensão destas interfaces criando interações com ferramentas de planejamento no nível de modelagem.

---

[10] XPDL - *XML Processing Description Language*

2.6 Funcionalidades de um Sistema de Workflow

Um Sistema de workflow pode ser analisado do ponto de vista de suas funcionalidades. Neste aspecto tais sistemas cobrem três áreas principais: A modelagem de processos, execução destes processos ou fluxos, e o acompanhamento ou gerenciamento.

2.6.1 Modelagem de Processos

Apesar das várias metodologias utilizadas para o levantamento de informações do processo e sua modelagem, como por exemplo, os métodos *participativo* e *analítico*, entre outros (REIJERS, 2003), existe uma fraqueza natural relacionada à forma de codificar estas informações em um modelo formal. Apesar do auxílio das ferramentas de desenho de processos atuais, ainda cabe ao desenhista do modelo, a maior parte das decisões de ordenação das atividades para obtenção do processo completo, e do controle dos nós de decisão e de suas ramificações. Segundo Aalst (2003), este não é um processo trivial, pois exige um conhecimento profundo do negócio que será modelado e da linguagem para codificação destes modelos. Ele propõe um método de levantamento de processos a partir de uma base de dados, chamado de *mineração de processos*[11]. Ainda assim, este método irá apenas guiar a construção do modelo do processo. Então, ainda persiste a demanda por automatizar, com base no levantamento das atividades, a sua melhor seqüência e contemplar todos os *nós* de decisão importantes.

Para modelar um processo de trabalho inicia-se pela sua tradução do mundo real para o ambiente computacional, gerando uma formalização através de técnicas de modelagem apropriadas. Antes de obter um processo, precisa-se primeiro compreendê-lo (GEORGAKOPOULOS *et al*, 1995). Isto é feito, normalmente, por meio de entrevistas com especialistas que possuem conhecimento do processo. É recomendado o uso de uma metodologia para conduzir estas entrevistas, como as utilizadas por analistas de sistemas. Quando as informações sobre o processo são obtidas é possível especificar o modelo de workflow para representar o processo. O resultado é um modelo ou representação do processo

[11] *Workfow mining*.

a ser executado. Assim a modelagem tem como objetivo "produzir uma abstração de um processo (modelo) que serve como base para especificar um workflow" (MYERS; BERRY, 1998).

Um modelo de processo é uma descrição que contenha todos os dados necessários sobre os processos a serem executados pelo sistema de workflow. Este modelo deve conter todas as informações relevantes ao processo, como: condições de início e fim, regras de execução, usuários ou agentes encarregados, informações ou documentos a serem manipuladas, interações com aplicações externas etc. Ou seja, um modelo de workflow é uma representação gráfica ou textual de um conjunto de atividades e o relacionamento existente entre estas, que responde as questões de quando, como e onde cada atividade é executada e por quem (ARAUJO; BORGES, 2001).

Os componentes fundamentais de um processo são as atividades que devem ser executadas para atingir um determinado objetivo do processo. Tais atividades são realizadas por papéis associados à atividade. Os atores podem assumir estes papéis durante a execução do processo para desempenhar cada atividade. Atores podem ser pessoas ou agentes automatizados. Em cada atividade, informações são manipuladas para sua execução. A Figura 3 mostra o relacionamentos destas entidades, constituindo-se no modelo de processos.

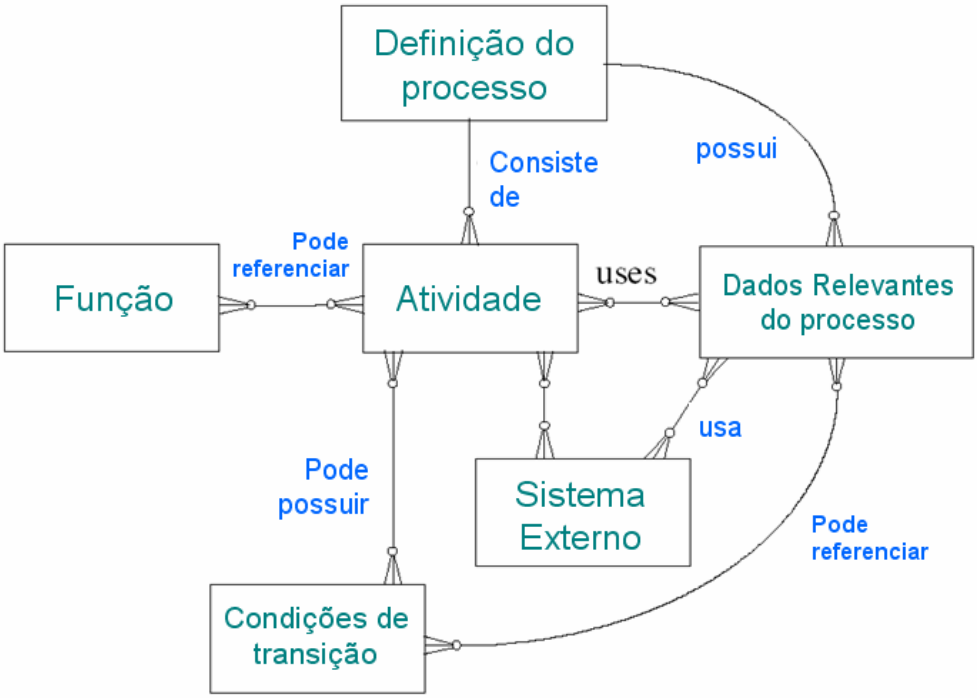


Figura 3: Definição básica de processo (fonte WfMC).

Atividades: O Elemento principal em um fluxo de trabalho são as atividades. Elas devem ser completadas para a conclusão do objetivo do processo. Para uma atividade ser realizada um ator deve assumir uma função, ou papel, relacionado com esta atividade. Um ator pode ser uma pessoa ou uma aplicação, ou agente informatizado. Por exemplo: a atividade autorizar a liberação de um empréstimo, pode ser feita por um *gerente de conta*. A função gerente de conta é um papel que determinada pessoa na organização assume para realização de suas atividades.

Executores: Para que uma atividade seja executada um ator deve ser associado a esta atividade. Esta associação deveria ocorrer durante a fase de modelagem do processo. Mas, nomear atores para cada atividade nesta fase pode não ser o ideal. Como a rotatividade pode ser grande, isto gera muito trabalho de manutenção da definição do processo. A melhor estratégia é associar funções às atividades, e durante a execução do workflow, associar os atores que podem assumir tais funções ou papéis. As funções representam um conjunto de atributos que indivíduos podem assumir. Além de indivíduos, sistemas computacionais, ou seus componentes, podem ser associados às atividades. Por exemplo, o processo de análise de crédito que caberia à figura de um *Analista de Crédito*, pode ser otimizado passando todas as regras para um sistema especialista, que por sua vez proveria uma interface com o sistema de workflow, seguindo os padrões de conectividade definidos.

Encaminhamento: As atividades em um fluxo de trabalho podem ser encadeadas de três diferentes formas: *seqüencialmente*, em *paralelo* ou *condicionalmente*. Seqüencialmente significa que tão logo uma atividade seja executada a atividade seguinte é ativada. A atividade subseqüente não pode ser iniciada até que a atividade atual seja concluída. Atividades executando em paralelo são ativadas simultaneamente e encaminhadas para seus respectivos responsáveis. Não necessariamente irão ser concluídas no mesmo tempo, já que podem seguir critérios diferentes e ou demandar operações distintas. Em um dado momento estes fluxos paralelos irão convergir em um fluxo seqüencial ou na finalização do processo. O encadeamento condicional surge quando o encaminhamento das atividades irá basear-se em uma decisão. Esta decisão será tomada com base nas informações constantes do processo e sua regra deve ser elaborada durante a modelagem do processo.

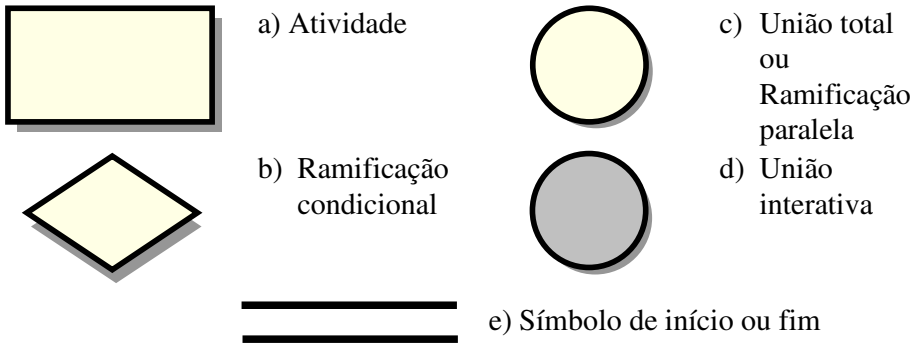


Figura 4: Itens da representação gráfica de workflow

Existem vários padrões para representação gráfica de workflow. Um dos mais citados é o modelo de Casati *et al* (1996), representado na Figura 4. Este padrão possibilita a representação das atividades e o encadeamento entre as mesmas. Os tipos de encadeamento estão representados na Figura 5.

Além dos encadeamentos seqüencial, paralelo e condicional, surge um novo tipo de encadeamento que é dado pela união interativa (CASATI, 1996). A união interativa pode aparecer após uma ramificação paralela, onde são criadas mais de uma linha de execução simultânea. Quando o paralelismo converge novamente para um fluxo seqüencial único, a condição de convergência pode ser:

a) que todas as atividades predecessoras, de cada linha de execução, sejam concluídas;
b) que pelo menos uma das atividades predecessoras de uma linha de execução seja concluída.

Para a situação (a) utiliza-se o símbolo de *união total*, e no caso da situação (b) emprega-se o símbolo de *união interativa*.

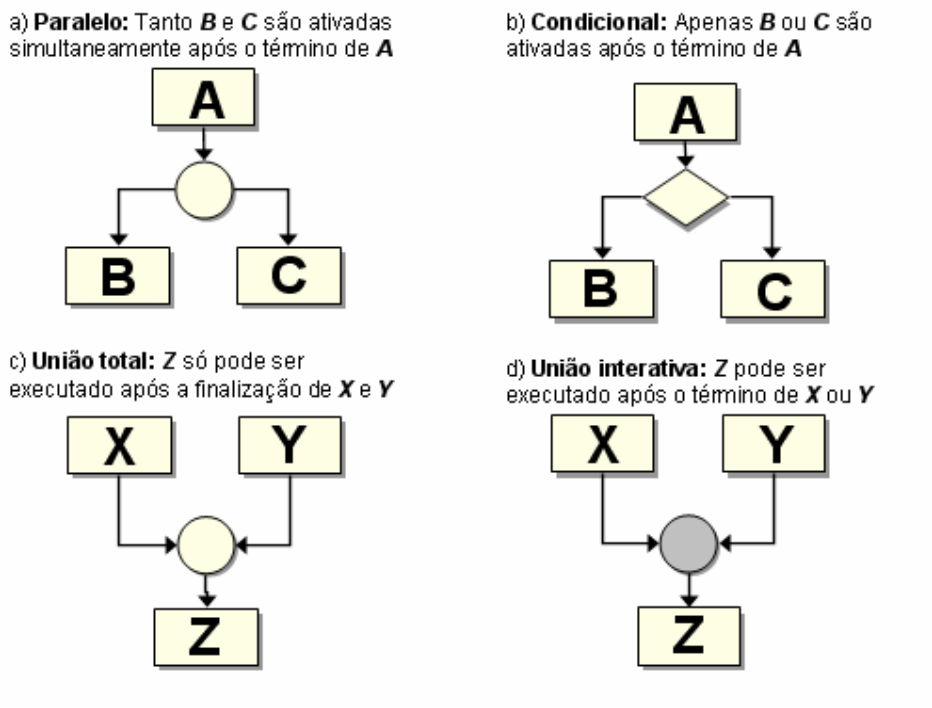


Figura 5: Tipos de encadeamento de atividades.

2.6.2 Execução

Uma vez definido e adequadamente modelado no sistema de workflow, o processo encontra-se pronto para execução. A execução de um processo dá-se pela instanciação de sua definição. Instanciar é colocar em execução a definição do processo para casos particulares. Como exemplo: Tendo-se a definição do processo de análise de crédito, para cada crédito solicitado uma nova instância do processo será criada e executada. O Sistema de Workflow deve ser capaz de executar várias instâncias de um processo ao mesmo tempo, ou ainda instâncias de processos diferentes. Cada instância ativa deve ter sua execução acompanhada pelo sistema e, seguindo o processo definido, encaminhar cada atividade para o ator, ou atores, responsável pela sua execução. Quando concluída a atividade, o ator deve informar ao sistema de workflow sobre sua conclusão. Uma vez concluída a atividade o workflow dará continuidade ao fluxo, gerando uma nova atividade e inserindo na lista de trabalho do ator correspondente ou finalizando o processo.

2.6.3 Acompanhamento e Gerenciamento

Sistemas de workflow devem prover recursos para o monitoramento da execução de uma instância de processo, acompanhando seu *status*. As ferramentas de workflow normalmente disponibilizam estes recursos na forma de relatórios ou telas de consulta. Estas ferramentas podem ter recursos adicionais para geração de estatística sobre a execução de atividades, ou do processo como um todo, analisando informações relativas ao tempo de execução e volumes.

Além dos recursos de acompanhamento descritos, o gerenciamento é facilitado através da ferramenta de administração que possibilita atribuir funções específicas para grupos de usuários ou papéis. Estas funções especiais podem ser privilégios para suspensão ou cancelamento de instâncias, e da própria criação da instância do processo.

2.7 Tipos de Workflow

Os *workflows* podem ser classificados de diversas formas. Destaca-se a classificação quanto à estruturação dos processos. Esta abordagem propõe três categorias para os diferentes tipos de aplicações baseados no grau de freqüência e fluxos de trabalho que apóiam (CHAFFEY, 1998; MARSHAK, 1995; GEORGAKOPOULOS *et al*, 1995).

Ad-hoc[12]: São os que possuem um fluxo de trabalho pouco estruturado. As tarefas e o fluxo de interação entre elas são normalmente imprevisíveis ou desconhecidas até o momento da execução. Não há um padrão predeterminado de movimentação das atividades entre pessoas. A ordenação e a coordenação de tarefas em um workflow do tipo *ad-hoc* não são automatizadas, necessitam da coordenação humana ou co-decisão (SCHAEL, 1991). Estes sistemas estão voltados para grupos dinâmicos que executam processos únicos e altamente individualizados, tipicamente envolve pequenos grupos de profissionais que tem a intenção de apoiar pequenas atividades que requerem uma solução rápida. Exemplos destes tipos de processos são aqueles que envolvem a produção de conhecimento como a elaboração de um relatório, revisão de um livro ou artigo, ou a construção de um sistema.

[12] *'Ad hoc' é uma expressão em latim que significa «para isto», «para um fim específico».*

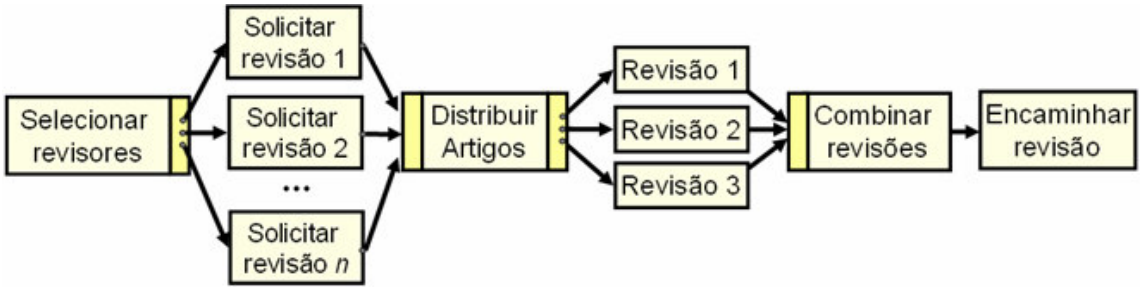


Figura 6: Workflow do tipo a*d-hoc* – revisão de artigos.

A Figura 6 representa um workflow simplificado tipo *ad-hoc* para o processo de conferência de artigos. O processo de revisão constituído pela seleção de revisores, distribuição dos artigos para os revisores selecionados, execução das revisões e colaboração entre os revisores para a produção de uma revisão final (conjunta) e, por fim, envio das revisões para os autores. Caracteriza-se como *workflow* do tipo *ad-hoc* por:

i) Negociação para escolha dos revisores;

ii) Colaboração entre os revisores para produção de uma revisão final combinada;

Administrativos: Estes gerenciam processos com maior grau de estruturação. São processos repetitivos que possuem maior previsibilidade na interação de suas tarefas. Sua ordenação e coordenação podem ser facilmente automatizadas. Geralmente apóiam processos administrativos que podem mudar bastante de uma organização para outra. São concebidos para o roteamento inteligente de formulários através da organização e praticamente todos os indivíduos podem ser seus usuários. Um exemplo pode ser um pedido de compra de materiais, que geralmente é solicitado por uma pessoa uma vez por semana e tramita por diversas áreas que vão gradativamente preenchendo as informações no "documento eletrônico" até a finalização do processo.

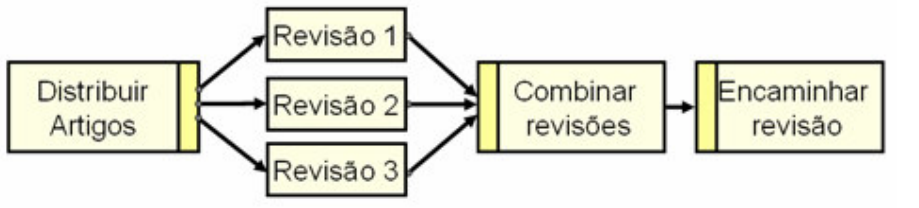


Figura 7: Workflow do tipo Administrativo – revisão de artigos.

Voltando ao exemplo do processo de revisão de artigos, considere que em um novo processo os revisores sejam conhecidos ou estejam selecionados, por exemplo, que os mesmos revisores fazem a revisão de todos os artigos. Considere ainda que os revisores não

colaboram na produção de uma revisão conjunta, apenas produzem revisões individuais que são consideradas pelo editor que toma a decisão final. Sob estas interpretações o workflow de revisão de artigos transforma-se em um workflow do tipo administrativo representado na Figura 7.

Produção: Seguem uma estruturação rígida e bem definida. As regras de interação do processo são previamente levantadas através de uma análise básica do processo. Estes processos são previsíveis e repetem-se com alta freqüência e geralmente possuem uma forte dependência de outros sistemas da organização como banco de dados. Workflows de produção diferem dos administrativos por um processamento de informações complexas envolvendo acesso a múltiplos sistemas. Parte das tarefas é executada por agentes de software e não por atores humanos. Processos de atendimento a clientes em um *Call Center* são um exemplo típico deste tipo de workflow, assim como processos de empréstimo bancário ou controle de sinistros de uma seguradora.

Um exemplo de processo de produção é um processo de vendas. Inicia-se pelo recebimento da solicitação de compra do cliente, seu cadastramento, caso seja um cliente novo. Depois, faz-se a análise de crédito do cliente, reserva-se o produto ou serviço, envia para faturamento e entrega do produto ou execução do serviço. Este processo envolve várias áreas internas e sistemas, como o sistema de cadastro de clientes, sistema de cobrança, faturamento, estoque, logística etc.

Em resumo, sistemas de workflow de produção automatizam processos complexos, estruturados, com pouca mudança de um caso para outro, envolvendo grande volume de transações. Difere dos tipos anteriores por interagir com outros sistemas e pelo uso de tarefas automatizadas.

Existem ainda outras formas de classificação de processos de workflow. A seguir são apresentadas duas destas abordagens.

2.7.1 Classificação Quanto à Abordagem de Comunicação

Contemplando os sistemas de workflow pela forma que provêem a comunicação entre atores e processos, visualizam-se dois mecanismos distintos que podem compreender: o uso de mensagens ou o compartilhamento de documentos. Em sistemas baseados em mensagens, os atores do processo e a máquina de workflow solicitam a realização de atividades através do

envio de mensagens. Uma mensagem pode conter além da solicitação de uma ação, os documentos, dados ou formulários necessários para a realização da tarefa. De forma geral, uma mensagem enviada significa incluir um item na lista de trabalho individual de um ou mais destinatários.

Quando a abordagem é baseada em compartilhamento de documentos, os documentos ou informações manipuladas no decorrer do processo estão armazenados em uma base comum. As listas de trabalho individuais apontam para quais documentos os atores devem utilizar para a realização de uma determinada tarefa e a aplicação de workflow se encarrega de oferecer acesso à base compartilhada (ARAUJO; BORGES, 2001).

2.7.2 Classificação Quanto à Orientação

Após estudar a classificação de workflow quanto à estruturação, Georgakopoulos *et al* (1995), conclui que estas e outras caracterizações não separam a semântica do workflow dos sistemas de gerenciamento de workflows (WfMS) que os suportam, nem da tecnologia e da infra-estrutura que estão envolvidas. Além disso, não se distingue entre os workflows de produção que acessam um pequeno número de sistemas de informação homogêneos; e os workflows altamente automatizados que envolvem muitos sistemas de informações compartilhados. Processos que, dentro de domínio de telecomunicações ou controle de ações militares, estão muito além nestas categorias. Tais processos têm exigências de estrutura e complexidade maiores do que as encontradas em workflows de produção. A quantidade de sistemas heterogêneos associados a estes processos é considerável.

Figura 8: Relacionamento entre tipos de workflow.

Uma forma alternativa é classificar os workflows em uma escala entre: orientado à pessoas e orientado à sistemas (Figura 8).

Orientado a pessoas: Workflows orientados a pessoas dependem da capacidade humana de coordenação das atividades e execução colaborativa das mesmas. Os requerimentos para um sistema de workflow neste cenário são:

a) suporte a colaboração;

b) o controle da consistência e resultados fica por conta dos atores humanos;

Orientado a sistemas: Workflows orientado a sistemas envolvem componentes de software que executam muito processamento computacional ou tarefas especializadas. Enquanto que, na orientação a pessoas a coordenação é feita por atores humanos, na orientação a sistemas este controle deve ser feito por componentes automáticos do sistema de workflow. Deve-se implementar controle de concorrência e técnicas de recuperação para assegurar consistência e segurança das informações. Os principais requerimentos incluem:

a) Combinar as necessidades dos processos e com os dados disponíveis a partir dos sistemas de informação existentes;
b) Interfaces entre os sistemas existentes, que são normalmente heterogêneos, assíncronos e distribuídos;
c) Encontrar componentes adequados de software para que executem tarefas do workflow;
d) Determinar, quais novos requerimentos de software, são necessárias para garantir a automação dos processos de negócios;
e) Assegurar que os sistemas sejam executados de forma correta e confiável

2.8 Considerações Finais

Neste capítulo buscou-se definir os sistemas de Workflows, detalhar suas características e mecanismos de funcionamento. As diversas referências sobre o assunto, na maioria das vezes relacionam workflows a processos de negócio. Não é a única área de aplicação desta ferramenta, mas é a que mais tem se beneficiado da sua existência.

Foi visto que os Sistemas Geradores de Workflow tem a vantagem de permitir uma definição de processo separada do mecanismo de execução. Além disto, é possível definir diversas interfaces que permitem a comunicação com outros sistemas. A modelagem de processos é uma destas interfaces. Porém, modelar processos não é uma tarefa trivial. O desenhista do modelo necessita ser assistido por uma ferramenta computacional que seja eficaz na determinação da seqüência correta de atividades que conduzem ao término do processo. Tais ferramentas, também devem buscar uma solução ótima para obter uma economia de recursos.

Capítulo 3

Planejamento Apoiado em IA

O planejamento em IA envolve determinar um conjunto ordenado de ações que quando executadas por um ou mais agentes a partir de um estado inicial que satisfaça as circunstâncias dadas, resulte num estado final que satisfaça a meta (RUSSELL; NORVIG, 2003). Este capítulo apresenta os conceitos de planejamento e as abordagens para solução dos problemas deste campo.

3.1 Conceitos de Planejamento

Suponha uma situação clássica onde um carro tenha o pneu furado, e necessite fazer a troca do pneu para seguir viagem. Considere ainda, que o motorista não tenha conhecimento das ações que deva executar para obter a troca do pneu, ou qual a sua ordem, e deseja fazê-lo. Esta situação constitui-se então em um problema de planejamento. Neste exemplo, o agente será o próprio motorista, o estado inicial é o carro com um pneu furado e a meta é o pneu substituído pelo estepe. Um plano simples para a solução deste problema seria a seqüência de ações mostradas na Figura 9, onde os círculos mostram o estado inicial e final, ou seja, a situação original que é ter um pneu furado no eixo do carro, e a situação desejada: ter o estepe substituindo este pneu. As caixas são as ações que devem ser executadas para a solução do problema.

O motorista conhece quais ações deve executar.

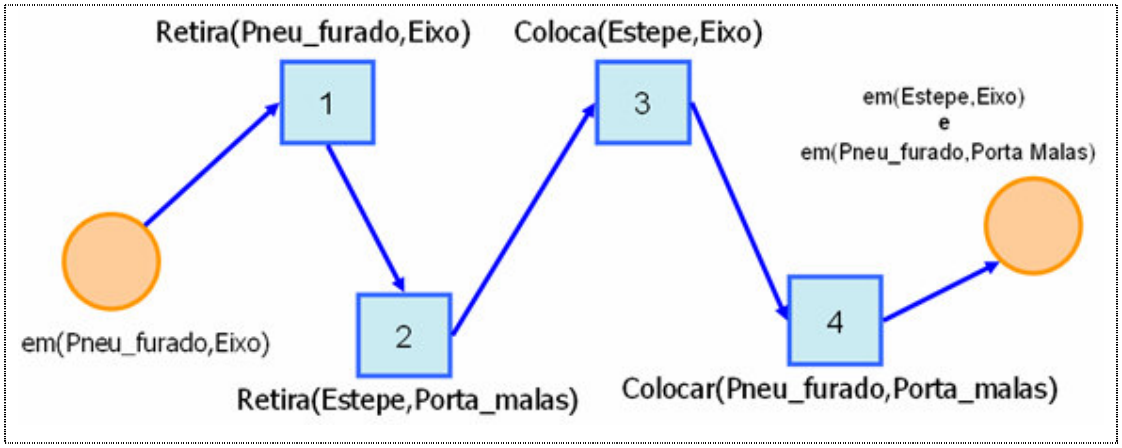


Figura 9: Exemplo de um plano para a solução do problema do pneu furado.

Um problema de planejamento geralmente é representado na forma de uma tripla ($\mathcal{A}$, I, $\mathcal{G}$) – iniciais das palavras em inglês *Actions, Initial State, Goal* – cujos elementos são:

a) Conjunto de ações que podem ser executadas – que representam o domínio do problema;
b) Estado inicial – uma descrição do estado corrente do mundo, sob o qual o agente iniciará suas ações;
c) Estado final, ou meta – uma descrição do objetivo do agente.

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 processo de descobrir uma sucessão de ações que alcançarão uma meta é chamado de planejamento. Pode-se considerar como agentes de planejamento simples os ***resolvedores de problemas*** baseados em busca e os agentes de ***planejamento lógico***, por exemplo os provadores de teoremas.

Quanto aos ambientes de planejamento geralmente são de dois tipos:

a) **Ambientes de Planejamento Clássico:** são aqueles completamente observáveis, determinísticos, finitos, estáticos (mudança só acontece quando o agente age), e discretos (no tempo, ações, objetos, e efeitos).

b) **<u>Ambientes de Planejamento não-Clássico:</u>** são mais complexos, por envolverem ambientes parcialmente observáveis ou estatísticos, e por isto necessitam de um conjunto de algoritmos e agentes desenhados para estas situações;

3.1.1 Mundo dos Blocos

Um dos domínios de planejamento mais famosos é conhecido como o mundo dos blocos. Este domínio consiste em um conjunto de blocos em forma de cubos que se encontram sobre uma mesa. Os blocos podem ser empilhados, mas apenas um bloco pode ficar diretamente em cima de outro. Um braço robótico pode apanhar um bloco e movê-lo para outra posição: ou para mesa ou para cima de outro bloco. O braço pode apanhar apenas um bloco de cada vez, assim não pode apanhar um bloco que tem outro sobre ele. A meta sempre será construir um ou mais empilhamentos de blocos, especificados em termos de quais blocos estarão em cima de quais outros blocos. Por exemplo, uma meta poderia ser colocar um bloco *A* sobre um bloco *B* e o bloco *C* sobre *D*.

```
Ação(Mover(b,x,y),
    PRE-CONDIÇÃO: Sobre(b,x) ∧ Livre(b) ∧ Livre(y)
    EFEITO:Sobre(b,y) ∧ Livre(x) ∧ ¬Sobre(b,x) ∧ ¬Livre(y)
)

Ação(MoverParaMesa(b,x),
    PRE-CONDIÇÃO: Sobre(b,x) ∧ Livre(b)
    EFEITO:Sobre(b,Mesa) ∧ Livre(x) ∧ ¬Sobre(b,x)
)
```

Figura 10: Operadores no domínio do mundo dos blocos.

O predicado ***Sobre(b,x)*** indica quando um bloco ***b*** está sobre ***x***, sendo ***x*** um bloco ou a mesa. A ação de mover um bloco do topo de ***x*** para o topo de ***y*** é descrita por ***Mover(b,x,y)***, como mostra a Figura 10. Para que um bloco seja movido, é necessário que nenhum outro esteja sobre ele. Isto pode ser representado adicionando-se um predicado ***Livre(x)***, o qual é interpretado como *verdadeiro* se houver espaço sobre ***x***, de tal forma que possa ser colocado outro bloco. Isto garante que, caso ***x*** seja a mesa, a interpretação de ***Livre(Mesa)***, será sempre verdadeira. Como um bloco tem espaço para apenas um bloco sobre ele, então o predicado ***Livre(x)*** também funcionará quando ***x*** for um bloco.

Contudo, a ação ***Mover(b,x,y)*** tem um inconveniente. No caso, onde ***y*** representa a mesa, um dos efeitos da ação seria que a mesa não ficaria livre para receber outro bloco ***¬Livre(Mesa)***. Mas, o fato de mover um bloco para a mesa não impede que outro seja movido para ela. Assim, uma nova ação ***MoverParaMesa*** é definida, a qual estabelece que o destino da movimentação é sempre a mesa.

A definição desta nova ação não impede que o planejador use a ação ***Mover(b,x,Mesa)*** no lugar de ***MoverParaMesa(b,x)***. Embora, isto não cause a geração de planos incorretos, pode produzir uma busca desnecessária no espaço de estados. Tal situação pode ser contornada incluindo-se um predicado ***Bloco(b)*** que é interpretado com *verdadeiro* se ***b*** é um bloco, e adicionando-se a condição ***Bloco(b) ∧ Bloco(y)*** à pré-condição da ação ***Mover***.

Uma última observação é feita sob o resultado de uma ação ***Mover(A,B,B)***. Embora esta ação não represente uma operação efetiva no estado atual, não produz um resultado válido, pois seus efeitos são contraditórios. Na maioria das vezes, tais problemas são ignorados já que raramente provocam a geração de planos incorretos. Uma alteração simples elimina de uma vez por todas este risco, bastando inserir na pré-condição uma condição tal que ***x ≠ y***, como pode ser analisado na Figura 11.

```
Inicio(
    Sobre(A,Mesa) ∧ Sobre(B,Mesa) ∧ Sobre(C,Mesa) ∧
    Bloco(A) ∧ Bloco(B) ∧ Bloco(C) ∧
    Livre(A) ∧ Livre(B) ∧ Livre(C)
)

Meta(
    Sobre(A,B) ∧ Sobre(B,C)
)

Ação(Mover(b,x,y),
    PRE-CONDIÇÃO: Sobre(b,x) ∧ Livre(b) ∧ Livre(y)
          ∧ Bloco(b) ∧ (b ≠ x) ∧ (b ≠ y) ∧ (x ≠ y)
    EFEITO:Sobre(b,y) ∧ Livre(x) ∧ ¬Sobre(b,x) ∧ ¬Livre(y)
)

Ação(MoverParaMesa(b,x),
    PRE-CONDIÇÃO: Sobre(b,x) ∧ Livre(b) ∧ (b ≠ y)
    EFEITO:Sobre(b,Mesa) ∧ Livre(x) ∧ ¬Sobre(b,x)
)
```

Figura 11: Problema do mundo dos blocos para empilhar três blocos (A / B / C).

A solução do problema representado anteriormente é dada pela seqüência de ações:

1: Mover(B,Mesa,C)

2: Mover(A,Mesa,B)

3.1.2 A Linguagem dos Problemas de Planejamento

As considerações feitas sobre as dificuldades de se resolver problemas de planejamento sugere que a representação de problemas – estados, ações e metas – deveria ser de tal forma que os algoritmos tirem vantagem da estrutura lógica do problema. A chave é achar uma linguagem expressiva o suficiente para descrever uma grande variedade de problemas, mas restritiva o bastante para permitir que algoritmos eficientes trabalhem com ela.

3.1.2.1 Linguagem STRIPS

Criada no início dos anos 70, a linguagem STRIPS[13] foi inicialmente utilizada em agentes robóticos e é considerada *clássica* em planejamento.

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. São permitidos literais proposicionais, como por exemplo,

$$pneu_furado \wedge estepe_cheio \qquad (3.1)$$

que poderia representar o estado inicial do problema do pneu furado. Também é permitido o uso de literais da lógica de primeira-ordem, como por exemplo,

$$Em(Aviao_1,Rio) \wedge Em(Avião_2,São_Paulo) \qquad (3.2)$$

Que poderia representar um estado no problema de entrega de pacote. No caso de literais da lógica de primeira-ordem estes devem ser aterrados e não podem incluir funções. Literais como,

$$Em(\underline{x, y}) \qquad (3.3)$$

$$Em(Avião(1),São_Paulo) \qquad (3.4)$$

[13] STRIPS – ST*anford* R*esearch* I*nstitute* P*roblem* S*olver*

não são permitidos. A hipótese de *mundo-fechado* (REITER, 1978) é considerada para tratar o *problema do quadro* (McCARTHY; HAYES, 1969) e significa que qualquer condição que não seja mencionada em um estado é assumida como falsa. Assim, em um estado inicial que se tenha:

$$Em(Avião_1, São_Paulo) \wedge \neg Em(Avião_1, Rio) \qquad (3.5)$$

O literal $\neg Em(Avião_1, Rio)$ pode ser suprimido da descrição.

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 aterrados e positivos, por exemplo,

$$Em(Pacote_1, São_Paulo) \qquad (3.6)$$

Um estado s satisfaz uma meta g se s contém todos os átomos de g (mesmo que s tenha outros literais), isto é, $g \subseteq s$. Considerando o exemplo da meta anterior, o estado do mundo a seguir satisfaz a meta:

$$Em(Avião_1, São_Paulo) \wedge Em(Pacote_1, São_Paulo) \qquad (3.7)$$

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 seria:

$$\begin{aligned} &Ação(Voar(p,de,para), && (3.8)\\ &\quad \textsc{Pré-condição:} && Em(p,de) \wedge Avião(p) \wedge Aeroporto(de) \wedge Aeroporto(para)\\ &\quad \textsc{Efeito:} && \neg Em(p,de) \wedge Em(p,para)\\ &) \end{aligned}$$

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:

a) **Descrição da ação:** Nome da ação e lista de parâmetros;

b) **Pré-condição:** uma conjunção de literais positivos, e que não contém funções, os quais devem ser verdadeiros antes que a ação seja aplicada. Qualquer variável que

aparece nas pré-condições deve fazer parte da lista de parâmetros do nome da função;

c) **Efeito[14]:** É uma conjunção de literais, sem funções, que descrevem como o estado é modificado quando a ação é executada. Literais positivos são reconhecidos como *verdadeiros* no estado resultante da ação. Enquanto, literais negativos são reconhecidos como *falsos*. Qualquer variável que apareça aqui também deve fazer parte da lista de parâmetros. Assume-se que para todo literal que não é mencionado na lista de efeitos, permanece imutável. Desta forma, contorna-se o *problema do quadro* (McCARTHY; HAYES, 1969).

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 linguagem STRIPS eram aceitas na esperança de produzir algoritmos de planejamento mais simples e mais eficientes, sem tornar difícil descrever 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 $10 \times 5 \times 5 = 250$ ações puramente proposicionais.

3.1.2.2 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 sua sucessora mais importante tenha sido a Linguagem de Descrição de Ações ADL[15] (PEDNAULT, 1986 *apud* PENBERTHY; WELD, 1992, p.105). Em ADL, a ação de ***Voar*** poderia ser escrita como:

---

[14] 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 que continha os literais positivos, enquanto que a segunda era a lista de remoção contendo os literais negativos.

[15] ADL - *Action Description Language.*

Ação(**Voar(p: Avião, de: Aeroporto, para: Aeroporto,** **(3.9)**
PRÉ-CONDIÇÃO: **Em(p,de) ∧ (de ≠ para)**
EFEITO: **¬Em(p,para) ∧ Em(p,para)**
)

A anotação ***p:Avião*** na lista de parâmetro é uma abreviação para ***Avião(p)*** na pré-condição. Embora não adicione nenhum poder expressivo, torna mais fácil a leitura e também menor o número de possíveis ações proposicionais que podem ser construídas. A pré-condição ***(de ≠ para)*** expressa o fato que um vôo não pode ser feito de um aeroporto para ele mesmo. Isto não pode ser expresso de forma sucinta em STRIPS.

Outro exemplo da diferença entre as linguagens é referente ao problema do mundo dos blocos (seção 3.1.1). Naquele domínio, para que um bloco seja movido, é necessário que nenhum outro esteja sobre ele, isto pode ser representado em lógica de primeira ordem por $\forall x$ $\neg Sobre(x,b)$ ou $\neg\forall\exists\, Sobre(x,b)$. Esta construção pode ser uma pré-condição de uma ação em ADL, mas não pode ser representada em linguagem STRIPS.

A Tabela 1 mostra uma comparação entre as linguagens STRIPS e ADL (RUSSELL; NORVIG, 2003):

Tabela 1 - Comparação entre as linguagens STRIPS e ADL

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**).

A possibilidade de representar *efeitos condicionais* permite a redução na quantidade de ações instanciadas para resolver um problema, visto que uma mesma ação pode ser utilizada

para diferentes situações. Os *efeitos condicionais* que seguem a condição só têm efeito se esta condição for verdadeira no estado sobre o qual a ação se aplica.

3.1.2.3 PDDL

Os vários formalismos de planejamento usados em AI foram sistematizados dentro de uma sintaxe padrão chamada de Linguagem de Definição do Domínio de Planejamento, ou PDDL (*Planning Domain Definition Language*), criada por McDermott (1998) 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 aos pesquisadores trocar *benchmarks*[16] de problemas e comparar resultados de desempenho dos planejadores durante competições. Desde sua criação, a linguagem PDDL vem firmando-se como a referência para a maioria dos planejadores.

Em 2002 surgiu uma extensão importante, a PDDL 2.1 (FOX; LONG, 2003). Suas 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, aparecem modificações no tratamento de expressões numéricas e permite especificar uma função de otimização do plano (denominada métrica), a ser 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. Condições numéricas escritas como comparações entre pares de expressões numéricas, sendo que os efeitos de uma ação permitem modificar os valores das expressões numéricas.
- Nível 3: Introduz o uso de ações temporais[17] 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,

[16] *benchmark* - teste de desempenho de um sistema.
[17] Traduzido livremente de *durative actions.*

- 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+ (FOX; LONG, 2001) 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 (EDELKAMP; HOFFMAN, 2004) introduz dois novos conceitos para a competição de 2004: predicados derivados[18] e temporização de literais[19]. Os primeiros referem-se à criação de regras que permitem deduzir valores independentes das ações do planejamento. Seguem uma regra 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 a seguir refere-se a uma janela de tempo na qual uma loja estaria aberta.

(:init (às 9 (loja_aberta)) (às 20 (not (loja_aberta)))) ***(3.10)***

3.2 Abordagens de Planejamento

Neste tópico descrevem-se os principais algoritmos de planejamento clássico e abordagens mais recentes.

3.2.1 STRIPS

O planejamento como elemento da área de Inteligência Artificial (IA) surgiu das pesquisas sob busca em espaço de estados, prova de teoremas, teoria de controle, necessidades práticas de robótica, escalonamento, entre outras (RUSSELL; NORVIG, 2003). O sistema STRIPS apresentado por Fikes e Nilsson (1971), é a primeira proposta que ilustra a integração destas influências.

---

[18] Tradução de *derived predicates.*
[19] Tradução de *timed initial literals.*

Tendo sido definido a linguagem STRIPS no item 3.1.2.1 pode-se definir agora a sua semântica.

Ações são aplicáveis em qualquer estado que satisfaça suas pré-condições, caso contrário, a ação não tem nenhum efeito. Para decidir pela aplicação, ou não, de um *operador* que utilize lógica de primeira-ordem em suas pré-condições, é necessária uma substituição θ para as variáveis na pré-condição. Seja o exemplo a seguir um estado corrente descrito por:

$$Em(P_1,RIO) \wedge Em(P_2, BHZ) \wedge Avião(P_1) \wedge Avião(P_2) \wedge Aeroporto(RIO) \wedge Aeroporto(BHZ) \quad (3.11)$$

Este estado satisfaz a pré-condição do *operador* a seguir:

$$\begin{aligned} &Ação(Voar(p,de,para), && (3.12)\\ &\quad \textsc{Pré-condição:} && Em(p,de) \wedge Avião(p) \wedge Aeroporto(de) \wedge Aeroporto(para)\\ &\quad \textsc{Efeito:} && \neg Em(p,de) \wedge Em(p,para)\\ &\quad) \end{aligned}$$

Com as substituições:

$$\{p/P_1, de/RIO, para/BHZ\} \quad (3.13)$$

Assim esta ação é aplicável;

Quando uma ação aplicável é executada, ela transforma a descrição do estado corrente ***s*** 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 passa para:

$$Em(P_1,BHZ) \wedge Em(P_2, BHZ) \wedge Avião(P_1) \wedge Avião(P_2) \wedge Aeroporto((RIO) \wedge Aeroporto((BHZ) \quad (3.14)$$

Observa-se que os literais não mencionados na conjunção da cláusula efeito permanecem inalterados.

O algoritmo do planejador STRIPS trabalha empilhando os literais da meta e aplicando sucessivamente as ações ao estado corrente até que o literal seja tornado válido ou todas as alternativas sejam tentadas. Caracteriza-se assim em uma busca no espaço de estados do problema.

O planejador STRIPS é um planejador *linear*. O conceito de *linearidade* vem da *suposição de linearidade*. Nesta suposição, cada literal da meta é tratado como uma submeta e considerados independentes uns dos outros. 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***" (RUSSELL; NORVIG, 2003). Tal situação é demonstrada com o problema de planejamento no mundo dos blocos (Figura 12), onde o estado inicial é dado por:

$$Sobre(C,A) \wedge Sobre(A,Mesa) \wedge Sobre(B,Mesa) \qquad (3.15)$$

E o estado meta por:

$$Sobre(C, Mesa) \wedge Sobre(B,C) \wedge Sobre(A,B) \qquad (3.16)$$

Nenhum dos planejadores lineares baseados em STRIPS poderia resolver este problema simples pois tratavam as submetas isoladamente caindo em um *mínimo local* como:

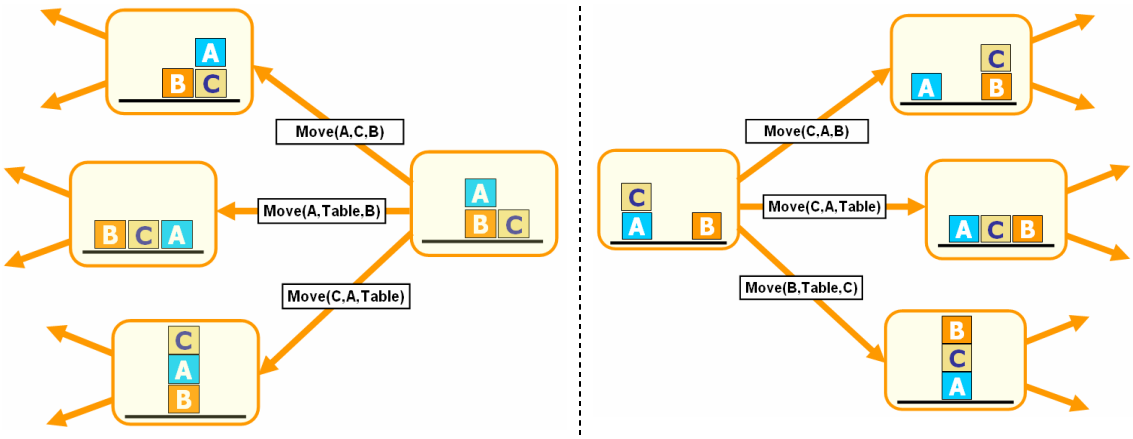
$$Sobre(C, Mesa) \wedge Sobre(A,B)$$

Figura 12: Anomalia de Sussman.

3.2.2 Busca no Espaço de Estados

A abordagem mais direta para a solução de um problema de planejamento é usar busca no espaço de estados. O problema de planejamento pode ser representado como um grafo cujos vértices são os estados e as arestas são as ações. Já que as descrições das ações em um

problema de planejamento especificam tanto pré-condições quanto efeitos, é possível fazer a busca em qualquer direção. Uma busca progressiva parte do estado inicial em direção a meta. Uma busca regressiva corre o sentido contrário, da meta para o estado inicial. Pode-se também utilizar da representação da meta e das ações para produzir automaticamente heurísticas efetivas.



a) Busca regressiva **b) Busca progressiva**

Figura 13: Ilustração das duas abordagens de busca

3.2.2.1 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 conduzem ao estado meta, gerando assim uma seqüência de ações que será a solução do problema. A formulação de problemas de planejamento como problemas de busca no espaço de estados, é dada por:

a) **Estado inicial:** É o estado inicial do problema de planejamento. Em geral, cada estado será um conjunto de literais positivos aterrados; literais que não figuram no estado inicial são interpretados como *falso*.

b) **Ações:** Todas aquelas cujas pré-condições estão satisfeitas no estado, são aplicáveis. O estado sucessor é gerado adicionando-se os literais positivos da cláusula de efeito da ação e removendo os literais de efeito negativo. Basta uma única função sucessora, em conseqüência da representação explícita da ação.

c) **Teste da meta:** Confere se o estado satisfaz a meta do problema de planejamento.

d) **Custo:** O custo de cada ação é normalmente uniforme, igual a 1. Embora seja fácil aplicar custos diferentes para ações diferentes, isto raramente é feito por planejadores STRIPS.

A busca de encadeamento progressivo pode ser muito ineficiente para resolver problemas de planejamento devido a duas características básicas:

- A busca progressiva não distingue ações irrelevantes. Todas as ações aplicáveis em cada estado são consideradas.
- Sem uma boa função heurística a busca torna-se facilmente exponencial. Isto inviabiliza os planejadores em termos de tempo e memória.

3.2.2.2 Busca Regressiva

Nos resolvedores de problemas, a busca regressiva pode ser difícil de implementar quando o estado da meta é descrito por um conjunto de ***restrições,*** ao invés de serem listados explicitamente com 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 13 (a), todas as três ações predecessoras produzem pelo menos um das pré-condições da meta.

Uma busca regressiva que permite ações ***irrelevantes*** ainda será 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 a busca progressiva.

Além de escolher apenas as ações que produzam os literais desejados, tem-se que 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: qual é o conjunto de estados $\boldsymbol{\lambda}$ sob os quais a aplicação de uma ação $\boldsymbol{A}$ produz um estado que contém a meta. Encontrar este conjunto de estados $\boldsymbol{\lambda}$ é chamado ***regressão*** da meta $\boldsymbol{G}$ através da ação $\boldsymbol{A}$. Considerando as definições de *relevância* e *consistência*, pode-se definir o processo geral da ***regressão*** como:

Definição 3.1: *Seja a descrição da meta representada por **G**, tal que **G** é um conjunto de **n** literais na forma* $\{p_1,...p_n\}$. *Seja **A** uma ação relevante e consistente, tal que **A** é uma tupla na forma **(P,E)**, onde **P** é um conjunto que representa os **k** literais conjuntivos da pré-condição da ação **A**; e **E** é um conjunto que representa os **j** literais conjuntivos do efeito da ação **A** . O estado **S** resultante da regressão é chamado de predecessor e é dado por* $\textbf{regressão}(G,A) = (G-(G \cap E_A)) \cup (P_A)$.

Esta definição corresponde à aplicação de duas regras:

- Qualquer efeito positivo de ***A*** que aparece em ***G*** é removido;
- Cada literal da pré-condição de ***A*** é adicionado, a menos que já esteja presente.

```
Função planejamento_regressivo(S, S0) devolve plano
    Entrada:    S -   estado corrente
                S0 -  Estado inicial,
                      Alvo do planejamento regressivo.
{
    Se S0 satisfaz S   //ou seja, se pré-condições
    então              // de S estão em S.
          retorna;
    senão
          A = escolhe_ação(S); //relevante e consistente.
          Se A = ∅
          então
                Falha!
          Senão
                S' = regressão(A,S)
                plano  = planejamento_regressivo (S', S0)
                Retorna = plano∪{A}
          Fim se;
    Fim se;
}
```

Figura 14: Algoritmo de um planejador regressivo.

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. Na Figura 14 é mostrado um algoritmo de um planejador regressivo genérico.

3.2.2.3 Heurísticas para Busca

As buscas regressiva ou progressiva não são eficientes sem uma boa função heurística. A chave para se determinar uma boa função heurística para direcionar o procedimento de busca é encontrar uma função que meça, de maneira rápida e simples, o custo para alcançar a meta a partir de um determinado estado. Ou seja, medir a que distância encontra-se o estado final. No planejador STRIPS, o custo de cada ação é uniforme, ou seja, tem valor *um*. Assim,

a distância é o número de ações. A idéia básica é olhar para os efeitos das ações, e para a meta que deve ser alcançada, e *"adivinhar"* quantas ações são necessárias para satisfazer todas as pré-condições da meta. Achar o número exato é **NP-Difícil** (BYLANDER , 1994) . Isto ocorre, porque a função tem a mesma complexidade inerente a solucionar o problema, pois se o custo de um caminho ótimo é conhecido, a solução para o problema de planejamento também o é. Mas é possível encontrar uma estimativa razoável na maioria das vezes sem muita computação. 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* para achar 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 dá 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:

a) Pode haver duas ações, cada uma das quais remove o literal da meta alcançada pela outra;

b) 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 a 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 $\boldsymbol{A} \wedge \neg \boldsymbol{B}$ no problema original, terá o efeito $\boldsymbol{A}$ 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 com a progressão ou regressão. Até o momento os planejadores progressivos que utilizem 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.

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 (BONET; GEFFNER, 2001) e FF (HOFFMANN; NEBEL, 2001).

Segundo Bonet e Geffner (2001), 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*[20]), 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* (RUSSELL; NORVIG, 2003), 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 mesma 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*, seção 3.2.4.

[20] É comum encontrar referências ao nome em inglês *Hill Climbing*.

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 um 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 (BONET; GEFFNER, 2001).

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. Em lugar de trabalhar separadamente em cada subproblema, têm que tomar decisões sobre como ordenar as ações de todos os subproblemas. Seria preferível uma abordagem que trabalhasse independentemente em 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 e seus vínculos causais. Os vínculos causais permitem aos sistemas de ordem parcial garantir que as submetas já satisfeitas não sejam desfeitas durante o restante do planejamento e do refinamento do plano parcial que está sendo construído. Muitas vezes, esta abordagem é confundida com o planejamento *não-linear*, seção 3.2.1. 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*[21] em relação ao tempo, que é não comprometer nada, até que seja realmente necessário, ou seja, atrasar uma escolha durante uma busca. Um planejamento de ordem do parcial nada mais é do que um sistema que adota o conceito de comprometimento-mínimo com relação ao tempo. Existe ainda o conceito de *comprometimento-mínimo* em relação à instanciação de variáveis,

[21] Também é utilizada a expressão em inglês *least commitment*.

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. No planejamento Progressivo ou Regressivo faz-se uma busca por ações no espaço de estados do mundo, que dependendo do conjunto das ações possíveis pode tornar-se um processo de busca exponencial. Isto, os inviabiliza em termos de consumo de tempo e memória. 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.

Começa-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. As ações nesta busca não são ações no mundo, mas ações nos planos: adicionando um passo para o plano, impondo uma ordem que coloca uma ação antes de outra, e assim por diante. Assim, o estado inicial representa um plano vazio e a meta representa o plano completo, ou 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_Calçado)

Action(Calçar_Meia_Direita,
    EFFECT: Meia_Direita_Calçada)

Action(Calçar_Sapato_Esquerdo,
    PRECOND: Meia_Esquerda_Calçada,
    EFFECT: Sapato_Esquerdo_Calçado)

Action(Calçar_Meia_Esquerda,
    EFFECT: Meia_Esquerda_Calçada)
```

Figura 15: 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. Partindo deste plano nulo o planejador faz escolhas entre as operações que alteram o plano até obter sua meta, conforme definido no parágrafo anterior.

Considere o problema clássico de calçar um par de sapatos. Podemos descrevê-lo como um problema de planejamento formal conforme a Figura 15.

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 estará manipulando as duas subseqüências independentemente, sem se comprometer se uma ação em uma seqüência está antes ou depois de uma ação na outra. A Figura 16 mostra o plano de ordem-parcial que é a solução do problema dos sapatos e meias. 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.

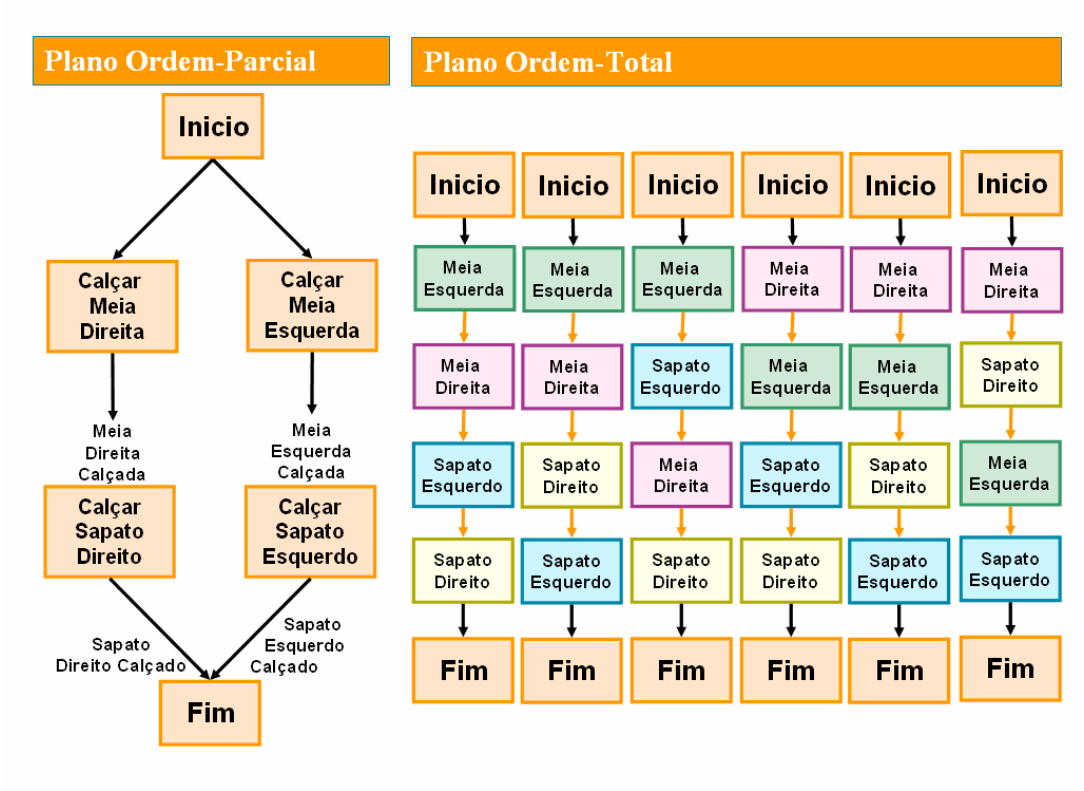


Figura 16:Plano de ordem-parcial e suas linearizações para o problema das meias e sapatos.

O plano final da figura anterior é mostrado a seguir na Figura 17. Por razões práticas não são mostradas as restrições de ordenação que colocam cada ação depois da ação simbólica *início* e antes de *fim*:

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 $\xrightarrow{Meia_Direita_Calçada}$ Calçar_Sapato_Direito,
Calçar_Meia_Esquerda $\xrightarrow{Meia_Esquerda_Calçada}$ Calçar_Sapato_Esquerdo,
Calçar_Sapato_Direito $\xrightarrow{Sapato_Direito_Calçado}$ *fim*,
Calçar_Sapato_Esquerdo $\xrightarrow{Sapato_Esquerdo_Calçado}$ *fim* }

Open Preconditions: *{}*

Figura 17:Estado final do planejamento do 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 pelos elementos a seguir:

- **Conjunto de ações:** Formam o conjunto de ações do problema de planejamento. O plano "vazio" contém somente as ações de ***Início*** e ***Fim***. A ação ***Início*** não tem nenhuma pré-condição e tem como efeito todos os literais no estado inicial do problema de planejamento. A ação ***fim*** não tem nenhum efeito e tem como pré-condições os literais da meta do problema de planejamento.
- **Conjunto de restrições de ordenação**: Cada restrição de ordenação está na forma A $\prec$ B, o qual é lido como "A antes de B" e significa que a ação A deve ser executada em algum momento antes de ação B, mas não necessariamente imediatamente antes de B. As restrições de ordenação devem descrever uma ordem parcial formal. Qualquer ciclo – tal como A $\prec$ B e B $\prec$ A - representa uma contradição. Assim, nenhuma restrição de ordenação pode ser adicionada ao plano se ela gera um ciclo.
- **Um conjunto de vínculos causais**: Um vínculo causal entre duas ações A e B no plano é escrito o como A $\xrightarrow{p}$ B (é lido como "A gera ***p*** por causa de B"). Por exemplo, o vínculo causal:

$$Calçar_Meia_Direita \xrightarrow{Meia_Direita_Calçada} Calçar_Sapato_Direito$$

afirma que *Meia_Direita_Calçada* é um efeito da ação

de *Calçar_Meia_Direita* e uma pré-condição de *Calçar_Sapato_Direito*. Também afirma que *Meia_Direita_Calçada* deve permanecer verdadeiro entre o momento da execução da primeira ação até o momento da execução da segunda. Em outras palavras, o plano pode não ser estendido adicionando uma ação nova **C** que esteja em **conflito** com o vínculo causal. Uma ação **C** **conflita** com $A \xrightarrow{p} B$, se C tem o efeito ¬p e se C pudesse (de acordo com as restrições de ordenação) vir depois de A e antes de B. Alguns autores chamam os **vínculos causais** de intervalos de proteção, porque o vínculo $A \xrightarrow{p} B$ protege ***p*** de ser negado dentro do intervalo de A para B.

- **Um conjunto de pré-condições abertas** Uma pré-condição está aberta se não é atingida por nenhuma ação no plano. O trabalho do planejador é reduzir o conjunto de pré-condições abertas para um conjunto vazio, sem introduzir uma contradição.

Define-se um **plano consistente** como um plano no qual não há nenhum ciclo nas restrições de ordenação e nenhum conflito com os vínculos causais. Um plano consistente sem pré-condições abertas é uma **solução**.

Como foi estabelecido que o Planejamento de Ordem Parcial é uma busca no estado de planos, o algoritmo de busca usado pode ser escolhido livremente. Os detalhes de um algoritmo POP[22] resumem-se então ao plano inicial, à função sucessora e ao teste da meta.

- **Plano Inicial:** Contém *início* e *fim*, a restrição de ordenação *início* $\prec$ *fim*, nenhum vínculo causal e todas as pré-condições de *fim* são pré-condições abertas;

- **Função Sucessor:** Escolhe arbitrariamente uma pré-condição aberta ***p*** em uma ação *B* e gera um *plano sucessor* para todo caminho consistente possível, escolhendo uma ação *A* que atinge ***p*** . A consistência é garantida por:
    - O vínculo causal $A \xrightarrow{p} B$ e a restrição de ordenação $A \prec B$ são adicionados ao plano. A ação ***A*** pode ser uma ação existente no plano ou uma nova. Caso seja nova, é adicionada ao plano e também adiciona-se *início* $\prec A$ e $A \prec$ *fim*.
    - Solucionam-se os conflitos entre o vínculo causal novo e todas as ações existentes e entre a ação *A* (se é nova) e todos os vínculos causais

[22] POP é a sigla para a expressão em inglês *Partial Order-Planning* (Planejamento de Ordem-Parcial)

existentes. Um conflito entre $A \xrightarrow{p} B$ e C é resolvido fazendo com que C aconteça em algum momento fora do intervalo de proteção, ou adicionando $B \prec C$ ou $C \prec A$. Adicionam-se estados sucessores para qualquer um, ou ambos, se eles resultarem em planos consistentes.

- **Teste da meta:** Verifica se um plano é uma solução ao problema de planejamento original. Já que somente planos consistentes são gerados, o teste de meta apenas precisa verificar se não há nenhuma pré-condição na lista de pré-condições abertas.

3.2.4 Planejamento por Grafos

O planejador *GraphPlan* (BLUM; FURST, 1995) representou um marco na área de planejamento por ser um algoritmo simples baseado em grafos e obter soluções para problemas de planejamento de forma muito mais rápida que seus antecessores.

O processo de resolução do *GraphPlan* alterna entre duas fases, a construção de um grafo de planejamento e a extração de uma solução desse grafo. A cada passo o algoritmo executa sucessivamente estas duas fases, até alcançar a solução.

3.2.4.1 Grafo de planos

Um grafo de planejamento consiste numa sucessão de ***níveis*** ou *camadas* de dois tipos: a primeira formada pelas proposições ou conjunto de literais e a segunda por ações (camada par). Estes níveis correspondem aos passos em ordem cronológica do plano onde o nível zero é o estado inicial. Os níveis ímpares (A_i) contém as ações e os níveis pares (S_i) contém os literais, que são pré-condções do nível seguinte ($i+1$) e efeitos do nível anterior ($i-1$). De maneira geral, os literais são todos que ***poderiam*** ser verdadeiros naquele passo cronológico, dependendo das ações executadas nos passos cronológicos anteriores. Da mesma forma, as ações são todas aquelas que ***poderiam*** ter suas pré-condções satisfeitas pelas proposições do nível anterior naquele passo cronológico.

As arestas ligam efeitos de ações de uma camada às proposições correspondentes da camada seguinte. As arestas sempre terão a direção de um nível menor para um nível maior, indicando a ordem de execução das ações em um plano.

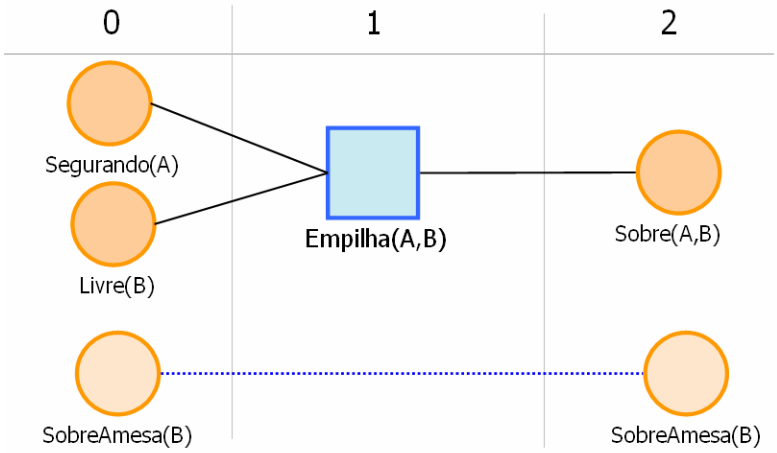


Figura 18: Representação de um grafo de três camadas para um problema do mundo dos blocos.

A Figura 18 mostra um grafo de planos onde os círculos representam os nós de literais e os quadrados os nós de ações. Além das arestas de conexão de níveis (linhas sólidas) que estabelecem a relação entre ações e literais (pré-condição e efeitos), também existe um tipo especial de aresta chamada de arestas de *persistência* (linhas pontilhadas), que conectam dois nós de literais. O grafo de planejamento precisa representar a inércia (falta de ação) tão bem como representa as ações. Isto é, precisa permitir a um literal permanecer verdadeiro de uma situação para outra, caso não seja alterado por nenhuma ação. Em um grafo de planejamento isto é alcançado com um conjunto de **ações de persistência**, representadas pelas *arestas de persistência*. Seu objetivo é manter na camada ***i*** os literais da camada ***i-2***, contornando-se assim o *problema do quadro* (McCARTHY; HAYES, 1969).

3.2.4.2 Expansão do Grafo

Na fase de expansão, adiciona-se ao grafo de planos uma camada de ações que tenham suas pré-condções atendidas na camada anterior, incluindo ações de persistência. Na primeira expansão, as ações cujas pré-condções são atendidas pelo estado inicial são incluídas no grafo formando mais dois níveis, um com as ações e outro com literais que são os efeitos das ações anteriores.

Durante o processo de expansão podem aparecer inconsistências e conflitos. Uma representação consistente de um estado não pode conter dois literais do tipo ***P*** e ***¬P***, pois estes são exclusivos mutuamente. O mesmo princípio aplica-se às ações. Para evitar estes conflitos e inconsistências o *GraphPlan* executa uma análise em busca de relações de exclusão mútua (*mutex*) entre todas as combinações de dois vértices de um mesmo nível do grafo.

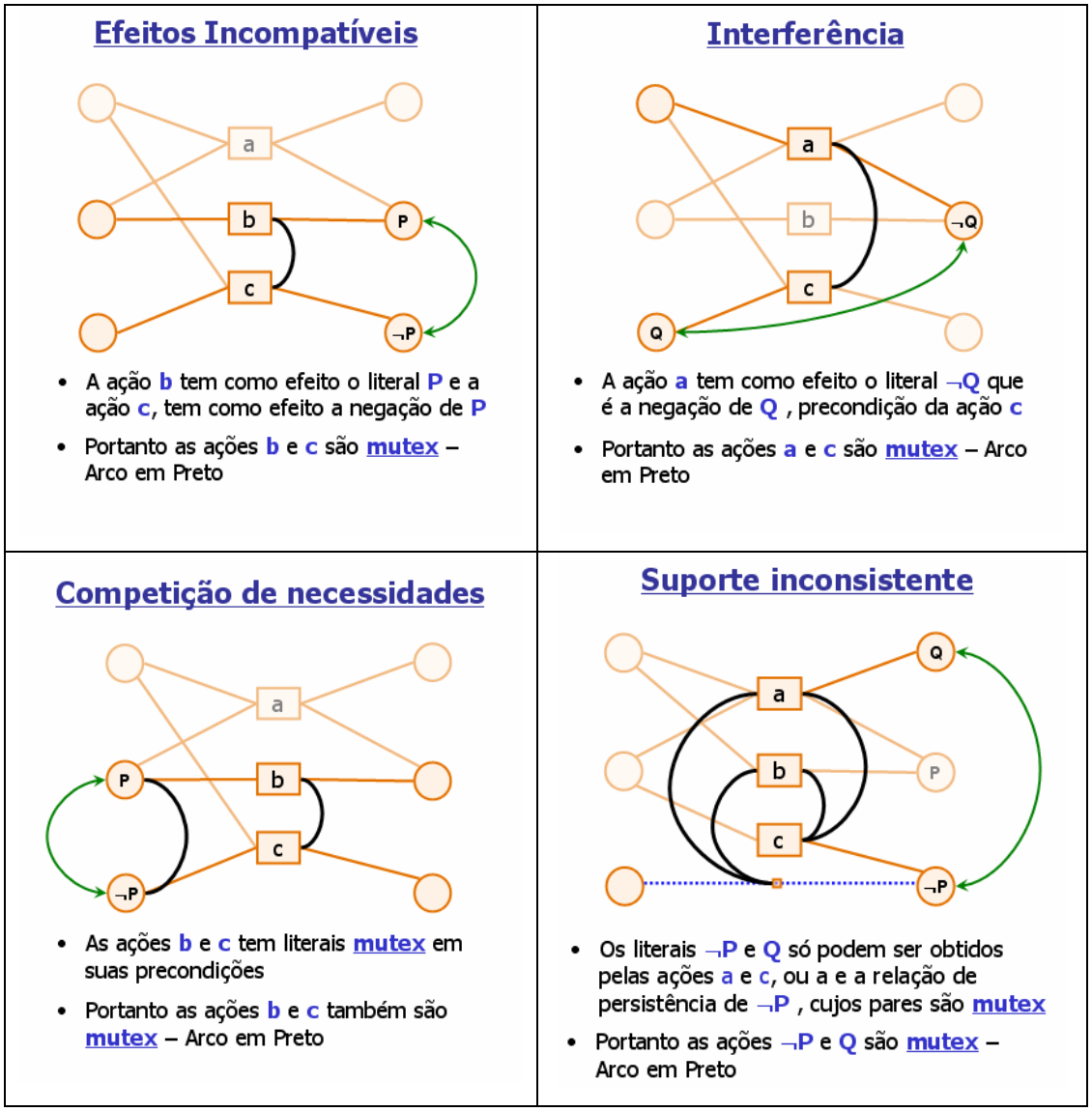


Figura 19: Relações *mutex* em um grafo de planejamento.

As relações de exclusão mútua (vínculos ***mutex***) para ações e literais são definidas na (Figura 19):

- Um par de **ações** pode ser ***mutex*** em um mesmo nível, se há:
  1. **Efeitos Incompatíveis**: Uma ação nega um efeito de outra.
  2. **Interferência**: um dos efeitos de uma ação é a negação de uma pré-condição de outra.
  3. **Competição de necessidades**: uma das pré-condções de uma ação é mutuamente exclusiva com uma pré-condição de outra.
- Dois **literais** podem ser ***mutex*** no mesmo nível se:
  1. Um é a **negação do outro;** ou se existe um,

Suporte inconsistente quando cada par possível de ações que poderiam atingir os dois literais é mutuamente exclusivo.

3.2.4.3 Algoritmo *GraphPlan*

Fica-se Alternando entre o nível de estado S_i e o nível de ação A_i até que seja alcançada uma situação onde, dois níveis consecutivos (de ações e literais) sejam idênticos. Neste momento, diz-se que o grafo terminou o **nivelamento**. Todo nível subseqüente será igual, então qualquer expansão adicional é desnecessária. As seguintes definições são verdadeiras:

1. Todo nível A_i contém todas as ações que são aplicáveis em S_i, junto com as **restrições** que dizem quais pares de ações não podem ser executadas.
2. Todo nível S_i contém todos os literais que poderiam resultar de qualquer possível escolha de ações em A_i-*1*, junto com restrição que dizem quais pares de literais não são possíveis.

```
function GRAPHPLAN(problem) returns solution or failure

  graph    INITIAL-PLANNING-GRAPH(problem)
  goals    GOALS[problem]
  loop do
     if goals all non-mutex in last level of graph then do
        solution    EXTRACT-SOLUTION(graph, goals, LENGTH(graph))
        if solution ≠ failure then return solution
        else if NO-SOLUTION-POSSIBLE(graph) then return failure
     graph    EXPAND-GRAPH(graph, problem)
```

Figura 20: Algoritmo *GraphPlan* (RUSSELL; NORVIG, 2003).

Primeiro, o algoritmo representado na Figura 20, verifica se todos os literais da meta estão presentes no nível atual através da instrução (***if goals ...)*** e também se estão sem vínculos **mutex** entre qualquer par deles. Se este é o caso, então **uma solução pode existir** dentro do grafo atual, assim o algoritmo tenta extrair aquela solução através da função ***Extract-Solution***. Caso contrário, **expande o grafo** adicionando as ações para o nível atual e os literais de estado para o próximo nível através da função ***Expand-Graph*** e repete o processo até encontrar uma solução ou decidir que nenhuma solução é possível.

3.2.5 Planejamento por Satisfabilidade

Até o início da década de 90, pouca atenção era dada à abordagem de planejamento utilizando-se algoritmos para a solução de problemas SAT. Um problema SAT consiste na verificação da satisfabilidade de uma expressão em FNC (*Forma Normal Conjuntiva*) ou FND (*Forma Normal Disjuntiva*), embora a primeira seja a melhor representação. Este descrédito estava baseado no baixo desempenho dos resolvedores de problemas de propósito geral. Isto levou ao desenvolvimento de algoritmos de propósito específicos, como os de planejamento que foram discutidos até agora.

Porém, o aperfeiçoamento dos algoritmos para solução de problemas SAT trouxe para esta área um novo interesse. Em 1992, a apresentação o algoritmo GSat (*Greedy local search procedure*) (SELMAN at al; 1992), motivou Kautz e Selman (1992) a formalizarem o planejamento em termos de satisfabilidade proposicional. Este foi o início do desenvolvimento do algoritmo de planejamento *SATPlan* baseado na solução de um problema SAT.

Apesar da abordagem totalmente diferente dos planejadores clássicos, o planejador SATplan ainda utiliza-se da representação do problema e do domínio na linguagem STRIPS. A estrutura típica de um planejador baseado em SAT é mostrada na Figura 21. Como a solução envolve a transformação do problema em sentenças lógicas, o primeiro passo é usar um *compilador* que recebe como entrada a descrição do problema em STRIPS e gera uma proposição em FNC com as descrições do estado inicial e meta, e das ações possíveis. Como o comprimento do plano (tamanho da solução do problema) não é conhecido *a priori*, o algoritmo supõe um comprimento, gera uma fórmula na lógica proposicional correspondente e tenta satisfazê-la um número finito de vezes. Se a fórmula não for satisfazível, o comprimento é incrementado iterativamente até que uma fórmula deste tipo seja encontrada ou atinja-se um limite de tentativas pré-definido. Caso seja satisfazível, então uma solução foi encontrada e a transformação da representação lógica para a descrição STRIPS é efetuada pelo *decodificador*.

O papel do *simplificador* é de utilizar técnicas rápidas (tempo linear) como a propagação de cláusulas unitárias e eliminação de literais para reduzir o tamanho da fórmula FNC.

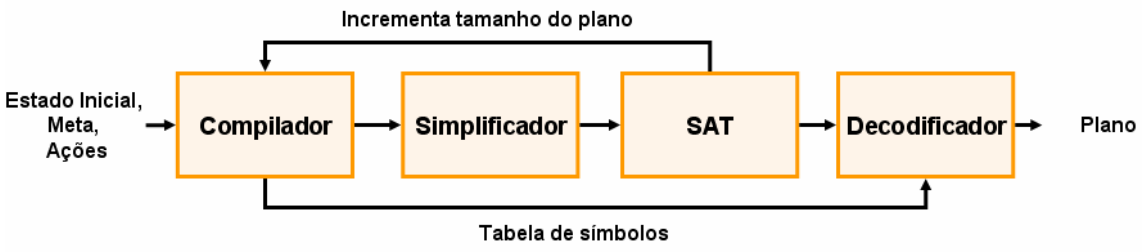


Figura 21: Arquitetura do planejador SATplan.

O Sucessor do *SATplan* foi o planejador *Blackbox* (KAUTZ; SELMAN, 1998), que combinou a abordagem SAT com *Graphplan*, conseguindo melhorar incrivelmente a eficiência. Este planejador é também um sistema muito flexível que permite escolher o algoritmo de resolução, ou mesmo empregar mais de um para resolver o mesmo problema.

3.3 Planejamento em Ambientes Dinâmicos

O planejamento clássico apresentado até agora 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 várias formas de classificar tais abordagens, entre elas destacam-se: monitoramento e replanejamento, planejamento reativo, planejamento contingente e outras.

3.3.1 Monitoramento e Replanejamento

Para lidar com informações incompletas ou incorretas o planejador pode monitorar a execução do plano à procura de alguma falha, ou seja, quando o estado do mundo, depois de uma ação não é o que foi previsto:

A monitoração da execução é o processo de encontrar diferenças entre o estado real do mundo e o estado previsto pelo planejador. Existem muitos motivos para ter-se um processo de monitoração em um sistema de planejamento:

- *Detecção de Falhas internas:* Um processo de monitoração pode detectar comportamentos anômalos na execução causados por falhas de sistemas ou equipamentos.
- *Detecção de Contingências:* Eventos inesperados podem invalidar a execução de um plano. A monitoração pode detectar alterações inesperadas no ambiente, assim como ações cujos efeitos não haviam sido previstos.
- *Detecção de Oportunidades:* Durante a execução de um plano, podem aparecer situações que não invalidam a execução do plano, pelo contrário podem representar vantagens. Tais situações podem significar uma simplificação ou otimização do plano, melhorando assim a sua qualidade.

Quando alguma destas situações é detectada, faz-se necessário um replanejamento para obter uma nova seqüência de atividades que conduzam ao objetivo.

3.3.2 Planejamento Reativo

Em ambientes dinâmicos, não há garantias de que a execução das ações de um plano produza os efeitos previstos, nem mesmo que possam ser executadas quando deveriam. Para um agente as conexões entre ações e seus efeitos são incertas, algumas vezes elas simplesmente falham. Uma vez que, uma atividade pode falhar, um plano não pode ser expresso como uma relação de ordem entre ações. O executor do plano não pode simplesmente selecionar cegamente atividades e submetê-las à execução, ao contrário, deve monitorar sua execução. O planejamento reativo baseia-se em prover uma capacidade ao agente para escolher uma ação com base no estado atual do mundo, utilizando-se de regras pré-definidas. Esta capacidade, por um lado, simplifica o problema de se trabalhar com ambientes dinâmicos ou inacessíveis, mas por outro lado incorre no aumento da complexidade de gerar as regras necessárias para que o agente possa tomar esta decisão durante a execução. O Planejamento reativo é desejável para lidar com situações de emergência, ou quando há pouco tempo disponível para atuar no ambiente.

Várias abordagens surgiram para lidar com estes problemas. Uma das primeiras abordagens reativas que surgiu, foi o *planejamento universal* (SCHOPPERS, 1987), baseado

na geração de uma árvore de decisão binária (BDDs[23]) do conjunto de ações possíveis. Porém, o grande volume de informações geradas para todas as situações possíveis, tornava o seu uso inviável. Até mesmo para pequenos problemas (GINSBERG, 1989). Outra abordagem clássica para o problema é a estratégia de Regras de Controle Situadas (SCRs[24]) (DRUMMOND, 1989). Este método dotava o agente executor de uma capacidade de reação, possibilitando a escolha e execução de atividades sem a existência de um plano completamente estruturado.

As SCRs não definem qual atividade deve ser executada, apenas asseguram que seja escolhida uma atividade, a qual seria a primeira de uma seqüência, que eventualmente atingirá a meta. A escolha e execução de uma atividade podem ser realizadas antes que a síntese da SCR tenha terminado. As SCR são usadas para reduzir a escolha não-determinística de uma atividade no conjunto de atividades possíveis para alcançar a uma dada meta. Pressupõe, portanto um controle não-determinístico pelo agente sob o qual ações são escolhidas.

Quando há tempo suficiente para fazer o planejamento ou replanejamento o seu resultado produz um comportamento dirigido à meta. Mas se a situação permitir, as atividades devem ser escolhidas sem a necessidade de um planejamento completo.

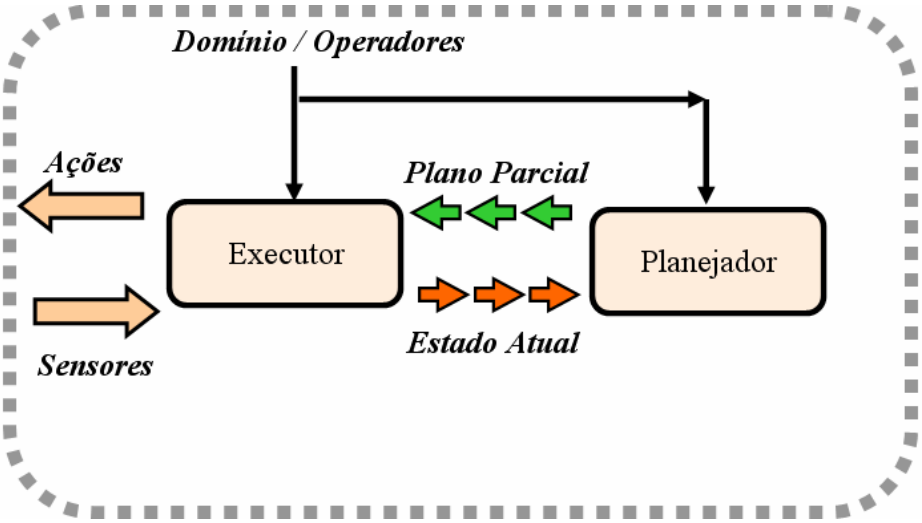


Figura 22: Esquema de um planejador reativo primitivo.

De maneira geral, um agente reativo envolve um módulo de execução, responsável por executar as ações e monitorar seus efeitos, e um módulo de planejamento, que deve ser simplificado para a escolha imediata da ação que melhor conduz em direção à meta (Figura 22).

[23] BDDs - *Binary Decision Diagrams.*
[24] SCRs - *Situated Control Rules*

3.3.3 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*.

3.3.3.1 Planejamento condicional

Nesta abordagem 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 precursor desta abordagem foi o planejador *Warplan*-C (WARREN, 1976). Este planejador considera a possibilidade de uma ação possuir dois efeitos disjuntivos contrários, $\boldsymbol{P}$ e $\neg\boldsymbol{P}$. Inicialmente é montando um plano totalmente ordenado considerando que as ações possuem apenas os efeitos condicionais positivos. Então o planejador é chamado, para cada ramo alternativo possível. O seguimento inicial do plano é mantido, mas a ação condicional é substituída por outra ação, cuja pré-condição é uma condição de proteção, que garante a execução do seguimento inicial do plano. O efeito desta nova ação é o efeito que não foi utilizado na ação original, ou seja, $\neg\boldsymbol{P}$. Então, um novo plano é gerado para este novo ramo. O resultado final é um plano condicional, mas não é feito nenhum tratamento para sintetizar um plano menor, convergindo em seqüências comuns de ações após uma ramificação.

Os algoritmos de elaboração de planos condicionais deste trabalho, e que serão discutidos no próximo capítulo, são semelhantes a abordagem do *Warplan-C*. Extensões são feitas para permitir diversos efeitos disjuntivos diferentes e para condensar os planos de ordem total em planos condicionais menores, convergindo as linhas de execução, além de extrair seqüências de ações parcialmente ordenadas.

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*, (HOFFMANN; BRAFMAN, 2005) e o POND[25] (BRYCE; KAMBHAMPATI, 2005).

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[26]), que possibilita executar planejamento *condicional*.

O *Contingent-FF* é um sistema de planejamento independente de domínio. O sistema estende 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 estendendo 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.

3.3.3.2 Planejamento Probabilístico

No planejamento probabilístico, não se faz uso da monitoração do ambiente, pois, pressupõe-se que a situação do estado corrente pode ser conhecida com base na probabilidade do sucesso do efeito de cada ação anterior no mundo. Uma das abordagens mais comuns é utilizar *Processos de Decisão de Markov (MDPs[27])*. Os planos são representados então. por cadeias de *Markov,* nas quais, um estado é conectado às possíveis ações que podem ser aplicadas nele. Este mapeamento é feito por uma função π, chamada de *política*, que fornece a probabilidade de êxito (GEFFNER, 1998). Assim, a *corretude* de um plano corresponde diretamente à sua probabilidade de êxito.

3.3.4 Planejamento Conformante

O *planejamento conformante*[28] é um caso particular do planejamento condicional e muito pesquisado atualmente. Nesta abordagem ocorre a geração de planos em situações onde não se conhece o estado inicial, nem o resultado da aplicação dos efeitos das ações. Também,

---

[25] POND - *Partially Observable Non-Deterministic planner.*
[26] LUG - *Labelled Uncertainty Graph*.
[27] Sigla em para *Markov Decision Processe.*
[28] Tradução livre para *Conformant Planning*.

não há possibilidade de monitorar o ambiente para obter o estado atual do mundo. Os planos gerados devem atingir a meta independente do estado inicial e do conhecimento do efeito das ações. Esta abordagem apresenta vantagens apenas quando não é possível obter informações do ambiente. Na verdade, o planejamento conformante recebeu mais atenção nos últimos anos do que o planejamento contingente. As evoluções neste campo foram maiores que no outro. Porém, suas abordagens estão rapidamente sendo transpostas para o campo de planejamento condicional. Exemplos são: o próprio POND, que executa tanto planejamento *conformante* quanto *contingente*, e a o planejador *Conformant-FF* (BRAFMAN; HOFFMANN, 2004).

3.4 Considerações Finais

Neste capítulo fez-se uma revisão das técnicas de planejamento desde sua origem até as mais atuais. Além do planejamento clássico com seu determinismo, foram analisadas as abordagens do planejamento em ambientes dinâmicos. Estes conceitos serão fundamentais no próximo capítulo onde serão apresentados algoritmos para integrar estas técnicas de planejamento com o workflow.

Capítulo 4

Usando o Planejamento para Modelagem de Workflow

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 no envolvimento de agentes para sua execução. Ambas as abordagens produzem um roteiro para a solução genérica de instâncias de um problema particular dentro do mesmo domínio. Este roteiro é uma lista de atividades para o workflow e uma seqüência de ações para o planejamento. Quando seguido corretamente, ambos conduzem à meta. No campo de workflow, a meta é a solução de um caso, que pode ser resolvido instanciando-se um processo, que é o roteiro. O processo deve ser completo, ou seja, deve contemplar todas as situações de exceção previstas de forma a sempre conduzir ao seu final. No campo de planejamento, a meta é a solução do problema original. A solução do problema significa satisfazer a meta, que pode ser uma disjunção de submetas, e que é atingida pela execução das atividades do roteiro, que é o plano elaborado.

O Agente de planejamento proposto nesta abordagem recebe como entrada um conjunto de dados no formato da linguagem XPDL que descreve entre outras informações as atividades do processo. Depois, manipula esta informação de forma a possibilitar a geração consistente de um equivalente em PDDL, linguagem padrão para padronização de domínios de planejamento. Logo após, chama o planejador para a produção de um plano consistente, que leve do estado inicial à meta. A meta, neste caso, é a conclusão do processo de workflow. Ela deve representar uma disjunção de situações que caracterizam o término do processo e não a satisfação de uma situação específica. Por exemplo, um processo de aprovação de crédito

pode ser concluído com a aprovação ou negação do mesmo, mas ambos os casos correspondem ao término do processo.

A proposta apresentada neste trabalho busca demonstrar a viabilidade prática em se usar técnicas de planejamento aplicadas à modelagem de processos. Para tanto, foi feita opção por se usar um ou mais planejadores externos, disponíveis e validados pela comunidade que trabalha com as pesquisas neste campo. Desta forma, o agente modelador/planejador (*Agente K*) tem a tarefa de mapear a linguagem XPDL em PDDL, fazendo as manobras necessárias para a correta execução por parte do planejador. Como os planejadores clássicos retornam apenas um plano, que leva do início até a meta, em uma seqüência de ações ou atividades, o *Agente K* tem a tarefa adicional de identificar a coleção de planos que formam um workflow completo e ordenar adequadamente as atividades para compor o mapa do processo.

4.1 Infra-estrutura da Nova Abordagem

Buscando convergir mais rapidamente estas áreas em uma nova proposta, este trabalho apóia-se no uso de ferramentas e *frameworks*[29] disponíveis. Portanto, o objetivo principal será integrar ferramentas de modelagem e execução de workflow com algoritmos de planejamento. Como plataforma para a modelagem de processos foi escolhida a ferramenta Enhydra[30] JaWE[31], uma ferramenta gráfica de código aberto para modelagem de processos de workflow, que segue as especificações do WfMC e suporta a linguagem XPDL como formato nativo. A seguir descrevem-se as principais funcionalidades da proposta ilustrada na Figura 23.

Modelagem de Processos (JaWE): A ferramenta JaWE (Figura 24) é modificada para interagir com o agente de planejamento *K* que interpreta a definição de processos dada na linguagem XPDL e retorna o processo estruturado na mesma linguagem.

---

[29] *Framework* – significa Arquitetura ou Estrutura. Neste trabalho significa um conjunto de programas de computador que serão utilizados e/ou modificados.

[30] O projeto Enhydra.org é similar ao Apache, mas com um foco em sistemas de E-Business software. Enhydra é um *application server* de código aberto suportado pelo consórcio Objectweb, uma comunidade de software de código-aberto criada no final de 1999 pela Bull, France Telecom R&D and INRIA (Institut National De Recherche En Informatique Et En Automatique). Em 2002, ObjectWeb evoluiu para um consórcio internacional hospedado pelo INRIA, instituto nacional francês para a pesquisa na informática e automação. O consórcio é uma organização independente sem fins lucrativos aberta às companhias, instituições e indivíduos. http://www.enhydra.org/ e http://consortium.objectweb.org

[31] Java Workflow Editor

Máquina de Workflow (Shark): A ferramenta Shark é utilizada para executar os processos de workflow gerados na ferramenta JaWE. A máquina de workflow interpreta e instancia os processos gerando as listas de trabalho correspondentes para cada responsável, em cada passo do processo. Inclui também os subsistemas de Administração e Monitoramento dos processos. Através desta ferramenta, também é possível atribuir agentes autômatos para execução de determinadas atividades. A maior parte destas funcionalidades já é nativa da ferramenta, a qual atende completamente as especificações do WfMC.

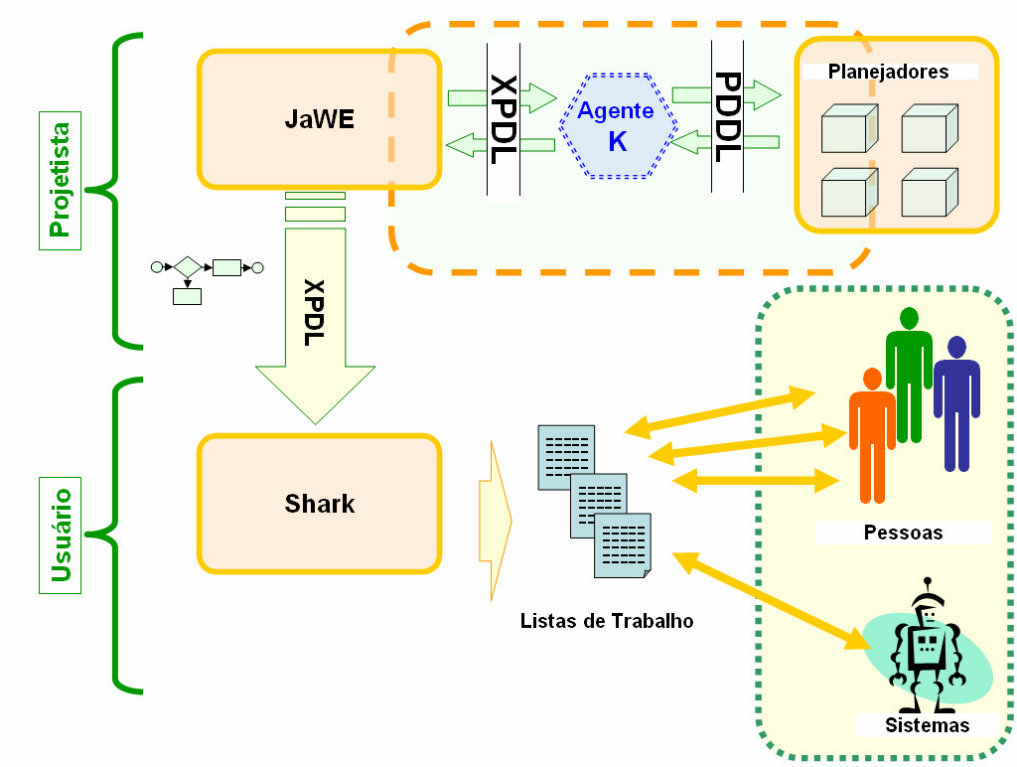


Figura 23: Esquema de integração de AI Planning com o workflow.

AgentePlanejador (Agente K): O agente de planejamento tem a função de converter as especificações da linguagem de modelagem de processos na linguagem de definição de domínios para planejamento (PDDL), submeter uma requisição de geração de plano para a ferramenta de planejamento e por fim converter o resultado (plano gerado) novamente no formato XPDL, devolvendo o controle para a ferramenta de modelagem. Será utilizada como base a extensão *PDDL versão 2.2* apresentada em (EDELKAMP; HOFFMAN, 2004).

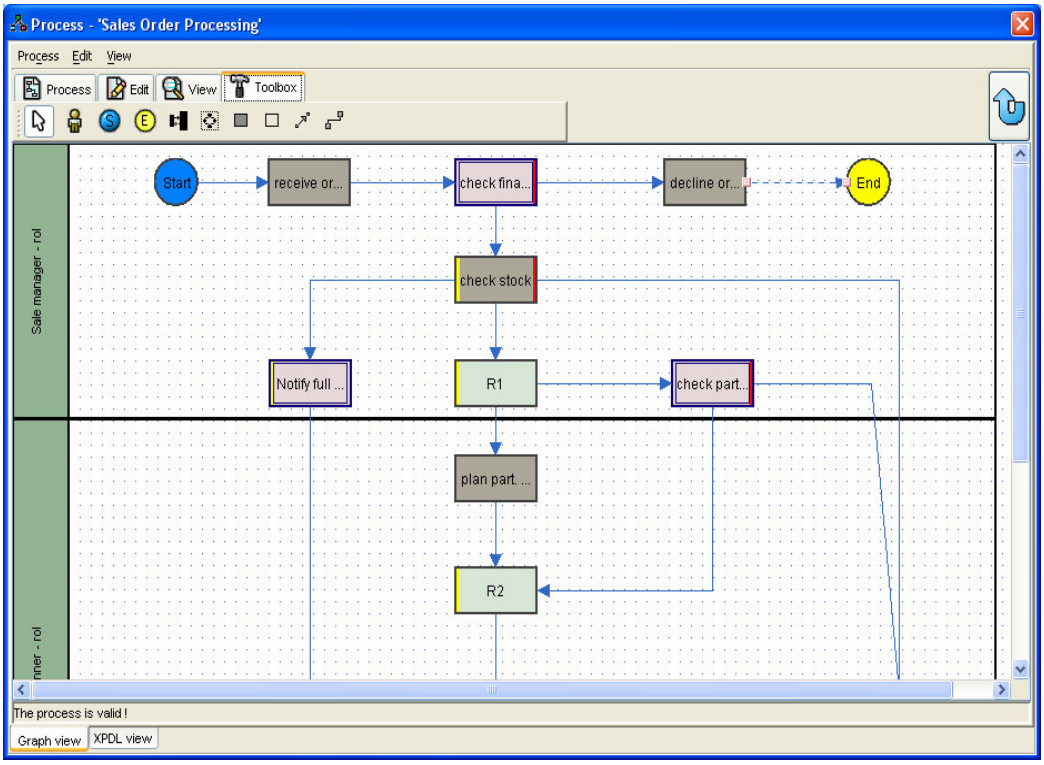


Figura 24: Tela de mapeamento de processos do JAWE.

4.2 Formalização do Planejamento

Tradicionalmente, um plano é visto como a solução de um problema de planejamento. O Planejamento busca descobrir uma sucessão de ações para chegar à meta, ou seja, o próprio plano (RUSSELL; NORVIG, 2003). As definições de planejamento apresentadas no relatório do PLANET complementam sutilmente esta definição, introduzindo o conceito de processo. O processo seria uma descrição de um conjunto ordenado de atividades. Já o plano, seria a descrição de uma seqüência de atividades para atingir um dado objetivo, isto é, um processo instanciado (PLANET, 2003). Estas definições são complementares, e introduzem o processo como o conjunto de todos os planos válidos.

A aplicação de planejamento ao domínio de workflow tomará como ponto de partida o modelo do planejamento clássico. Para este trabalho, o planejamento é definido através da tripla $(\mathcal{A}, I, \mathcal{G})$ cujos elementos são o conjunto de ações, o estado inicial, e o estado final, ou meta. Seja $\mathcal{P}$, o conjunto de todas as proposições que representam fatos no mundo. O estado corrente, ou mundo, é designado por w e representa o subconjunto de proposições satisfeitas em $\mathcal{P}$ tal que $w \subseteq \mathcal{P}$ no mundo. Em STRIPS, as ações são triplas do tipo $(pre(a), add(a), del(a))$

cujos elementos pertencem ao conjunto de proposições $\mathcal{P}$ e correspondem respectivamente às pré-condições e aos efeitos das ações – esta última através das listas de adição e remoção. Uma ação a é aplicável em w if $w \supseteq pre(a)$. Aplicar a em w, substitui w por w' tal que $w' = w - del(a) + add(a)$. Assume-se que $del(a) \cap add(a) = \{\}$.

As pré-condições de ações são definidas como fórmulas da lógica de primeira ordem (LPO). As metas também passaram a ser representadas como tais fórmulas. Um efeito será representado no mesmo formato. Um efeito condicional é designado pela tripla ($con(e), add(e), del(e)$). Uma ação a é aplicável em w se $w \models pre(a)$. Aplicar a ação a em w resulta em um novo estado alterado com todos os efeitos da ação. A intersecção dos conjuntos, formados cada um respectivamente pelas listas de adição e remoção, é vazia.

4.3 Atividades versus Operadores

Em planejamento, os operadores representam as ações que poderão ser tomadas pelos agentes. 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 outro descrevendo seus 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 desenhista 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. Opcionalmente, à medida que vai construindo o processo, o projetista também pode estabelecer relações entre as atividades na forma de conexões, que produzem um fluxo de uma atividade para outra. Estas conexões são chamadas *restrições de transações* e podem implementar regras lógicas (condições) para sua ativação. Entretanto, não existe uma relação direta de causa e efeito necessário para o uso com o mecanismo de planejamento. Por isto, dependemos da informação contida nas regras de produção.

```
<Activity
   Id="Act4"
   Name="Entrega Pedido">
   <Description>
       Entrega o pedido para o
       cliente
   </Description>
   <Performer> Agente_Entregador
   </Performer>
   <ExtendedAttributes>
     <ExtendedAttribute
        Name="Pré-condição"
        Value="Pedido_Pronto "/>
     <ExtendedAttribute
        Name="Efeito"
        Value="Pedido_Entregue"/>
   </ExtendedAttributes>
</Activity>
```

(a) Atividade em XPDL

```
(:action Entrega_Pedido
  :precondition (Pedido_Pronto)
  :effect  (Pedido_Entregue)
)
```

(b) Atividade em PDDL

Figura 25: Comparação entre atividade em XPDL e ação em PDDL.

A Figura 25 mostra um mapeamento de uma atividade descrita em XPDL para um operador de ação descrito em PDDL. A regra de produção expressa pelos atributos estendidos de pré-condição e efeito é mapeada diretamente para a sintaxe PDDL. As cláusulas das regras de produção devem seguir a sintaxe das LPO para prover uma tradução mais direta entre as duas linguagens.

O conjunto de todas as regras de produção de um processo é designado por L_r onde, cada elemento representa uma regra de produção e está na forma *(c,e)* onde *c* é uma cláusula na forma LPO que denota uma pré-condição; *e* é uma cláusula em LPO que denota um efeito.

Definição 4.1: *Uma atividade é uma tupla na forma $A = (D,T,R)$, onde D é a descrição da atividade, $T \subseteq (A \times A)$ é um conjunto de pares ordenados que representam as transições entre atividades, e $R \subseteq L_r$ é o conjunto de todas as regras válidas para a atividade.*

Inicialmente, antes de serem submetidas ao agente de planejamento, cada atividade deve possuir apenas os valores de *D* e *R*, o agente de planejamento deve retornar o valor de *T* para todas as atividades necessárias ao processo.

4.4 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.

Quando as atividades envolvem ramificações, elas podem ser de dois tipos: separação ou junção (*Split/Join*). As ramificações de separação 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. A ramificação condicional obriga a escolha de um caminho para a execução. As ramificações de entrada 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.

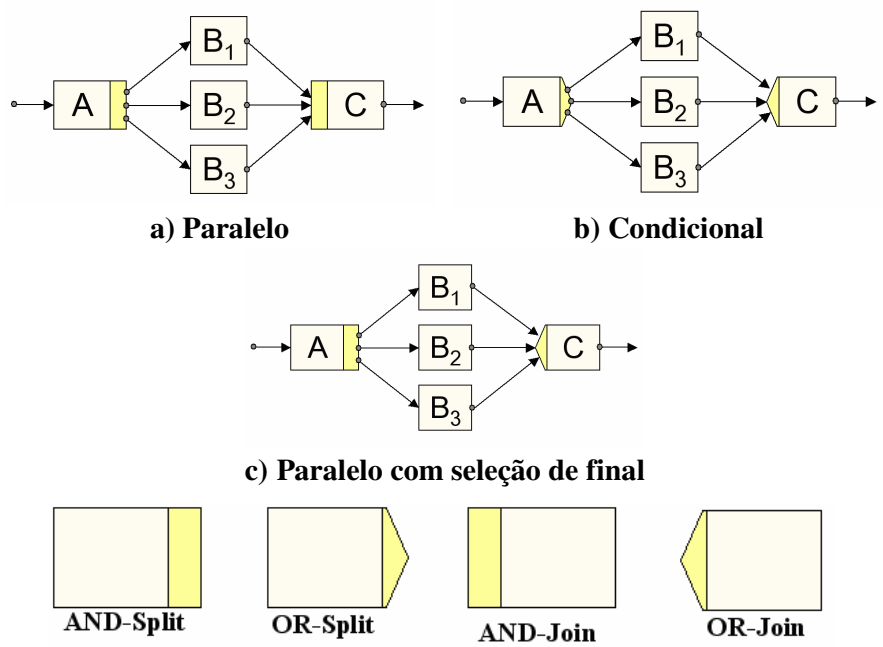


Figura 26: Modelo de ramificação de atividades.

A Figura 26 mostra as três combinações permitidas para as ramificações de atividade no contexto deste trabalho. Um fluxo ***paralelo*** irá permitir a execução simultânea de duas ou mais atividades a partir de uma atividade tipo *AND-Slipt* e voltando a convergir o fluxo mais adiante em outra atividade do tipo *AND-Join*. O fluxo ***condicional*** irá habilitar apenas uma das atividades subseqüentes e voltará a convergir em uma atividade tipo *OR-Join* mais adiante, que precisa apenas ser alcançada por uma das vias. Por último o fluxo do tipo ***paralelo com seleção de final*** irá habilitar várias atividades simultaneamente, mas no ponto de convergência necessita apenas ser alcançado por uma das linhas para retomar a execução do fluxo principal.

4.5 Mapeamento de Atividades em Operadores

O mapeamento de atividades em operadores, não representa problema na maioria dos tipos de conexão. Tanto as conexões de *OR-Join*, *AND-Join* ou *AND-Split* podem ser obtidas naturalmente do planejamento através da correta representação proposicional das pré-condições e efeitos. Porém, existe uma dificuldade inerente ao mapeamento de atividades em ações. Devido à possibilidade de ramificação de fluxo de uma atividade do tipo *OR-Split*, o mapeamento direto torna-se inviável. Suponha o exemplo de um fluxo de aprovação de crédito: Feito um pedido, um agente deve executar uma atividade *A* que determina se o cliente possui crédito ou não. Em um mapeamento direto, seria impossível para o planejador avaliar um caminho válido para atingir a meta, pois existe uma contradição entre os efeitos da atividade *A*, que é verificar o crédito. O resultado desta atividade pode ser ora positivo, ora negativo.

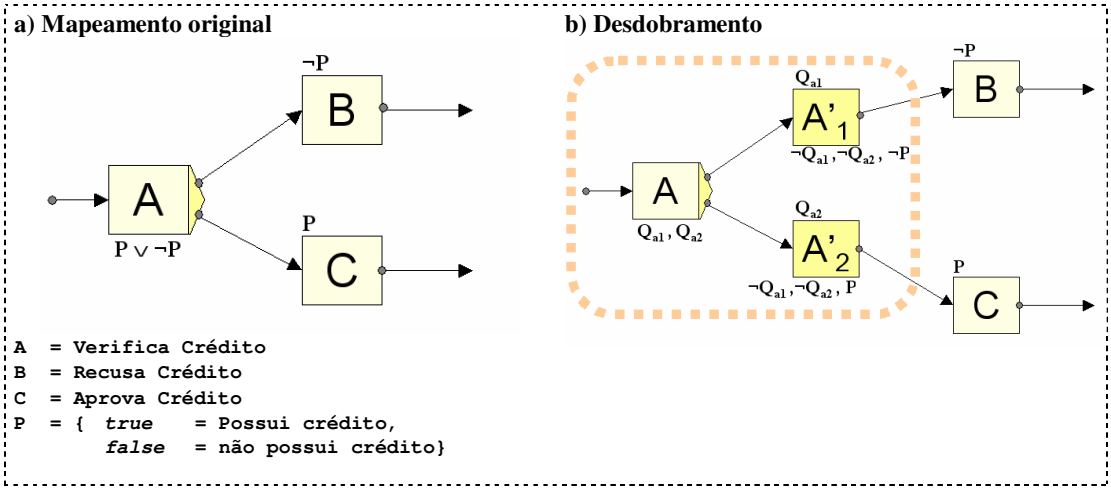


Figura 27: Desdobramento de atividades *OR-Split*.

Para contornar este obstáculo é proposta o desdobramento da atividade *A* em subatividades para cada ramo previsto. Este desdobramento só tem efeito durante o processo de planejamento. Como pode ser visto na Figura 27.b a atividade *A* é dividida em duas, uma para cada um dos efeitos disjuntivos previstos (*P*, ¬*P*). A primeira subatividade faz o roteamento para a atividade *B*. Para tanto, é retirada a lista de efeitos da atividade *A* e substituída por uma lista de efeitos que irão estabelecer um vinculo causal entre as atividades *A* e *A'*. Nas subatividades, cada elemento do efeito disjuntivo da atividade *A* é remontado. As pré-condições destas atividades correspondem à nova lista de efeitos da atividade anterior. E

seus efeitos negativam as pré-condições que estabelecem o vínculo causal entre atividades e subatividades para impedir que o planejador possa retornar a uma delas sem passar pela atividade principal novamente.

Como exemplo, considere que o diagrama da Figura 27 represente um segmento de processo de um agente financeiro, o qual recebe a solicitação de empréstimo de um cliente. Este cliente pode apresentar como garantia um fiador ou um imóvel. Assim as atividades da figura recebem o seguinte significado:

A = Recebe solicitação de crédito;

B = Empresta com garantia de fiador;

C = Empresta com garantia de imóvel;

```
Desdobra_Atividades(N){
A = elements_in(N);
R = {};
for i=1 in length(N) do
   if is_type_ORSplit(A[i])
   then
      T = extract_disj_term(A[i].effects);
      for k=1 in length(T) do
       Q   = new_term();
       LC  = LC  ∪ {Q};
       nLC = nLC ∪ {~Q};
      end for;

      for k=1 in length(T) do
       A' = new_activity();
       A'.precond = LC[k];
       A'.effects = nLC ∪ T[k];
       R = R ∪ {A'};
      end for;

      Tc=extract_conj_term(A[i].effects);
      A[i].effects = Tc ∪ LC;
   end if;

   R = R ∪ {A[i]};

end for
return R
```

Figura 28: Algoritmo: Desdobramento de atividades.

O algoritmo de desdobramento das atividades é mostrado na Figura 28. Ele recebe como entrada um conjunto de atividades (*N*) e retorna um novo conjunto (*R*) acrescido das subatividades do desdobramento e com as atividades *OR-Split* modificadas.

Neste cenário, a atividade *A* possui dois efeitos possíveis $\{\neg P, P\}$. O primeiro é uma *solicitação pendente (com fiador)* e o outro é a negação do primeiro, ou seja, uma *solicitação*

pendente (sem fiador). A pré-condição da atividade *B* é a existência de uma *solicitação pendente (com fiador)*, ou seja, $\neg P$. A pré-condição da atividade *C* é existência de uma *solicitação pendente (sem fiador)*, ou seja, *P*. Como o efeito da atividade *A* é uma disjunção de termos na forma ($\neg P \vee P$), faz-se necessário desdobrá-la.

Aplicando o algoritmo, todas as atividades são avaliadas, começando pela atividade A_i para $i=1$, tal que $A_i = A$. Em seguida, utiliza-se uma função para verificar se a atividade A_i é do tipo *OR-Split*, esta função analisa se há termos disjuntivos na cláusula de efeitos da atividade. Isto é verdadeiro para a atividade *A*, então o próximo passo é obter da cláusula de efeito da atividade os termos disjuntivos ($\neg P$ e *P*). O laço seguinte monta dois conjuntos de novos termos, *LC* e *nLC*. O primeiro contém os termos que serão a ligação causal entre as novas subatividades e a atividade original – na Figura 27 são representados por Q_{a1} e Q_{a2}. O segundo é um conjunto da negação dos termos de *LC*, tal que $\forall x \in LC \rightarrow \exists(\neg x) \in nLC$. Na seqüência tem-se outro laço que processa todos os termos disjuntivos, gerando uma subatividade (*A'*) para cada um deles. Cada subatividade recebe como pré-condição um dos novos termos gerados no passo anterior, isto garante a ligação causal com a atividade original – pode-se observar na Figura 27 que Q_{a1} faz a ligação causal entre A'_1 e *A*. A cláusula de efeitos da subatividade é montada com o conjunto de negação dos novos termos, *nLC*, e inclui ainda o termo disjuntivo que está sendo processado. Isto garante ao mesmo tempo a manutenção do efeito original e a negação dos termos de ligação, impedindo que o planejador monte duas subatividades consecutivas, fato que não pode ocorrer visto que, o objetivo é ramificar a execução. A nova subatividade é adicionada ao conjunto de retorno e o passo seguinte ajusta a atividade original (A_i) para ter como efeitos o conjunto de termos de ligação causal com as novas subatividades. A atividade *A* modificada é adicionada ao conjunto de retorno.

Depois de processadas todas as atividades, o algoritmo retorna o conjunto resultado com as atividades originais, subatividades e atividades modificadas.

4.6 Planejando o Processo de Workflow

Considerando que as atividades foram corretamente modeladas e que o Agente K gerou os desdobramentos das atividades do *OR*, então o próximo passo é gerar uma definição na linguagem PDDL e submetê-la à execução do planejador.

O Planejador retorna um plano, caso ele exista, na forma de uma seqüência ininterrupta de ações a partir do estado inicial até a meta. No caso mais simples, a modelagem não inclui atividades do tipo *OR*. Portanto, podem ser todas seqüenciais ou possuir algum grau de paralelismo. Caso sejam seqüenciais, o resultado do plano é o próprio modelo de workflow.

Definição 4.2: *Seja um plano* $P = \{A_1,\ldots,A_n\}$*, tal que* A_1 *é primeira atividade a ser executada e a próxima atividade é dada por* A_{i+1}*, onde i é o índice da atividade corrente.*

Definição 4.3: *Seja um modelo de processo* $M = \{a_1,\ldots,a_k\}$ *onde cada elemento a é uma tupla da forma* (A_i, A_k) *tomados sobre um plano P, tal que* $efeito(A_i) \cap pré(A_k) \neq \emptyset$ *e* $k > i$*. Ou seja que se existe a transição de uma atividade para outra é preciso que pelo menos um dos efeitos da primeira componha uma das pré-condições da segunda.*

4.6.1 Fluxo Paralelo

A definição de um modelo particular pode envolver eventualmente a execução de atividades em paralelo. Então, suponha inicialmente que só existam atividades seqüenciais e em paralelo, ou seja, nenhuma atividade condicional ou paralela com seleção de final.

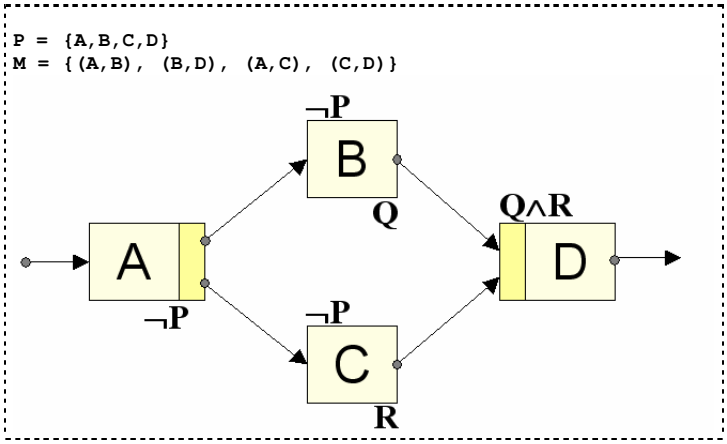


Figura 29: Exemplo de um plano com atividades em paralelo.

A Figura 29 mostra um exemplo de um workflow com atividades em paralelo. Um plano possível, produzido pelo planejador seria *P={A,B,C,D}*. Entretanto este resultado não indica qual a ordenação entre cada par de atividades. Este plano deve ser submetido a um algoritmo de varredura que irá verificar se cada par na forma (A_i , A_{i+1}) satisfaz a **definição 4.3**, onde $efeito(A_i) \subseteq pré(A_{i+1})$, caso não satisfaça então o algoritmo busca uma

atividade A_k tal que $k>i+1$, que satisfaça a condição. Uma vez encontrado um par ordenado, este é incluído em M. A atividade A_{i+1} que não estabeleceu o vínculo com sua atividade imediatamente anterior (A_i) é submetida ao algoritmo de varredura para trás buscando nas atividades anteriores aquela que pode estabelecer um vínculo.

```
extract_model(P){
for i=1 in (length(P)-1) do
   A = elements_in(P);
   if effect(A[i]) ∩ precond(A[i+1]) ≠ ∅
   then
      M := M ∪ pair(A[i],A[i+1]);
   else
    k = i + 1;
    while k <= length(P) do
     k = k + 1;
     if effect(A[i]) ∩ precond(A[k]) ≠ ∅
     then
         M := M ∪ pair(A[i],A[k]);
         exit_while;
     end if
    end while

    h = i + 1;
    while h > 0 do
     h = h - 1;
     if effect(A[h]) ∩ precond(A[i+1])≠∅
     then
         M := M ∪ pair(A[h],A[i+1]);
         exit_while;
     end if
    end while

   end if
end for
return M
}
```

Figura 30: Algoritmo: Extração do modelo do plano.

Para exemplificar consideremos que o diagrama da Figura 29 represente um segmento de processo de um agente financeiro, o qual recebe a solicitação de empréstimo de um cliente, que apresenta como garantia um imóvel. As regras do processo podem determinar que seja feita a avaliação do imóvel e a avaliação da situação financeira do cliente. Assim as atividades da figura recebem o seguinte significado:

A = *Recebe solicitação de crédito;*

B = *Executar análise financeira;*

C = *Executar avaliação do imóvel;*

D = *Calcular limite de empréstimo com base na situação financeira e valor do imóvel;*

Dado a independência entre as atividades *B* e *C*, considera-se que existe a possibilidade de execução simultaneamente. Consideremos como efeito da atividade *A* uma *solicitação pendente*. A pré-condição das atividades *B* e *C* é a existência de uma *solicitação pendente*. O efeito da atividade *B* é a *análise financeira realizada* e da atividade C é a *avaliação do imóvel realizada.* Como pré-condição da atividade *D* temos a *análise financeira realizada* e a *avaliação do imóvel realizada.*

Como as pré-condições das atividades *B* e *C* são iguais, o planejador pode apresentar como solução a seqüência *{A, B, C, D}* ou *{A, C, B, D}*. Consideremos que a solução encontrada foi a primeira. Aplicando o algoritmo da Figura 30, primeiro é avaliada a atividade A_i para $i=1$, tal que $A_i=A$. Em seguida aplica-se a **definição 4.3** para verificar se o par (A_i , A_{i+1}) é ordenado. Como o conjunto de efeitos da Atividade *A* é igual ao conjunto de pré-condições de *B*, então o par (*A, B*) é incluído no modelo *M*.

O processamento prossegue para avaliar a atividade A_i para $i=2$, tal que $A_i=B$. Entretanto, o par (A_i , A_{i+1}) não atende a **definição 4.3** visto que nenhum elemento do conjunto de efeitos da atividade *B* pertence ao conjunto das pré-condições de *C*. Então o par (*B,C*) não pode ser incluído no modelo *M*. Para tratar esta situação, o algoritmo continua com a criação de um laço que verifica todas as atividades subseqüentes, procurando por uma atividade que componha um par com A_i e que atenda a **definição 4.3**. Na primeira iteração deste laço é verificado o par (*B,D*), o qual atende a **definição 4.3**, visto que o efeito da atividade *B* pertence ao conjunto de pré-condições da atividade *D*. Então, o par é adicionado ao modelo *M* e o laço é concluído. Como a atividade A_{i+1} não participou do novo par, o algoritmo prossegue com outro laço, buscando nas atividades anteriores uma que possa estabelecer um par ordenado. De fato, caso isto não ocorresse, o par (*A, C*) não seria inserido no modelo, ou seja, o encaminhamento paralelo não ocorreria, e o processo iria falhar. A primeira iteração deste novo laço testa o par (A_h , A_{i+1}), tal que $h = i - 1$. Isto corresponde ao teste do par de atividades (*A, C*) que satisfaz a **definição 4.3**. Portanto, é inserido no modelo *M* e encerra o laço. Retorna-se novamente ao laço principal para avaliar a atividade A_i para $i=3$, tal que $A_i = C$. Então o par (A_i , A_{i+1}) é testado pela **definição 4.3** para verificar se é ordenado. O par (*C, D*) é adicionado ao modelo *M*, visto que o efeito da atividade *C* pertence ao conjunto de pré-condições da atividade *D*. Em razão da última atividade não poder

estabelecer nenhum par ordenado com nenhuma atividade sucessora, o laço principal processa apenas até a penúltima atividade. Portanto, o algoritmo termina.

O resultado é um modelo *M* na forma {(*A*,*B*), (*B*,*D*), (*A*,*C*), (C,*D*)}.

4.6.2 Fluxo Condicional

Nos processos onde existem atividades com ramificações condicionais, o planejamento torna-se mais complexo. Isto porque o planejador clássico gera um único plano linear o qual atinge a meta, e não considera ramificações condicionais. Mas, precisamos de pelo menos um plano para cada condição possível, a fim de montar o mapa de processo de forma satisfatória.

Considerando a posse do conjunto de atividades, incrementado com o desdobramento daquelas que são condicionais, a solução adotada consiste nos passos a seguir. Primeiro, gerar uma descrição do problema em PDDL. Esta descrição é composta pelo domínio (que é formado pelo conjunto de todas as ações), pelo estado inicial e a meta. Obtida a codificação em PDDL, esta é submetida ao planejador. Este, por sua vez, poderá ou não retornar um plano válido.

```
Extrai_Plano_Condicional(CA_mod){
01: P = gera_plano(CA_mod);
02: P_Cond = P;
03: for i=length(P) to 1 do
04:   A = elements_in(P);
05:   if is_type_Split(A[i])
06:   then
07:     CA' = remove A[i] de CA_mod;
08:     P_novo = Extrai_Plano_Condicional(CA')
09:     P_Cond = PC_Cond ∪ P_novo;
10:   end if;
11: end for
12: return P_Cond
```

Figura 31: Algoritmo: Extração de plano condicional.

Se for retornado um plano, este é então, verificado no sentido inverso da seqüência de execução das ações, ou seja, partindo-se da última ação para a primeira (Figura 31). Esta verificação procura pela existência de uma ação correspondente a uma subatividade. Uma vez encontrada tal ação, ela é retirada do conjunto de ações e o plano atual é armazenado. Antes de executar novamente o planejador, é feita uma ligação das atividades iniciais do plano que

não foram avaliadas. Isto é feito para garantir que o próximo planejamento comece exatamente no estado onde foi encontrada a ação correspondente à subatividade, que implica em uma ramificação condicional. O processo repete-se até que todas as ações representativas de uma ramificação *OR-Split* sejam eliminadas, significando que todas as linhas de execução do processo foram geradas.

Tendo gerado o conjunto de planos, o modelo de processo *M* é obtido aplicando-se o algoritmo de extração de modelo da Figura 30 em todo o conjunto de planos $CP=\{P_1,...,P_n\}$. Concluído o processo e de posse dos pares ordenados de atividades, o Agente de planejamento reconstrói o modelo de processo em XPDL e devolve o controle ao módulo de edição de workflow, *JaWE*.

Com estes algoritmos é possível implementar um planejador condicional. Este planejador será denominado de *METAplan*.

4.7 Implementação dos Algoritmos

Para comprovar os resultados previstos neste trabalho, foi realizada a implementação de um *Sistema para Modelagem Automática de workflow via Planejamento*, denominado *SisMAP*, utilizando-se a linguagem de programação Java. O Planejamento Condicional foi obtido utilizando-se do planejador *METAplan* descrito na seção anterior.

Um dos objetivos é tirar proveito dos planejadores atuais, que estão em constante desenvolvimento, permitindo ao sistema de modelagem uma independência e ganhos, à medida que ocorrem avanços nas áreas de planejamento. Para realizar os experimentos, escolheu-se o planejador *FF* para ser o mecanismo de planejamento clássico do *METAplan*, por ser um bom representante da família de planejadores clássicos, que possui alto desempenho na obtenção de planos ótimos, o que é importante para se ter processos de workflow otimizados, ou seja, sem atividades desnecessárias. Entretanto, esta abordagem pode ser utilizada com qualquer planejador clássico disponível. Desde que, trabalhe com os formalismos da linguagem PDDL e implemente pelo menos os requerimentos para STRIPS e ADL.

A Figura 32 mostra a interface de usuário deste sistema. Para provar sua eficiência foi utilizado como caso de estudo o diagrama de um processo de tratamento de uma reclamação de clientes apresentado em (FRANCIELLI, 2005).

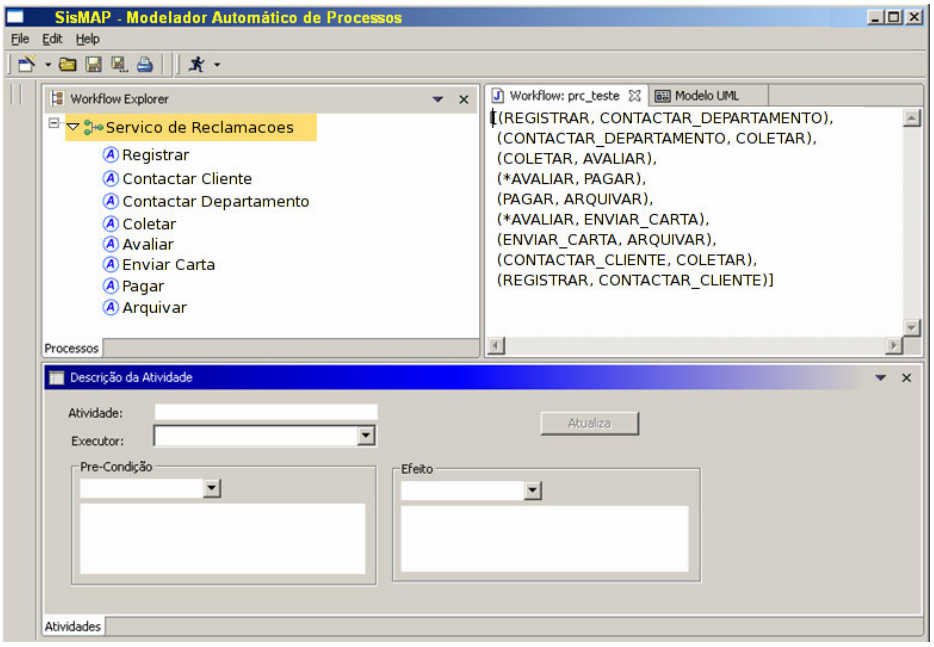


Figura 32: Tela do Sistema de Modelagem Automática de workflow - *SisMAP*.

O diagrama de atividades apresentado na Figura 33 ilustra a seqüência de atividades. O processo começa com o registro de uma reclamação, depois esta reclamação em aberto, ou seja, sem solução, é encaminhada para uma área que irá contatar o cliente para obter maiores informações e o departamento envolvido na reclamação. Estas últimas atividades são feitas em paralelo. Com base nas informações coletadas ocorre uma tomada de decisão que pode conduzir a um pagamento ao cliente ou ao envio de uma carta.

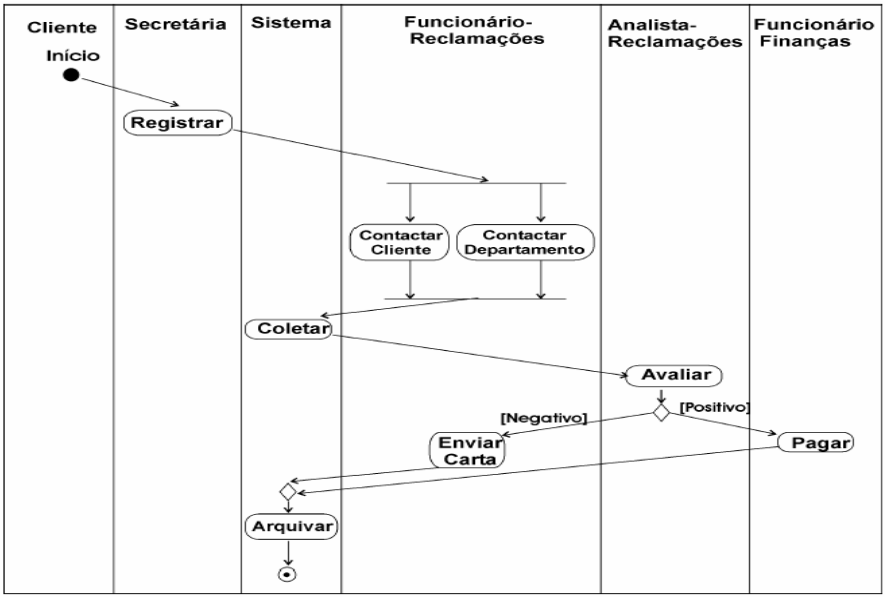


Figura 33: Diagrama de Atividades para o “Serviço de Reclamações”.

O processo mostrado contempla os dois principais desafios da abordagem, que é um nó de decisão e um nó de encaminhamento em paralelo. Ambos devem ser resolvidos pelos algoritmos anteriores utilizando-se também os planejadores.

O primeiro passo é cadastrar todas as atividades com suas pré-condições e efeitos. A Tabela 2 mostra como foi realizado este cadastramento. Note que não é feita nenhuma menção explícita ao processo paralelo, isto porque o sistema deve identificar automaticamente a independência entre as atividades [*Contatar Cliente*] e [*Contatar Departamento*]. A condição de decisão é expressa através da representação disjuntiva do efeito da atividade [*Avaliar*], indicando que o sistema deverá prover a ramificação do processo.

Tabela 2 : Especificação das atividades do processo.

Processo : Reclamação de Clientes			
Atividade	**Agente**	**Pré-condição**	**Efeito**
Registrar	Secretaria		Nova_Reclamacao
Contatar Cliente	Funcionario-Reclamações	Nova_Reclamacao	Cliente_contactado
Contatar Departamento	Funcionario-Reclamações	Nova_Reclamacao	Depto_Contactado
Coletar	Sistema	Depto_Contactado e Cliente_contactado	Coleta_ok
Avaliar	Analista-Reclamações	Coleta_ok	**Positivo ou Negativo**
Enviar Carta	Funcionario-Reclamações	Negativo	Carta_Enviada
Pagar	Funcionario-Finanças	Positivo	Pago
Arquivar	Sistema	Carta_Enviada ou Pago	Fim

O resultado é um modelo de processo formado por pares de ações conectadas umas às outras. Este modelo, obtido a partir do sistema, quando confrontado com o diagrama de atividades da Figura 33 mostra-se correto, contemplando os encadeamentos seqüenciais, paralelo e condicional previstos. Desta forma, o sistema mostra-se eficaz na geração da modelagem de processos. O modelo gerado pode ser então, convertido com facilidade para uma linguagem XPDL e submetido a uma máquina de workflow.

4.7.1 Discussão dos Resultados

Tendo alcançado o objetivo principal de extrair os planos condicionais de planejadores clássicos, utilizar esta técnica para a geração de processo de workflow, e provado sua eficácia, cabe avaliar sua eficiência. Como já abordado, o sistema *SisMAP* é eficaz na geração do processo de workflow, retornando sempre uma solução, caso ela exista. Isto é o que foi

chamado de eficácia. A eficiência busca avaliar se este resultado pode ser obtido em um tempo admissível.

Para estabelecer uma comparação, escolheu-se o planejador condicional POND, considerado um dos planejadores *estado-da-arte* desta área. O sistema *SisMAP* esta apto a trabalhar com um planejador condicional genérico. Podendo, portanto, utilizar-se do planejador *METAPlan* ou do POND. Para o planejador condicional POND o *SisMAP* monta a definição do problema em PDDL utilizando a sintaxe da Figura 34 para representar ações com efeitos condicionais. A declaração *observe(X)* especifica que o efeito não é determinístico. Assim, o planejador sabe que deve produzir uma ramificação do plano neste ponto. A figura mostra uma ação *B* que pode produzir dois efeitos diferentes (representados por *R* ou *S*) de forma não-determinística. Para manter o padrão utilizado neste trabalho faz-se a decomposição da ação B em duas outras ações denominadas B1 e B2. Cada uma destas ações produz um dos efeitos condicionais originais (*R* ou *S*).

```
(:action B
  :precondition (P)
  :effect (and  (observe (QA1)) (observe (QA2)))
       )

(:action B1
  :precondition (QA1)
  :effect (and (S))
)

(:action B2
  :precondition (QA2)
  :effect (and (R))
)
```

Figura 34: Segmento da definição de um problema de planejamento condicional em PDDL utilizado pelo POND.

Os experimentos a seguir foram realizados em um micro computador *Athlon XP 2000+* 1.67 GHz e com memória RAM de 700Mb, rodando *Linux*. Foram feitos dois experimentos denominados *#1* e *#2*. No experimento *#1* o objetivo é medir o desempenho das duas abordagens aplicadas a problemas de planejamento com poucas ações seqüenciais e diferentes quantidades de ações de condicionais, mapeadas a partir de atividades *OR-Split*. Já para o experimento *#2* o objetivo era avaliar os mesmos volumes de ações condicionais, mas incrementando-se a quantidade de ações seqüenciais.

O resultado do primeiro experimento é mostrado no gráfico da Figura 35. Uma análise empírica do comportamento representado no gráfico leva a conclusão que as duas abordagens

têm um desempenho quase similar até o limite de 14 atividades do tipo *OR-SPLIT* incluídas no problema de planejamento. Após este limite, o planejador POND passa a demonstrar um consumo de tempo maior para retornar o plano condicional, em relação à abordagem do *METAplan*.

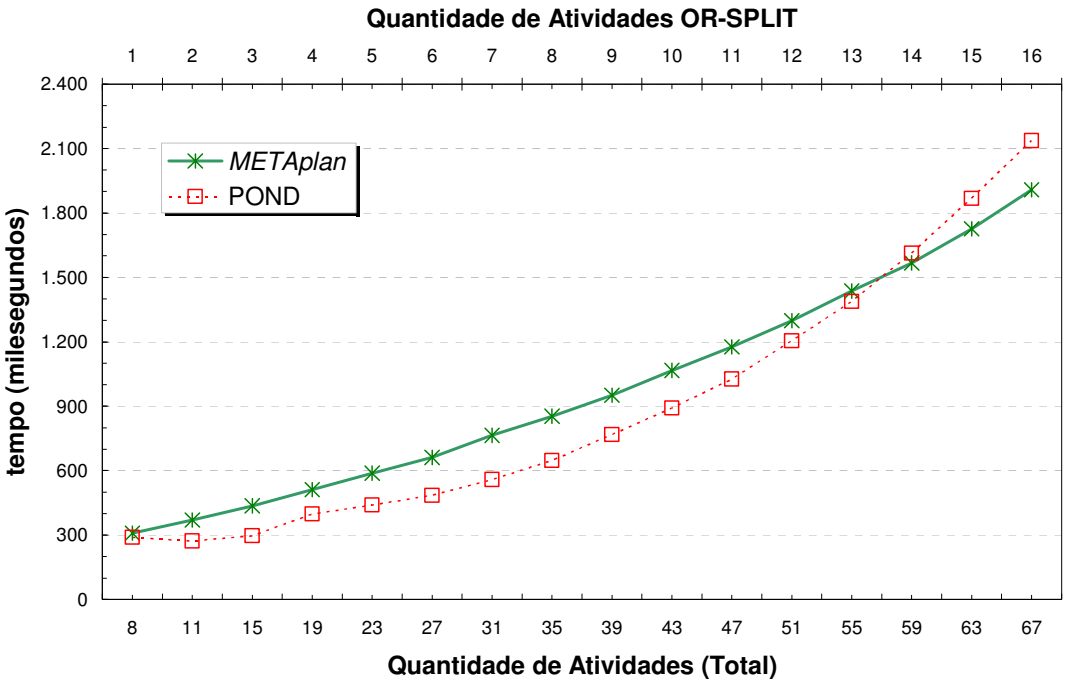


Figura 35: Tempo de execução versus quantidade de atividades *OR-SPLIT*.

Outro fato é que o gráfico mostra uma curva de tendência de comportamento exponencial do tempo, em relação à quantidade de atividades *OR-Split* que foram decompostas, para o planejador POND. Para o *METAplan*, embora ainda persista o comportamento exponencial, este ocorre em uma base menor.

O planejador POND não foi capaz de resolver os problemas nos quais um dos ramos conduzia à meta e outro não. Para a modelagem de workflow isto significa que apenas os caminhos gerados são significativos. Desconsiderar ramos que não conduzem à conclusão do processo, não invalida a modelagem. O modelo pode representar estes ramos como *indefinidos*, permitindo que o desenhista de processos reveja seu levantamento de atividades. Nas duas versões disponíveis do POND (versões 1.1 e 1.1.1), o sistema entrou em um laço infinito e não produziu nenhum resultado. Esta situação não apresentou problemas para o *METAplan*, utilizando o FF (versão 2.3).

O experimento *#2* foi conduzido com a mesma estrutura de problemas do experimento anterior, mas acrescendo-se mais 46 ações seqüenciais. O resultado é apresentado no gráfico

da Figura 36. Pelo comportamento mostrado no gráfico percebe-se claramente o desempenho superior da abordagem *METAplan*.

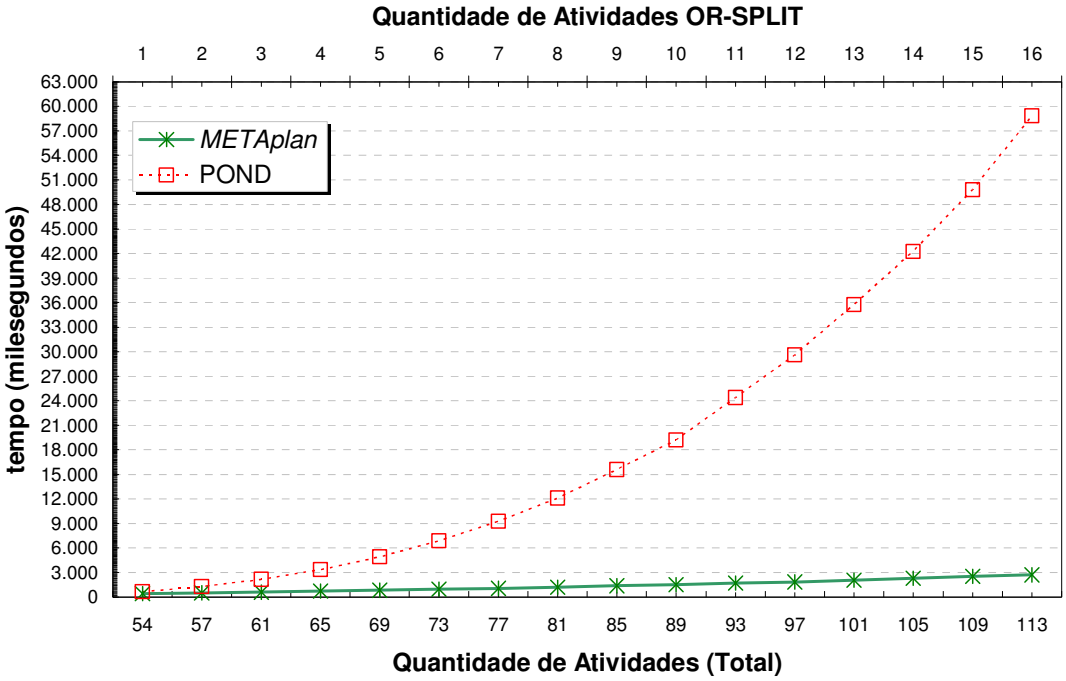


Figura 36: Tempo de execução versus quantidade de operadores.

4.8 Trabalhos Relacionados

Muitos trabalhos foram apresentados com a perspectiva de integração das técnicas de workflow e planejamento nos últimos anos. Pode-se dizer que um dos marcos recentes neste campo foi o trabalho apresentado por Myers e Berry (1998), no qual é feito um estudo das correspondências entre os dois campos e das contribuições possíveis trazidas pela adoção das técnicas de IA.

Um trabalho apresentado logo após mostra o desenvolvimento inicial de um sistema de workflow com controle reativo baseado em IA, chamado SWIM (BERRY, 1999), o qual busca estender o paradigma de workflow para responder a ambientes dinâmicos e com incertezas. Entretanto, não é dado tratamento quanto à geração automática de processos. Os autores endereçam para trabalhos futuros esta e outras questões, como recuperação de falhas e workflow evolutivo.

Outra proposta mostra o uso de técnicas de planejamento hierárquico e condicional no projeto de programas de controle em processos de automação e manufatura (CASTILLO, 2003). Neste caso, o planejamento é utilizado para gerar um programa de controle com

características hierárquicas e modulares. Fazendo uma análise do trabalho, fica clara a semelhança do resultado com a geração de processo de workflow.

Algumas propostas buscam dar flexibilidade à execução do workflow, especialmente aos da categoria *Ad-Hoc*, que não possuem uma seqüência pré-definada. Nesta linha, o trabalho de Bezerra e Wainer (2003), mostra uma contribuição importante com o uso de restrições violáveis para definir *workflows parciais*, isto é, que não possuem uma definição completa podendo ser dinamicamente planejados.

Um estudo de caso apresentado por R-Moreno *et al* (2000), explora os ganhos que um sistema de gerenciamento de workflow pode obter incorporando técnicas de planejamento contingente. O estudo é baseado no sistema de workflow usado pela *British Telecom* na época do estudo, denominado COSMOSS[32] (Sistema Orientado a Clientes para a Gerência de Serviços Especiais). O trabalho ilustra uma situação de instalação de linha telefônica, onde existem várias atividades que podem conduzir a ramos de execução diferentes. Então, é proposto um plano gerado pelo planejador *Cassandra* que contempla todas as atividades do processo.

Mas a proposta que mais se aproxima da apresentada aqui, é a desenvolvida por Aler *et al* (2002). Os autores nos apresentam o SHAMASH, uma ferramenta de modelagem de processos cujas facilidades incluem a definição e uso de padrões organizacionais e a geração automática de modelos, simulação e otimização dos mesmos. Sua arquitetura faz uso de regras de produção para definir a relação entre atividades e as condições de simulação e otimização, empregando um algoritmo RETE. O melhor modelo de workflow é obtido através de uma busca em um espaço de estados gerado por uma máquina de simulação que aplica as regras de produção criadas na definição. Os autores indicam na seção de trabalhos futuros que estenderam o modelo para aplicar outras técnicas de IA. Destacam que as atividades são definidas pelo usuário no módulo de autoria, depois traduzidas na linguagem de planejamento PDL[33], carregado um planejador através de uma interface, este por sua vez gera um plano (uma seqüência de atividades), que finalmente é traduzido de volta para o contexto do SHAMASH. É usado um planejador não-linear, PRODIGY4.0, para gerar estes planos. Os autores concluem que este esquema permite aos usuários concentrarem-se nos requerimentos do processo, deixando que o planejamento produza o modelo mais eficiente. A proposta deste trabalho, apresentada nas seções anteriores, mostra de forma mais consistente a transformação

---

[32] COSMOSS - *Customer Orientated System for the Management Of Special Services.*
[33] PDL do inglês *Planning Description Language*

de atividades do workflow em operadores do planejamento. Demonstra também um algoritmo para obter de planejadores clássicos, um fluxo de trabalho que contemple nós de decisão e consequentemente caminhos condicionais.

4.9 Considerações Finais

Este capítulo mostrou como transformar um problema de modelagem de processos de workflow em um problema de planejamento clássico, e demonstrou que o *METAplan*, utilizando-se dos algoritmos necessários para obter um plano condicional a partir planejadores clássicos, é viável. Nos experimentos realizados, tanto a eficácia quanto a eficiência desta abordagem foram confirmadas. Os processos foram corretamente modelados e em tempo admissível para o problema. A comparação com um planejador condicional POND mostrou uma superioridade do *METAplan* com relação ao tempo de execução quando se incrementa a quantidade de atividades e também quanto à eficácia.

Os resultados obtidos são empíricos, e merecem ser mais explorados. Comparações com outros planejadores condicionais e um estudo mais profundo da complexidade dos algoritmos envolvidos, deve ser realizado. Como se trata da primeira versão deste sistema, otimizações na implementação do algoritmo ainda poderão trazer ganhos adicionais.

Capítulo 5

Conclusão

Neste trabalho tratou-se a integração entre Sistemas de Gerenciamento de Workflow e técnicas de planejamento apoiado em IA, com foco na modelagem de processos assistida por um agente de planejamento. 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 é clara, e está apoiada no encadeamento de atividades e ações que buscam atender um objetivo específico, ou seja, uma meta. Buscou-se então validar a aplicação de planejamento na modelagem de processos através da construção de agente que fizesse uso de ferramentas de planejamento disponíveis, dando flexibilidade de escolha entre planejadores. Isto permite tirar proveito da constante evolução pela qual esta área vem passando nos últimos anos.

A modelagem de processos por sua vez, também é um dos campos de maior atenção para a área de workflow. Quando a quantidade de atividades em um modelo de processo é grande e o nível de variáveis também, o trabalho de modelagem feito pelo projetista pode ser altíssimo e sujeito à falhas e definições ineficientes de processos. Com o uso do planejamento, podemos obter com rapidez e segurança planos otimizados que formam os processos de negócio. Isto foi demonstrado pela construção e teste do *SisMAP* que produziu os resultados previstos no trabalho.

Entretanto, este trabalho vai um pouco além neste tema, com a proposta de extrair planos condicionais utilizando-se de planejadores clássicos. Os resultados foram animadores pela flexibilidade e desempenho do *METAplan* associado ao *FF*, em comparação a um planejador condicional de última geração (POND). Estes resultados positivos tornam recomendável uma pesquisa específica para construir planejadores condicionais de propósito

gerais, baseados nas técnicas atuais de planejamento e na incorporação dos algoritmos apresentados neste trabalho.

Outro trabalho que deve ser endereçado é a integração de planejamento com a execução. O workflow é frequentemente relacionado à ambientes dinâmicos, devido às evoluções sociais discutidas na introdução deste trabalho. Portanto, podem beneficiar-se bastante das técnicas de monitoramento e replanejamento, planejamento reativo, entre outras.

Estas questões estão sendo discutidas no grupo de workflow do programa de pós-graduação a que este trabalho está vinculado.

Referências

AALST, M. P.; KEES, H.*Workflow Management:Models, Methods and Systems.* The MIT Press. Cambridge, Massachusetts. p.368, 2002.

AALST, M. P., et al. *Workflow mining: A survey of issues and approaches*. In: Data & Knowledge Engineering, v. 47(2), Elsevier: p.237-267, 2003.

ALER, R. et al, *A knowledge-based approach for business process reengineering, SHAMASH*, Knowledge Based Systems, 15(8). p. 473-483, 2002.

ALLEN, R. *Workflow: An Introduction.* Open Image Systems Inc., United Kingdom, Acesso em: 15 Ago. 2005.

ARAUJO, R. M.; BORGES, M. R. *Sistemas de Workflow*. In: *XX Jornada de Atualização em Informática* - Congresso da SBC, Fortaleza: 2001.

BERRY P. M.; DRABBLE B. *SWIM: An AI-based System for Workflow Enabled Reactive Control.* Proceedings of the IJCAI Workshop on Workflow and ProcessManagement, IJCAI-99, 1999.

BEZERRA, F. L.; WAINER, J. *Flexibilidade em workflows baseados em restrições.* Simpósio Brasileiro de Multimídia e Web. Salvador, BA, Brasil, 2003.

BLUM, A.; FURST, M. L. *Fast planning through planning graph analysis*. Proceedings . IJCAI-95, Montreal, Canadá, 1995.

BONET, B.; GEFFNER, H. *Heuristic Search Planner 2.0*. AI Magazine, 22(3), p.77-80, 2001.

BRAFMAN, R.; HOFFMANN, J. *Conformant Planning via Heuristic Forward Search: A New Approach*. In: KOENIG, S.; ZILBERSTEIN, S.; KOEHLER, J. *Proceedings of the 14th International Conference on Automated Planning and Scheduling*. ICAPS-04, Whistler, Canada, p. 355-364, 2004.

BRYCE, D.; KAMBHAMPATI, S. *Cost Sensitive Reachability Heuristics for Handling State Uncertainty*. In: 21st Conference on Uncertainty in Artificial Intelligence. University of Edinburgh, Edinburgh, Scotland, 2005.

BYLANDER, T. *The Computational Complexity of Propositional STRIPS Planning*. Artificial Intelligence, v. 69, p.165-204, 1994.

CASATI, F. et al., *Workflow Evolution*. In: *Proceedings of ER'96*. Springer Verlag, Cottbus, Germany, 1996.

CASTILLO, L.; FDEZ.-OLIVARES, J.; GONZÁLEZ, A. *Integrating Hierarchical and Conditional Planning techniques into a software design process for Automated Manufacturing*, 13Th International Conference on Automated Planning&Scheduling (ICAPS), Workshop on Planning under Uncertainty and Incomplete Information. 2003.

CHAFFEY, D. *Groupware, Workflow and Intranets – Reengineering the Enterprise with Collaborative Software*. Digital Press, 1998.

DRUMMOND, M. *Situated Control Rules*. Proceedings of the First International Conference on Principle of Knowledge Representation and Reasoning, Morgan Kaufmann Pub., Toronto, 1989.

EDELKAMP, S.; HOFFMAN, J. *PDDL2.2: The language for the classical part of the 4th international planning competition*. Technical Report 195, Albert-Ludwigs-Universitat,Freiburg, Germany, 2004.

FIKES, R. E.; NILSSON, N. J. *STRIPS: A new approach to the application of theorem proving to problem solving*, Artificial Intelligence 2, p. 189–208, 1971.

FOX, M.; LONG, D. *PDDL+ : Planning with time and metric resources*, Technical report. University of Durham, UK, 2001.

FOX, M.; LONG, D. *PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains*. Journal of Artifificial Intelleligence Research (JAIR), v. 20, p. 61-124, 2003.

FRANCIELLI, F. *Problema do Escalonamento em Tempo Real dos Sistemas de Gerenciamento de Workflow Baseado em um Modelo de Rede de Petri Híbrida P-Temporal*. 2005. Tese (Mestrado em Computação) – Faculdade de Computação, Universidade Federal de Uberlândia, Minas Gerais. 2005.

GEFFNER, H. *Modelling Intelligent Behaviour: The Markov Decision Process Approach*. In : COELHO, H. (ed), *Lecture Notes In Computer Science*, v. 1484. Proceedings of the 6th Ibero-American Conference on Ai: Progress Artificial intelligence. Springer-Verlag, London, 1-12, 1998.

GEORGAKOPOULOS, D.; HORNICK, M.; SHET, A. *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. Distributed and Parallel Databases, 3(2), p.119-153, 1995.

GIL, Y. et al. *Artificial Intelligence and Grids:Workflow Planning and Beyond*. IEEE Intelligent Systems, v. 19, n. 1, p.26-33, 2004.

GINSBERG, M. L. *Universal planning: An (almost) universally bad idea*. AI Magazine, 10(4):40-44.1989.

HOFFMANN, J. NEBEL, B. *The FF planning system: Fast plan generation through heuristic search*. Journal of Artificial Intelligence Research, 14, p.253-302, 2001.

HOFFMANN, J.; BRAFMAN, R. *Contingent Planning via Heuristic Forward Search with Implicit Belief States*. In: BIUNDO, S.; MYERS, K.; RAJAN, K., *Proceedings of the 15th International Conference on Automated Planning and Scheduling*. ICAPS-05, Monterey, CA, USA, 2005.

KAUTZ H.; 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.

KAUTZ H.; SELMAN, B. *Planning as satisfability.* In: LOYD, J. (ed), *Proceedings of the Tenth European Conference on Artificial Intelligence*, p.359-363, 1992.

MARSHAK, R.T. *Workflow: Applying Automation to Group Processes*. In : COLEMAN, D.; KHANNA, R. *Groupware Technology and Applications*, Upper Saddle River, NJ, USA: Prentice Hall, 1995.

MCDERMOTT, D., et al. *PDDL - The Planning Domain Definition Language*, Technical Report CVC TR-98-003 / DCS TR -1165, Yale Center for Communicational Vision and Control, 1998

MEIDANIS, J.; VOSSEN, G.; WESKE, M. *Using Workflow Management In DNA Sequencing*. In: Proceedings: *The First International Conference On Cooperative Information Systems*. Los Alamitos, California, USA. p.114-123. 1996.

MYERS, K. L.; BERRY, P. M. *Workflow Management Systems: An AI Perspective*, Technical Report, Artificial Intelligence Center. Menlo Park, CA (USA): SRI International, 1998.

PEDNAULT, E. P. D. *ADL: Exploring the Middle Ground between STRIPS and the Situation Calculus*, In : BRACHMAN, R.; LEVESQUE, H.; REITER, R. (eds.), *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR-89)*, San Francisco: Morgan Kaufmann, p.324-332,1989.

PEDNAULT, E. P. D. *Toward a mathematical theory of plan synthesis*. PhD thesis, Stanford University, 1986, apud, PENBERTHY, J. S.; WELD, D. *UCPOP: A sound, complete, partial order planner for ADL,* In: Proceedings Third International Conference on Principles of Knowledge Representation and Reasoning, p.103-114, 1992., Acesso em: 01 Set. 2005.

PENBERTHY, J. S.; WELD, D. *UCPOP: A sound, complete, partial order planner for ADL,* In: Proceedings Third International Conference on Principles of Knowledge Representation and Reasoning, p.103-114, 1992., Acesso em: 01 Set. 2005.

PLANET. *PLANET Workflow Management R&D RoadMap*. Disponível em <www.planet-noe.org>. 2003. Acesso em: 01 Set. 2005.

PLESUMS, C. *Introduction to Workflow*. In: Fischer, L. (Ed.): *Workflow Handbook 2002*, Lighthouse Point (FL); : Future Strategies Inc., pp. 19-38, 2002.

REIJERS, H. A., *Design and Control of Workflow Processes*. Springer-Verlag: New York, 2003.

REITER, R. *On closed world data bases*. In: GALLAIRE, H.; MINKER, J. (eds), *Closed Logic and Data Bases*. Plenum Press, 1978.

R-MORENO, M.D.; KEARNEY, P.; MEZIAT, D. *A Case Study: Using Workflow and AI planners*. In: *Nineteenth Workshop of the UK Planning and Scheduling Special Interest Group* (PLANSIG 2000), Reino Unido: p.175-184, 2000.

RUSSELL, S. E NORVIG, P. *Artificial Intelligence – A Modern Approach*, 2. ed. Prentice Hall, 2003.

SCHAEL, T. *et al*. *Design Principles for Cooperative Office Support Systems in Distributed Process Management*. In : *Support Functionality in the Office Environment*, A. Verrijn-Stuart (ed), North Holland. , 1991

SCHOPPERS, M. J. *Universal plans for reactive robots in unpredictable environments*. In: *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, p.1039-1046, 1987.

SELMAN, B; LEVESQUE, H. J.; MITCHELL, D. *A new method for solving hard satisfability problems*. In: ROSENBLOOM, P.; SZOLOVITS, P. (eds), *Proceedings of the Tenth National Conference on Artificial Intelligence*. Menlo Park, California: p. 440-446, 1992.

SUTTON, M. J. D. *Document Management for the Enterprise : Principles, Techniques, and Applications*, New York: John Wiley & Sons, 1996.

WARREN, D. H. D. *Generating Conditional Plans and Programs.* In Proceedings of the Summer Conference on Artificial Intelligence and Simulation on Behavior, p.344-354, 1976.

WFMC - WORKFLOW MANAGEMENT COALITION, 2004, *The workflow reference model*. 2004. Disponível em <http://www.wfmc.org>. Acesso em: 01 Set. 2005.