Fabrício Jailson Barth

Atena: um sistema para suporte ao planejamento na área de Gestão de Projetos

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

São Paulo
2003

Fabrício Jailson Barth

Atena: um sistema para suporte ao planejamento na área de Gestão de Projetos

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

Área de concentração:
Sistemas Digitais

Orientador:
Prof. Dr. Edson Satoshi Gomi

São Paulo
2003

Ficha Catalográfica

Barth, Fabrício Jailson

Atena: um sistema para suporte ao planejamento na área de Gestão de Projetos. São Paulo, 2003. 70 p.

Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Inteligência artificial 2. Geração de Planos em inteligência artificial 3. Sistemas especialistas 4. Administração de projetos I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais. II.t.

"O impossível de hoje será o possível de amanhã
se fizermos o possível de hoje."
- *Paulo Freire*

Agradecimentos

Ao meu orientador, Prof. Edson S. Gomi, que esteve sempre disposto a cobrar por um trabalho cada vez melhor.

Aos inúmeros amigos e colegas "descobertos" durante estes anos, que me mostraram o caminho para chegar até aqui, me mostraram outras formas do pensar e dividiram muitas cervejas, pizzas, e idéias.

Aos professores do Departamento de Engenharia de Computação e Sistemas Digitais, pelos ensinamentos ...

A Fernanda, por ter me ajudado a escolher o nome do sistema.

Resumo

Nesta dissertação é apresentado um sistema de suporte ao planejamento na área de Gestão de Projetos, denominado Atena. Para que um gerente de projetos possa criar um plano executável, que leve em consideração as restrições de custo e prazo, além das complexidades técnicas e as incertezas de execução, o gerente de projetos precisa utilizar o seu conhecimento e experiência sobre a área do projeto (ou domínio do problema) e seu conhecimento sobre como planejar e controlar um projeto. Em muitas situações, mesmo que ele possua todo o conhecimento necessário, o esforço de desenvolvimento de um bom plano desde o início e que seja feito de forma rápida é extremamente elevado se o projeto abranger centenas ou milhares de atividades. Para entender melhor esse problema, basta imaginar o esforço necessário para se definir as atividades do projeto, agrupá-las em fases logicamente coerentes e criar as relações de dependência entre elas. O sistema Atena foi desenvolvido com o intuito de auxiliar o gerente de projetos durante o processo de construção de planos de projetos. O software desenvolvido utiliza um algoritmo planejador de ordem parcial, que manipula conhecimento sobre atividades que podem ser executadas em projetos. As definições básicas desse conhecimento foram desenvolvidas a partir da Ontologia Tove e codificadas utilizando-se a representação Strips. Com o intuito de avaliar a funcionalidade do sistema, foram realizados diversos experimentos com informações de projetos reais. A comparação das soluções propostas pelo Atena com os planos desenvolvidos pelos gerentes de projetos mostra que os resultados obtidos são coerentes com os projetos reais. Espera-se que o sistema Atena possa servir como base para o desenvolvimento de ferramentas para o suporte ao planejamento de projetos e também para a representação do conhecimento sobre a execução de atividades de projetos, através da estruturação e disponibilização desse conhecimento aos integrantes da equipe do projeto.

Abstract

In this dissertation it is presented a system named Atena, designed to help project managers to create project plans. To create an executable plan, satisfying time and cost restrictions, and considering the project technical complexities and execution uncertainty, a project manager has to use his/her knowledge about the problem domain and his/her knowledge about project planning and control. Moreover, in many situations, the effort necessary to develop a good plan from scratch would be extremely high for projects with hundreds or thousands of activities. It is not difficult to imagine, for these situations, the effort necessary to define project activities, to group them in coherent phases, and to create dependencies relationships between them. The Atena System was designed to help the project manager during project plan construction process. The developed software uses a parcial order planning algorithm that uses knowledge about activities that can be executed in projects. The descriptions about project activities are based on Tove Ontology and formally written using Strips notation. The system performance was verified through various experiments using informations from real projects. The plans proposed by Atena were compared with the plans developed by project managers, and the solutions obtained are coherent with real projects. We hope that Atena System could be used as basis for the development of a set of tools to support project planning, to support representation about knowledge of project activities, and to make this knowledge available to members of project team.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

IA Inteligência Artificial

PMBOK *Project Management Body Of Knowledge*

PMI *Project Management Institute - Standards Committee*

POP *Partial Order Planning*

SBC Sistema Baseado em Conhecimento

STRIPS *Stanford Research Institute Problem Solver*

TOVE *Toronto Virtual Enterprise*

WBS *Work Breakdown Structure*

1 Introdução

1.1 Motivação e Contexto

O conceito de projeto abrange trabalhos que visam à criação de produtos ou a realização de serviços que tem como características primárias os fatos de serem únicos (mesmo havendo produtos ou serviços similares haverá aspectos - pessoas, tecnologia, cliente - que os tornam diferentes), serem criados por meio de atividades não repetitivas e terem incertezas associadas à sua realização. Idealmente, um projeto deve ter um escopo bem definido, cujo cumprimento deve ser atingido obedecendo-se às restrições de prazo e custo estabelecidos. Isso leva à necessidade de um bom planejamento, que minimize a possibilidade de descontrole ao longo da execução do projeto. A descrição formal do planejamento é apresentada num documento denominado Plano do Projeto.

Para que um gerente de projetos possa criar um plano executável, que leve em consideração as restrições de custo e prazo, as complexidades técnicas envolvidas e as incertezas de execução, o gerente de projetos precisa utilizar o seu conhecimento e experiência sobre a área do projeto (ou domínio do problema) e seu conhecimento sobre como planejar e controlar um projeto.

Em muitas situações, mesmo o gerente de projeto possuindo todo o conhecimento necessário, o esforço para que ele possa desenvolver um bom plano desde o início e rapidamente é extremamente elevado se o projeto abranger centenas, milhares ou mais atividades. Para entender melhor esse problema, basta imaginar o esforço necessário para se definir as atividades do projeto, agrupá-las em fases logicamente coerentes e criar as relações de dependência entre elas.

Ao se pensar no problema de minimização do esforço dedicado ao desenvolvimento de planos, gerou-se uma motivação para desenvolver um Sistema Baseado

em Conhecimento (SBC) que auxilia o gerente de projetos durante a fase de planejamento.

Sistemas Baseados em Conhecimento são programas que realizam inferências sobre uma base de conhecimento com a finalidade de resolver problemas (NILSSON, 1998). A importância da construção de Sistemas Baseados em Conhecimento para as organizações encontra-se na capacidade desses sistemas de preservar, aproveitar e fazer uso do conhecimento e da experiência dos seus membros no processo de tomada de decisões.

O Sistema *Atena*[1] é um sistema baseado em conhecimento que recebe como entrada uma descrição de situação inicial e uma descrição dos objetivos a serem atingidos, e sugere um ou mais planos possíveis, que descrevem o conjunto de atividades que devem ser executadas, os recursos necessários (pessoas e equipamentos) e as relações de dependência entre as atividades, sendo que, os planos sugeridos devem ser consistentes com os objetivos fornecidos e com as soluções usualmente adotadas por um especialista na área.

Para executar esta tarefa, o sistema deve possuir uma base de conhecimento capaz de representar o conhecimento da empresa pertinente ao domínio do problema, ou seja, ser capaz de representar as atividades a serem executadas no âmbito de um projeto, os recursos que ela dispõe e a alocação dos recursos nas atividades. Além disto, é necessário possuir um mecanismo de inferência que raciocine sobre o conhecimento armazenado, e possa propor possíveis soluções para o problema de planejamento.

No processo de construção da base de conhecimento dos projetos já executados pela empresa, tenta-se reutilizar definições de atividades, recursos e outros objetos, além das relações pertinentes ao domínio de discurso, através da ontologia TOVE (*TOronto Virtual Enterprise*) (FOX; GRüNINGER, 1998) e adaptar, se necessário, esta representação ao mecanismo de inferência.

Para o desenvolvimento do mecanismo de inferência serão adotados algoritmos e técnicas de planejamento da área de Inteligência Artificial (IA). Ao longo dos anos, diversos algoritmos tem sido desenvolvidos para resolver problemas de planejamento. Com o avanço do poder de processamento muitas técnicas de planejamento podem ser agora implementadas (YANG, 1997). Pode-se desenvolver

[1] No apêndice I é apresentada a justificativa para a escolha desse nome

sistemas de planejamento para gerar planos automaticamente, para selecionar ações entre uma quantidade de alternativas, para procurar por falhas em planos complexos e para sugerir como reduzir o custo de um plano fornecido. Diversas soluções tem sido implementadas nas última décadas nas mais diversas áreas (AYLETT et al., 2000). Por exemplo, soluções para planejar o dia-a-dia de um astronauta em uma missão espacial (KORTENKAMP, 2003), planejamento das atividades de um satélite (AARUP et al., 1994), planejamento de questões militares estratégicas (CURRIE; TATE, 1991) e no auxílio na elaboração de processos de negócio (MORENO; KEARNEY, 2002).

1.2 Objetivos

O objetivo geral deste trabalho é o de contribuir para minimizar o esforço de desenvolvimento de planos para projetos. De forma concreta, o objetivo deste trabalho de pesquisa é desenvolver um sistema baseado no conhecimento da organização para auxiliar o gerente de projetos durante a fase de planejamento de um projeto. Para isso, é necessário:

i. identificar em que contexto o sistema irá atuar e quais os requisitos o sistema deve possuir;

ii. desenvolver um modelo para representar o conhecimento desta organização, eventualmente, reutilizando modelos já existentes;

iii. realizar um estudo das técnicas de planejamento em inteligência artificial e comparar a utilidade dessas técnicas no domínio de gestão de projetos;

iv. definir uma arquitetura que comporte os modelos e técnicas escolhidas;

v. testar a funcionalidade do sistema.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte forma:

i. no capítulo 2 são apresentados conceitos de Gestão de Projetos, dando ênfase aos problemas enfrentados durante o desenvolvimento de planos para projetos;

ii. no capítulo 3 é discutido a importância da utilização de ontologias pré-existentes no desenvolvimento de novos sistemas baseados em conhecimento, além de apresentar a ontologia TOVE, a qual pretende-se utilizar no desenvolvimento do modelo para representar o conhecimento sobre projetos;

iii. no capítulo 4 é apresentado um tutorial sobre as técnicas de planejamento na área de Inteligência Artificial e é justificada a escolha do algoritmo utilizado na implementação do mecanismo de inferência do sistema desenvolvido neste trabalho;

iv. no capítulo 5 são descritos os requisitos que guiaram a implementação do sistema *Atena*, qual a arquitetura adotada e quais algoritmos e técnicas foram utilizados para implementar o sistema;

v. no capítulo 6 são descritos o método de avaliação dos resultados, os experimentos realizados com o sistema e a avaliação dos resultados, e;

vi. no capítulo 7 são apresentadas as considerações finais.

2 Gestão de Projetos

Este capítulo tem como objetivo descrever conceitos e definições relevantes para compreensão da gestão de projetos, dando ênfase aos aspectos relacionados com o desenvolvimento de planos de projetos.

2.1 Conceito de Projeto

Projetos são caracterizados por serem executados por pessoas e sofrerem restrições[1] de recursos, prazo e custo (PMI, 2000). Em qualquer trabalho, as atividades precisam ser planejadas, programadas e, durante a execução, precisam ser controladas.

Um projeto é definido como sendo "uma empreitada temporária criada para gerar um produto ou um serviço único" (PMI, 2000). Temporário significa que cada projeto tem um começo e um fim definidos e único significa que o produto ou o serviço é diferente, de alguma maneira, dos produtos ou serviços similares já produzidos.

Os projetos podem ser realizados em todos os níveis da organização. Podem envolver uma única unidade de uma organização ou podem cruzar os limites da organização. Os projetos são freqüentemente componentes críticos da estratégia de negócio de uma organização. Exemplos dos projetos incluem: desenvolver um novo produto ou serviço, efetuar uma mudança na estrutura da organização, projetar um veículo novo, entre outros.

[1]Restrição é uma limitação do Projeto, que pode ser de ordem temporal, de ordem orçamentária, ou restrição de habilidades e capacidades (recursos).

2.2 Conceito de Gestão de Projetos

A gestão de projetos é a aplicação de conhecimento, habilidades, ferramentas e técnicas sobre as atividades do projeto a fim de atender ou exceder os requisitos do projeto (PMI, 2000). Os *stakeholders* do projeto são todas as pessoas e organizações cujos interesses são afetados pelo projeto. As ações para atender as necessidades e expectativas dos *stakeholders* do projeto envolve balancear demandas competitivas entre escopo, prazo, custo e qualidade.

A gestão de projetos é uma atividade interativa - uma ação, ou falta de ação numa área, usualmente afeta também outras áreas. Por exemplo, uma mudança de escopo quase sempre afeta o custo do projeto. Entretanto, ela pode ou não afetar a moral da equipe e a qualidade do produto.

Estas interações freqüentemente exigem balanceamento entre os objetivos do projeto - consegue-se uma melhoria numa área somente através do sacrifício de desempenho em outra. Balanceamentos específicos de performance podem variar de projeto a projeto e de organização a organização. Uma gestão de projetos satisfatória requer uma administração efetiva dessas interações.

Para auxiliar o entendimento da natureza da integração na gestão de projetos, e para enfatizar a importância da própria integração, o *Project Management Body of Knowledge* (PMBOK) (PMI, 2000) descreve a gestão de projetos em termos de processos e de suas interações.

Um processo é composto por uma série de ações que visam transformar um insumo (entrada) em um produto (saída) com o auxílio de recursos, infra-estrutura e regras de controle. Os processos dos projetos, enquadram-se nos processos de gestão de projetos ou nos processos de execução do projeto.

2.3 Processos da Gestão de Projetos

Os processos de gestão da gestão de projetos são processos abstratos que dividem os projetos em várias fases visando um melhor controle gerencial e uma ligação mais adequada de cada projeto aos seus processos de execução (figura 2.1), que por sua vez, são específicos do domínio da aplicação. Por exemplo, os processos de execução de um projeto na área de telecomunicações são, na sua maioria,

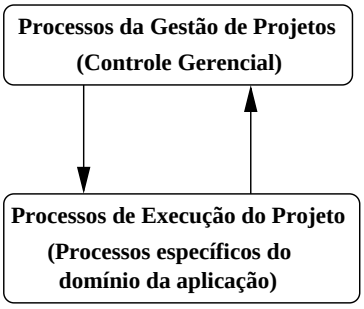

Figura 2.1: Relação entre os processos da gestão de projetos e os processos de execução.

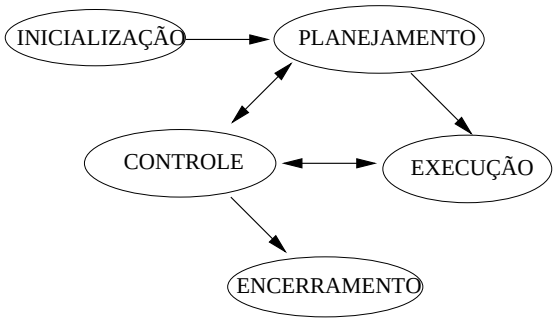

Figura 2.2: Conjunto de processos da gestão de projetos.

diferentes dos processos de execução de um projeto na área de desenvolvimento de software, porém os processos de gestão de projetos são os mesmos para ambas as áreas.

Os processos de gestão de projetos são organizados em cinco grupos: inicialização, planejamento, execução, controle e finalização (figura 2.2).

Os processos de inicialização marcam o nascimento do projeto. É neste momento que é autorizado o início do projeto e é nomeado o Gerente do Projeto.

O planejamento define o que deve ser feito, de uma maneira alinhada e colaborativa (GOMI, 2002a). O controle consiste no acompanhamento das atividades, com base no plano do projeto, com a finalidade de medir o progresso, comparar o previsto com o realizado e fazer os ajustes necessários no projeto.

Na fase de execução é quando os trabalhos são realizados, conforme definidos no planejamento. Na finalização os resultados do projeto são registrados, guardando a história do projeto, e o aceite formal é obtido.

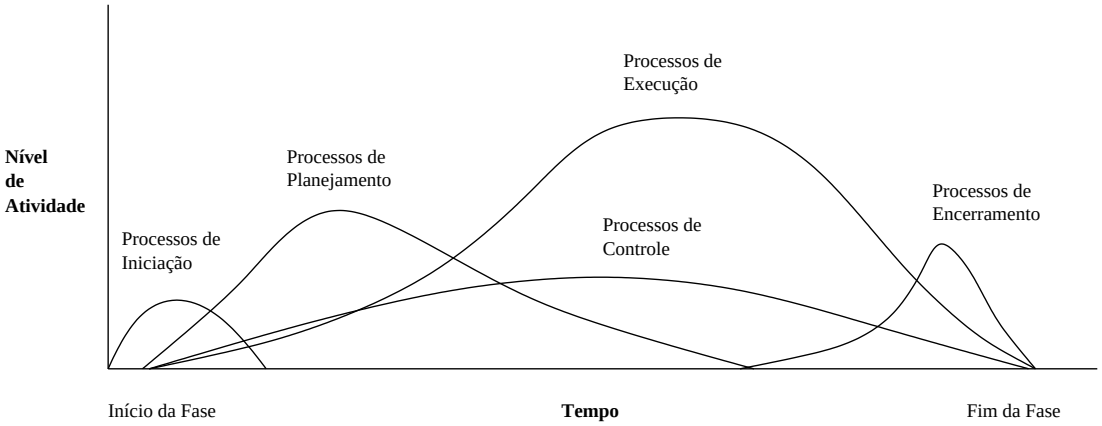

Figura 2.3: Sobreposição dos grupos de processos.

Os grupos de processos se ligam pelos resultados que produzem - o resultado ou saída de um grupo torna-se entrada para outro. Estas conexões são mostradas na figura 2.2. Além disso, os grupos de processos da gestão de projetos não são separados ou descontínuos, nem acontecem uma única vez durante todo o projeto; eles são formados por atividades que se sobrepõem, ocorrendo em intensidades variáveis ao longo do projeto.

A figura 2.3 ilustra como os grupos de processos se sobrepõem e variam dentro de um projeto.

2.4 O planejamento do projeto

Nos processos de planejamento a equipe é montada, o prazo e o custo são estimados, os riscos são identificados, as ações corretivas são definidas, a forma de comunicação é estabelecida, o escopo do produto é detalhado e o escopo do projeto é definido.

O escopo do produto é composto pela especificação técnica que descreve o conjunto de funcionalidade e o desempenho desejado para o produto, e deve ser elaborado antes do escopo do projeto. O escopo do projeto define o conjunto dos trabalhos que serão executados para construir e entregar o produto.

A base para o planejamento de qualquer projeto é a definição do escopo do projeto . Com o escopo do projeto é possível planejar o prazo e o custo para

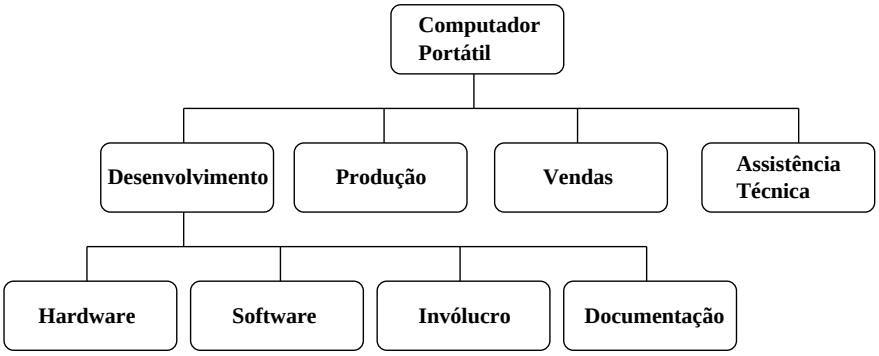

Figura 2.4: Exemplo de WBS.

execução dos trabalhos. O escopo do projeto é descrito pela Estrutura Analítica do Projeto (*Work Breakdown Structure - WBS*).

2.4.1 Estrutura analítica do projeto (WBS)

O termo "Estrutura Analítica do Projeto" (*Work Breakdown Structure - WBS*) é uma estrutura que tem como objetivo fornecer um *check-list* que descreve todas as tarefas de um projeto, e como tal, ela:

i. fornece uma ilustração detalhada do escopo do projeto;

ii. auxilia na estimativa do custo do projeto;

iii. auxilia na montagem da equipe e distribuição do trabalho, e;

iv. permite monitorar o progresso;

A WBS é dividida em pacotes de trabalho. Os pacotes de trabalho correspondem a um agrupamento de atividades que possuem um objetivo comum de alcançar a entrega de um sub-produto do projeto. Um subproduto é um resultado do trabalho, tangível e verificável. Os subprodutos do projeto, compõem uma estrutura lógica, criada para assegurar uma adequada definição do produto do projeto. Por exemplo, considere a WBS especificada na figura 2.4 que identifica as tarefas de um projeto de desenvolvimento de um computador portátil que será executado por um fabricante de produtos eletrônicos.

Nesta WBS está definido que para conseguir desenvolver um computador portátil será necessário um conjunto de atividades de desenvolvimento (que engloba o desenvolvimento do hardware, do software, do invólucro, da documentação, do sistema de produção, do sistema de vendas e do sistema de assistência técnica).

Após a montagem da WBS, para cada atividade será necessário descrever o trabalho a ser feito, definir o critério de finalização, listar os materiais e equipamentos que serão necessários para a execução da atividade e definir os tipos dos profissionais que executarão as atividades (PMI, 2000).

2.4.2 Diagrama de rede

Uma vez definidas as atividades, é preciso criar o diagrama de rede para os pacotes de trabalho da WBS (PMI, 2000). Neste diagrama os relacionamentos entre as atividades devem refletir a seqüência de execução do trabalho. O seu objetivo é descrever quais atividades podem ser feitas em paralelo e quais precisam ser feitas em série. Um exemplo de diagrama de rede pode ser visualizado na figura 2.5 que descreve o conjunto de atividades da WBS da figura 2.4. Ao final da execução de todas as atividades deste diagrama de rede o objetivo do projeto deverá ser alcançado.

Em uma rede de atividades, as atividades devem ser encadeadas levando-se em consideração as relações de dependência:

i. *finish-to-start*: quando uma termina e a outra começa;

ii. *start-to-start*: quando uma começa junto com a outra;

iii. *finish-to-finish*: quando uma termina junto com a outra, e;

iv. *start-to-finish*: quando uma começa e a outra termina.

Na figura 2.6 é possível visualizar a representação gráfica das relações de dependência.

O diagrama pode conter marcos, que são atividades que têm duração zero e servem para identificar eventos importantes do projeto. São pontos de verificação

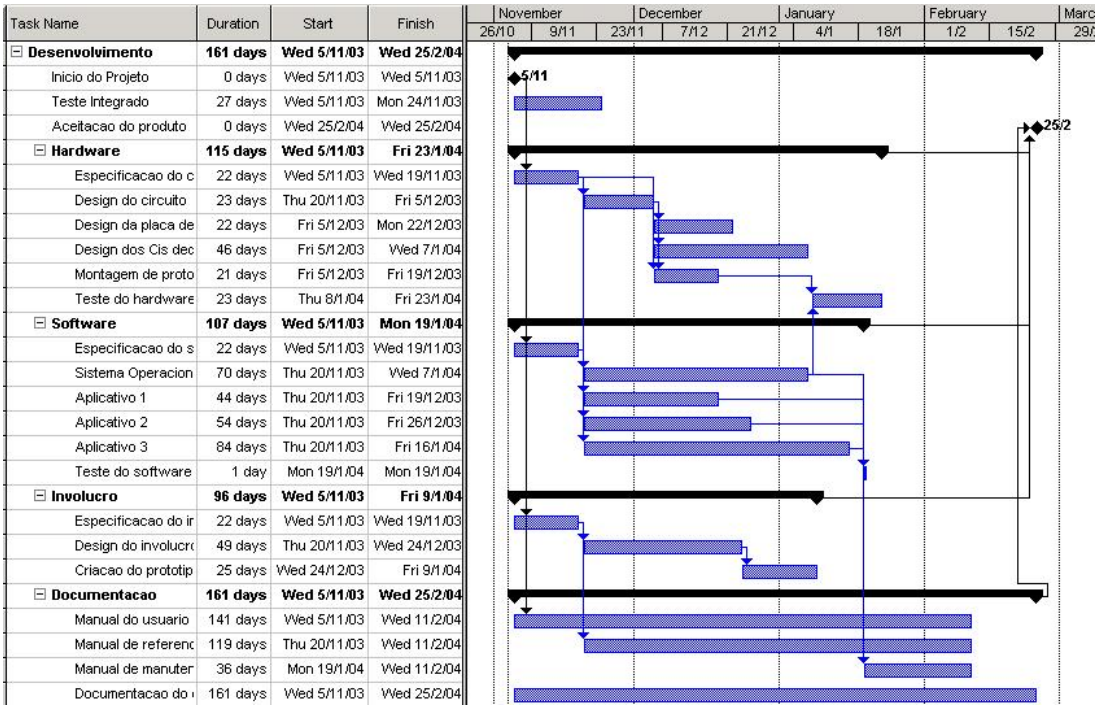

Figura 2.5: Exemplo de diagrama de rede.

que permitem o monitoramento do progresso, como início e o fim do projeto, por exemplo.

Após a elaboração do diagrama, a próxima atividade é atribuir a quantidade necessária de recursos (pessoas e materiais), seus custos e prazos, para cada atividade. No caso da mão-de-obra, é necessário indicar o perfil do profissional considerado na estimativa, pois isso tem um impacto direto no tempo e no orçamento.

Concluído o diagrama de rede, o próximo passo é montar o cronograma inicial, que é a primeira versão do cronograma do projeto. Esta versão inicial do cronograma ainda não considera as limitações reais de recursos, como quantidade de pessoas disponíveis para executar as tarefas. Logo, o resultado é a produção ideal, ou seja, todos as atividades que podem ser executadas em paralelo estarão descritas como tal. No momento em que restrições de recursos são adicionadas ao diagrama de rede, algumas das atividades que antes estavam descritas em paralelo poderão agora estar descritas em série.

Com base na seqüência de execução dos trabalhos, o próximo passo é identifi-

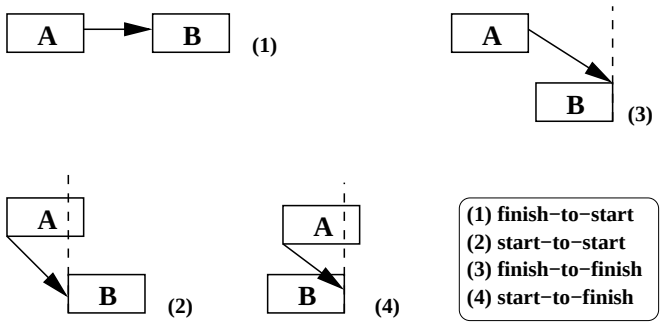

Figura 2.6: Representação gráfica das relações de dependência (Adaptado de Gomi (2002b)).

car as folgas e o caminho crítico. O caminho crítico é o caminho onde não existe folga entre as atividades e qualquer atraso em uma das atividades deste caminho provoca atraso do projeto.

Concluída a elaboração do cronograma inicial, o próximo passo é distribuir os recursos existentes entre os pacotes de trabalho. Eventualmente os recursos serão insuficientes para executar todas as tarefas que poderiam ser trabalhadas em paralelo. Para solucionar este problema, deslocam-se no tempo as tarefas que possuem flutuação, para um momento em que o recurso necessário esteja disponível. O ideal é evitar picos no cronograma, onde muitas tarefas estejam sendo executadas simultaneamente, distribuindo-se os recursos e sempre priorizando as atividades do caminho crítico, em detrimento daquelas que têm folga. No cronograma resultante, o caminho crítico poderá mudar, assim como as folgas das atividades.

Para se obter o planejamento global do projeto são executados um conjunto de processos que serão vistos na próxima seção.

2.4.3 Processos de planejamento e elementos do plano

Os processos inerentes a qualquer projeto na fase de planejamento, são os indicados na figura 2.7.

Onde, cada processo é definido como (PMI, 2000):

i. planejamento do escopo: desenvolve o escopo do trabalho a ser executado

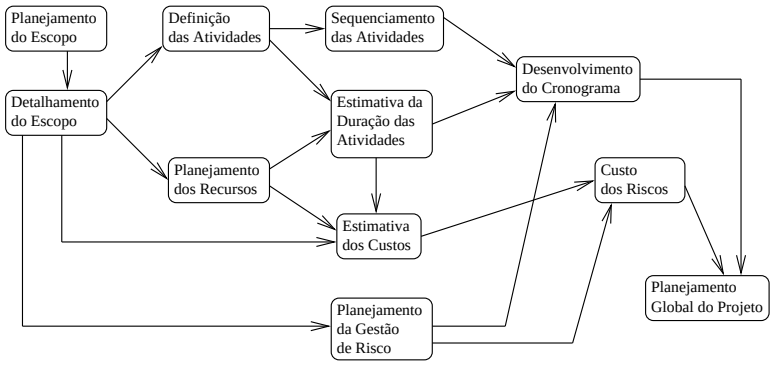

Figura 2.7: Relacionamento entre os processos de planejamento.

como a base para futuras decisões no projeto;

ii. *detalhamento do escopo*: sub-divide o projeto em pequenas partes, componentes mais fáceis de serem gerenciados. Durante este processo é construído a estrutura analítica do projeto (*Work Breakdown Structure - WBS*);

iii. *definição das atividades*: identifica as atividades que devem ser executadas para produzir as várias entregas do projeto;

iv. *planejamento dos recursos*: determina que recursos (pessoas, equipamentos, materiais, etc) e que quantidades de cada deverá ser utilizado para executar as atividades do projeto.

v. *sequenciamento das atividades*: identifica e documenta as dependências de interação entre as atividades.

vi. *estimativa da duração das atividades*: estima o número de períodos de trabalho que serão necessários para completar as atividades individualmente. Esforço é o tempo necessário para realizar uma atividade. O esforço é geralmente medido em horas. Serve, entre outras coisas, para medir a duração de uma atividade, calculada em dias. O cálculo da duração das atividades é a quantidade de esforço dividido pela quantidade de horas trabalhadas por dia;

vii. *estimativa dos custos*: desenvolve uma aproximação (estimativa) de custos dos recursos necessários para completar as atividades do projeto. Essa estimativa deverá ser analisada e aprovada e, então, ser considerada a base

para comparações (*baseline*) ao longo da execução do projeto. A estimativa de custos será realizada na forma *bottom-up*, através da soma de todos os custos previstos para as atividades do projeto;

viii. *planejamento da gestão de risco*: determina como agir e como planejar a administração dos riscos do projeto;

ix. *desenvolvimento do cronograma*: analisa as sequências de atividades, duração de atividades e recursos necessários para criar o cronograma do projeto, determinando as datas de início e fim de cada uma das atividades, além das folgas e caminho crítico;

x. *custo dos riscos*: determina um custo adicional para cada atividade que apresenta um risco significativo ao projeto;

xi. *planejamento global do projeto*: pega os resultados dos outros processos de planejamento e coloca-os em um documento de forma consistente e coerente.

Estes processos estão sujeitos a várias iterações até completar o processo de planejamento. Alguns processos da fase de planejamento estão fortemente ligados e requerem ser executados essencialmente na mesma ordem na maioria dos projetos.

O objetivo de todas estas atividades é a elaboração do planejamento global do projeto, ou simplesmente planejamento do projeto. Este planejamento contém as seguintes informações:

i. cronograma: atividades com datas de início e de término, folgas de cada tarefa e análise do caminho crítico;

ii. análises de custo e fluxo de caixa, e;

iii. plano de necessidade de recursos humanos e materiais.

2.5 A sistemática do planejamento e controle de projetos

Segundo Srivastava, Kambhampati e Do (2001), atualmente o planejamento e controle de projetos de larga escala são feitos utilizando uma ferramenta de gestão

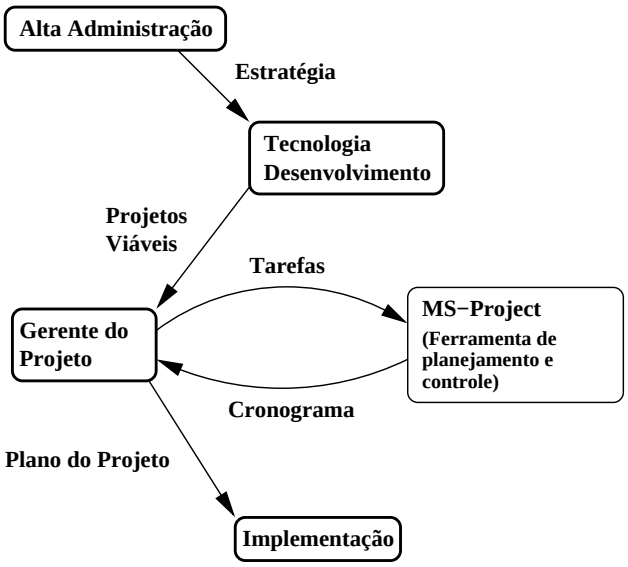

Figura 2.8: Relação entre as fases de planejamento e escalonamento (Adaptado de Srivastava, Kambhampati e Do (2001)).

de projetos, tais como o *Microsoft Project* (MS-Project) ou *Artemis*.

Vem da alta administração da empresa a idéia (estratégia) para lançar um novo produto no mercado. Esta idéia é passada à equipe de desenvolvimento que analisa a viabilidade técnica da idéia. Os projetos aprovados são então passados aos gerentes de projetos que são responsáveis pela elaboração do planejamento do projeto. Esse planejamento pode ser feito com o auxílio de uma ferramenta como o *MS-Project* (figura 2.8).

A elaboração do conjunto de atividades para um determinado projeto é realizada de duas maneiras: ou iniciando um plano totalmente novo ou utilizando algum tipo de modelo já existente.

Existem WBS's padrões para determinados tipos de projetos, que podem servir como ponto de partida para a criação da WBS específica do projeto. Por exemplo, existem metodologias de desenvolvimento de software, como o *Unified Process*, que prevêem um conjunto padrão de atividades que deveriam existir em todos os projetos de desenvolvimento de software (PRESSMAN, 2002).

Na maioria das vezes, com o desenrolar dos projetos, as empresas começam a manter o cronograma dos projetos já executados para futura consulta e reutilização em projetos similares.

2.6 Considerações sobre planejamento em Gestão de Projetos

Independente da maneira como é definido o conjunto de atividades para um determinado projeto, seja através de modelos ou iniciando um plano totalmente novo, em ambos os casos tem-se os seguintes problemas:

i. *falta de conhecimento*: gerentes de projetos nem sempre possuem conhecimento sobre o domínio do problema ou sobre como planejar um projeto específico;

ii. *complexidade do projeto*: muitos projetos, dado a sua complexidade, são naturalmente difíceis de serem planejados, mesmo com o auxílio de modelos que tentam retratar a experiência passada do gerente do projeto ou da empresa, e;

iii. *falta de alternativas*: dado a forma como hoje em dia é construído um plano, o gerente de projetos tende a visualizar sempre *uma única solução* para determinado conjunto de problemas.

3 Representação de Conhecimento de Empresas

Neste capítulo são descritos os conceitos e definições de Sistemas Baseados em Conhecimento, enfocando a importância da utilização de ontologias pré-existentes no seu desenvolvimento. Além disto, é apresentada a ontologia TOVE, que será utilizada no desenvolvimento do modelo para representar o conhecimento sobre projetos de empresas.

3.1 Sistemas Baseados em Conhecimento

Nos últimos anos, tem havido um crescente interesse em descrever sistemas complexos de processamento de informação em termos do conhecimento por eles manipulado. A construção de protótipos de sistemas especialistas em âmbito acadêmico tem dado lugar ao desenvolvimento de Sistemas Baseados em Conhecimento (SBCs) aplicados a negócios (FALBO, 1998).

Nilsson (1998) define Sistemas Baseados em Conhecimento como sendo programas que raciocinam sobre uma extensa base de conhecimento com a finalidade de resolver problemas. A importância da construção de Sistemas Baseados em Conhecimento para as diversas organizações encontra-se na capacidade desses sistemas de preservar, aproveitar e fazer uso do conhecimento dos membros da organização no processo de tomada de decisões. Sistemas Baseados em Conhecimento têm sido aplicados nos mais variados ramos, como transporte, medicina, agricultura e indústria. Alguns exemplos de aplicação de sistemas baseados em conhecimento podem ser encontrados em (PANKASKIE; WAGNER, 1997) que descreve o sistema CLEM utilizado pela Universidade de Pittsburgh para monitorar eventos clínicos e em (HEISSERMAN; CALLAHAN; MATTIKALI, 2000) que

descreve um sistema associado a um sistema de CAD (*Computer Aided Design*) para projetar tubulações em aviões comerciais da Boeing.

Sistemas Baseados em Conhecimento são compostos basicamente de três entidades (REZENDE, 2003):

i. base de conhecimento, que é um formalismo processável computacionalmente, onde é representado todo o conhecimento sobre um determinado domínio;

ii. motor de inferência, implementado por meio de um método ou estratégia de resolução para o formalismo escolhido para a representação do conhecimento, e;

iii. interface com o usuário, que é responsável pela obtenção de informações junto ao usuário, além da apresentação de resultados.

Vale a pena realçar que não é intenção discutir o processo de desenvolvimento de Sistemas Baseados em Conhecimento como um todo neste capítulo, mas apenas seus aspectos relacionados à construção da base de conhecimento. Para uma discussão mais ampla sobre o processo de desenvolvimento, métodos, técnicas e ferramentas para a construção de Sistemas Baseados em Conhecimento, vide (JACKSON, 1998), (LIEBOWITZ, 1999) e (O'LEARY, 1998).

3.2 Base de Conhecimento

Durante a construção de Sistemas Baseados em Conhecimento, a primeira tarefa é a definição de uma representação do mundo, coerente com o senso comum e suficientemente precisa para permitir que a ferramenta implementada apresente um comportamento interessante tomando como base essa representação (MCCARTHY; HAYES, 1969). Além disto, é necessário que a representação adotada seja compreensível para o mecanismo de inferência adotado.

O formalismo escolhido deve ser suficientemente expressivo (mas não mais do que o suficiente) para permitir a representação do conhecimento a respeito do domínio escolhido de maneira completa e eficiente.

A construção da base de conhecimento é, em geral, a tarefa mais cara e demorada de um projeto de sistema baseado em conhecimento. Uma alternativa para minimizar o esforço realizado pelo desenvolvedor da base de conhecimento, é utilizar meios que permitam a reutilização de conhecimento.

Ontologias servem como ferramenta para organização, reuso e disseminação de conhecimento, facilitando a construção de novos Sistemas Baseados em Conhecimento (FREITAS, 2003).

Apesar da palavra "ontologia" denotar uma teoria sobre a natureza do ser ou existência, em Inteligência Artificial ela é interpretada como o conjunto de entidades com suas relações, restrições, axiomas e vocabulário (FREITAS, 2003). Uma ontologia define um domínio, ou, mais formalmente, especifica uma conceitualização sobre o domínio em consideração (GRUBER, 1995).

Uma ontologia é uma descrição formal de entidades e propriedades, compartilhando uma terminologia para os objetos de interesse do domínio e definindo o significado de cada termo (GRüNINGER; FOX, 1994).

Entre os benefícios do uso de ontologias, tem-se (FREITAS, 2003):

i. oportunidade para os desenvolvedores reutilizarem ontologias e bases de conhecimento, mesmo com adaptações e extensões, e;

ii. disponibilização de uma vasta gama de "ontologias de prateleira", prontas para o uso e reuso.

A ontologia TOVE (*TOronto Virtual Enterprise*) (FOX; GRüNINGER, 1998; ENTERPRISE INTEGRATION LABORATORY, 2002) provê um arcabouço que define objetos e relações pertinentes ao domínio empresarial, tais como: atividades, recursos e relação entre atividades e recursos.

De acordo com Grüninger e Fox (1994), a modelagem de empresas é um componente essencial na definição de tarefas e funcionalidades de vários elementos (produto, cliente, recurso, etc) de uma empresa. O objetivo da ontologia TOVE é criar uma representação genérica do conhecimento de uma empresa que pode ser reusado entre uma variedade de empresas.

3.3 A Ontologia Tove

Com o intuito de minimizar o esforço da construção da Base de Conhecimento do sistema tratado neste trabalho, esta seção apresenta um arcabouço para representação de atividades, estados e recursos em uma arquitetura de integração de empresas.

O desenvolvimento da ontologia TOVE é guiado por dois objetivos. O primeiro objetivo é a integração das atividades de gerenciamento da cadeia de suprimentos (*supply chain management* - recebimento de material, manufatura de produtos e distribuição aos consumidores). O segundo objetivo é ligado ao desenvolvimento e execução das empresas, ou seja, formalizar o conhecimento necessário para reengenharia de processos de negócio e a criação de um conjunto de facilidades e aplicações para este conhecimento em uma empresa particular.

Muitos dos objetos e relações descritos na ontologia TOVE não são pertinentes ao domínio de Gestão de Projetos. Nesta seção serão apresentadas apenas as definições dos objetos e relações que serão utilizadas na construção da base de conhecimento do sistema aqui descrito. Uma descrição completa sobre a ontologia TOVE pode ser obtida em (ENTERPRISE INTEGRATION LABORATORY, 2002).

3.3.1 Atividade e estados

Uma atividade é o objeto básico com que os processos e as operações de uma empresa podem ser representados, ou seja, é através do objeto atividade que especifica-se como a visão do mundo muda para a empresa.

O objeto atividade de empresa é definido através da relação com um objeto estado *enabled* e com um objeto estado *caused*.

Um objeto estado *enabled* define o que deve ser verdadeiro no mundo para que a atividade possa ser executada, ou seja, as pré-condições. Um objeto estado *caused* define o que é verdadeiro no mundo após a execução da atividade, ou seja, os efeitos. Na figura 3.1 é possível visualizar uma representação gráfica para o conjunto de atividade e estados.

Existem dois tipos de estados: *terminal* e *não terminal*. O estado não-terminal é definido como uma conjunção de estados terminais. Existem quatro

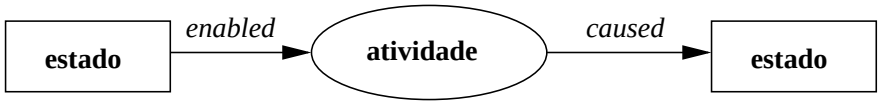

Figura 3.1: Conjunto de atividade e estados.

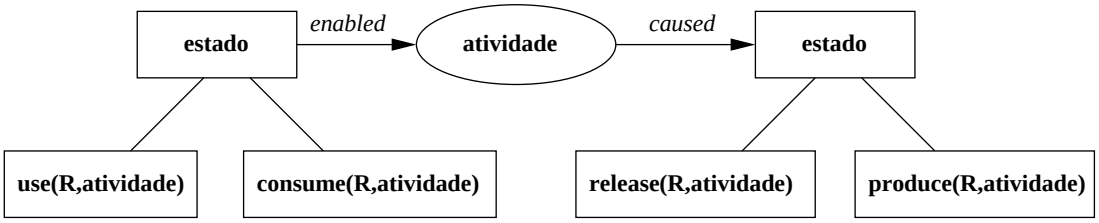

Figura 3.2: Estados terminais e não-terminais.

tipos de estados terminais representados pelos seguintes predicados:

- $use(R, \alpha)$: significa que um recurso R é usado por uma atividade α;
- $consume(R, \alpha)$: significa que um recurso R é consumido por α;
- $release(R, \alpha)$: significa que um recurso R é liberado por α, e;
- $produce(R, \alpha)$: significa que um recurso R é produzido por α.

O estado *enabled* é um estado não-terminal que é definido pela conjunção de estados terminais representados pelos predicados $use(R, \alpha)$ e $consume(R, \alpha)$, enquanto que o estado não-terminal *caused* é definido pela conjunção de estados terminais representados pelos predicados $release(R, \alpha)$ e $produce(R, \alpha)$. Ou seja, percebe-se que ambos os estados, *enabled* e *caused*, são definidos com base nos recursos. Na figura 3.2 é possível visualizar uma representação gráfica da relação dos estados não-terminais com as atividades, e dos estados não-terminais com os terminais.

3.3.2 Pré-condições, efeitos e recursos

Intuitivamente, o recurso é *usado* e *liberado* por uma atividade se nenhuma das propriedades do recurso são mudadas quando uma atividade é terminada com sucesso. O recurso é *consumido* ou *produzido* se alguma das propriedades do

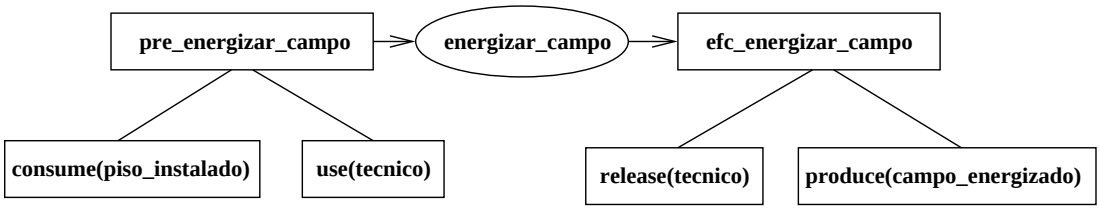

Figura 3.3: Exemplo de conjunto de atividade e estado.

recurso é alterada depois de terminada a atividade. Isto inclui a existência e quantidade de recursos, ou alguma propriedade arbitrária, como cor. Desta forma, $consume(R_1, \alpha_1)$ significa que o recurso R_1 será utilizado pela atividade α_1 e depois da atividade completada não existirá mais. E $produce(R_2, \alpha_1)$ significa que um recurso R_2, que não existia antes da execução da atividade α_1 é criado após a execução da mesma.

O *consumo* e *uso* de recursos são utilizados nos estados *enabled* como pré-condições para a atividade, enquanto que a *liberação* e *criação* são utilizados nos estados *caused*, como resultados da atividade.

Na figura 3.3, $pre\_energizar\_campo$ é o estado não terminal que habilita a atividade $energizar\_campo$, enquanto que $efc\_energizar\_campo$ é o estado não-terminal resultante da execução desta atividade. A conjunção de sub-estados terminais de $pre\_energizar\_campo$ são $consume(piso\_instalado)$ e $use(tecnico)$. A conjunção de sub-estados terminais de $efc\_energizar\_campo$ são $release(tecnico)$ e $produce(campo\_energizado)$.

Isto significa dizer, que para executar a atividade $energizar\_campo$ é necessário *usar* o perfil de um *técnico* e *consumir* o material $piso\_instalado$. Após a execução da atividade $energizar\_campo$ é *liberado* o perfil de um *técnico* e produzido o material $campo\_energizado$.

4 Planejamento em Inteligência Artificial

Neste capítulo é apresentado um tutorial sobre algumas técnicas utilizadas na área de Inteligência Artificial para solucionar problemas de planejamento. Ao final, é comparado a utilidade das técnicas descritas, justificando a escolha do algoritmo utilizado no mecanismo de inferência do sistema desenvolvido neste trabalho.

4.1 O problema de planejamento

Planejamento é o processo que leva ao estabelecimento de um conjunto coordenado de ações, visando determinados objetivos. Em outras palavras, planejar é elaborar um conjunto de ações e colocá-las na ordem correta para se atingir um determinado fim. Para exemplificar, considere o problema do desenvolvimento de um computador portátil (ver seção 2.4.1). Atingir esse objetivo requer a execução de um conjunto de ações (desenvolvimento do hardware, desenvolvimento do software, produção do equipamento desenvolvido, entre outras) na ordem correta, e um gerente de projetos deve gastar algum tempo raciocinando sobre essas ações e sua ordem apropriada para projetar o plano que irá alcançar o objetivo.

Com a finalidade de atribuir a um sistema computacional a capacidade de raciocinar sobre ações e sua ordem apropriada para alcançar um objetivo, a área de planejamento em IA dedica-se ao estudo de formas de representação dessa classe de problemas, bem como a elaboração de algoritmos para solucioná-los.

Um problema de planejamento em IA consiste em, dado uma descrição do mundo (estado inicial), uma descrição dos objetivos a serem alcançados (estado objetivo) e uma descrição do conjunto de ações que podem ser executadas, encon-

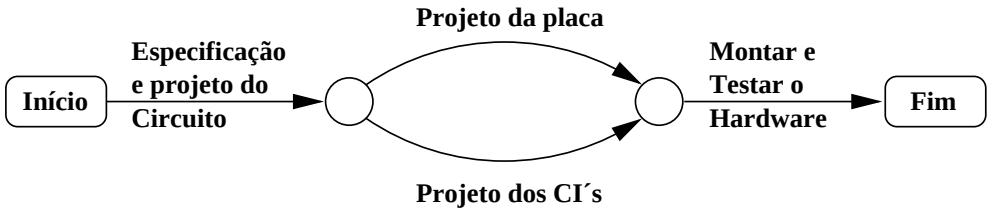

Figura 4.1: Exemplo de um gráfico de plano.

trar um plano - seqüência de ações - que, quando executadas a partir da descrição do estado inicial, conseguem satisfazer a descrição do estado objetivo.

Considere o exemplo onde o objetivo a ser alcançado é o desenvolvimento do hardware de um computador portátil (estado final). As ações que podem ser executadas para alcançar o objetivo são: fazer a especificação do circuito digital, concepção das placas de circuito impresso, desenvolvimento dos circuitos integrados, montagem e teste do hardware. Um plano possível que consegue satisfazer a descrição do estado objetivo é o que aparece na figura 4.1.

Diversos algoritmos tem sido desenvolvidos para resolver os problemas de planejamento. A primeira iniciativa, utilizando provadores de teoremas de primeira ordem, foi o cálculo de situações. Após o advento do cálculo de situações surgiram outras técnicas, baseadas na busca em um grafo de estados ou busca em um espaço de planos parciais (YANG, 1997; WELD, 1994). Ambas as técnicas serão vistas nas próximas seções.

4.2 O Cálculo de Situações

O Cálculo de Situações proposto por McCarthy e Hayes (1969), é um formalismo que modela ações e efeitos, abragendo predicados para *situações*, que descrevem instantes do mundo; *fluentes*, que denotam propriedades do mundo que podem mudar de uma situação para outra; e *ações*, que transformam uma situação em outra.

No Cálculo de Situações é assumida a existência de uma situação inicial, denotada pela constante $s0$, e que o mundo muda de uma situação para outra quando as ações são executadas. As relações entre situações formam a estrutura de uma árvore, onde duas diferentes seqüências de ações levam a diferentes

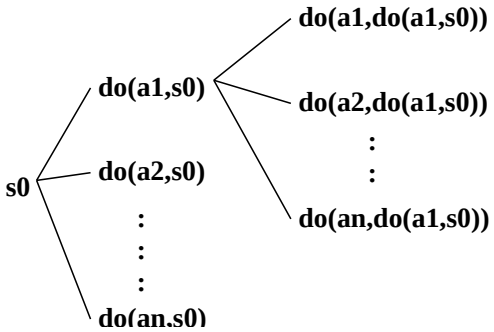

Figura 4.2: Árvore de situações.

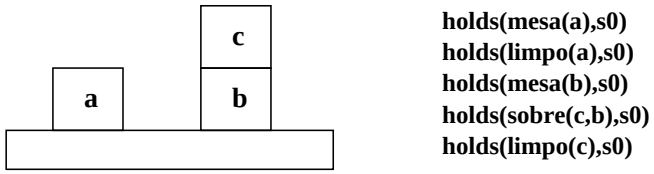

Figura 4.3: Situação inicial do exemplo do mundo dos blocos.

situações. Desta maneira, cada caminho que inicia na situação inicial $s0$ pode ser entendido como um futuro hipotético. A estrutura da árvore do Cálculo de Situações mostra todos os possíveis caminhos que um evento pode desdobrar no mundo. Desta maneira, uma arbitrária seqüência de ações identifica um caminho na árvore de situações (figura 4.2).

O que diferencia uma situação de outra são as propriedades do mundo que podem ser observadas em cada uma das situações e, sendo assim, uma maneira para se descrever uma situação é estabelecendo quais fluentes valem nessa situação (PEREIRA, 2002).

4.2.1 Descrição da situação inicial

No cálculo de situações, a situação inicial de um mundo, é descrita através de um conjunto de sentenças na forma $holds(\beta, \sigma)$, denominadas *sentenças de observação*. O termo σ é um termo *ground* (constante), geralmente representado por $s0$, indicando a situação inicial, enquanto que β é o fluente válido nesta situação (SHANAHAN, 1997). Por exemplo, as sentenças apresentadas na figura 4.3 descrevem a situação inicial do mundo dos blocos, representada na mesma figura.

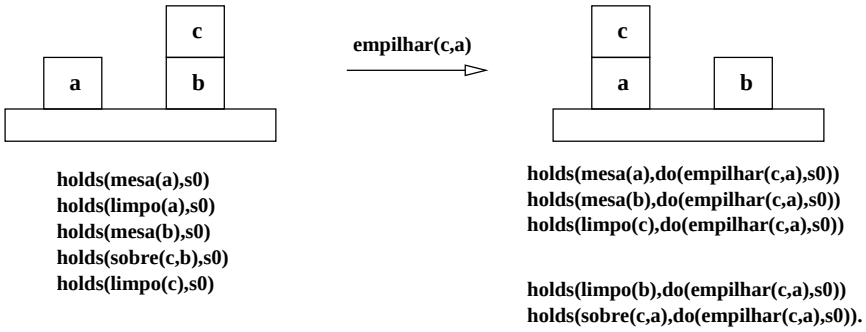

Figura 4.4: Transformação do mundo.

Como em qualquer problema computacional, é necessário abstrair apenas os objetos e relações relevantes ao domínio do problema, pois até mesmo em domínios muito simples, a descrição completa de uma situação real é praticamente impossível.

4.2.2 Descrição das ações do domínio

Enquanto a noção de situação é estática, a noção de ação é dinâmica. De fato, a situação de um mundo só se transforma como conseqüência da ocorrência de uma ação. Algo que é executado de maneira intencional por um agente.

Considerando-se que no mundo haja um único agente, que as ações desse agente sejam determinísticas e que não ocorram eventos inesperados, uma ação α pode ser modelada como uma função $do : \alpha \times \beta \rightarrow \beta$, onde o domínio β denota a situação do mundo antes de α ser executada e o contradomínio β denota a situação do mundo após a sua execução (GENESERETH; NILSSON, 1987). Sempre que uma ação é executada, algumas propriedades do mundo que eram falsas tornam-se verdadeiras, enquanto outras que eram verdadeiras tornam-se falsas. Desta forma, o mundo pode persistir numa determinada situação até que uma ação seja executada e altere uma de suas propriedades (fluentes) (figura 4.4).

No exemplo da figura 4.4, os fluentes $mesa(a)$, $mesa(b)$ e $limpo(c)$ permaneceram com os seus valores verdade inalterados, enquanto que, os fluentes $limpo(b)$ e $sobre(c, a)$ tornaram-se verdadeiros e os fluentes $limpo(a)$ e $sobre(c, b)$ tornaram-se falsos.

As relações do tipo *ação* $\rightarrow$ *efeito*, no Cálculo de Situações, são representadas

através de sentenças, denominadas *axiomas de efeito*, que descrevem como as ações de um agente afetam os valores dos fluentes do domínio. Um axioma de efeito tem a forma $holds(\beta, do(\alpha, \sigma))$ e estabelece que o fluente β é um efeito da execução da ação α na situação σ (SHANAHAN, 1997). Por exemplo, o conjunto de axiomas a seguir descreve como as ações afetam os valores dos fluentes do domínio do mundo dos blocos.

holds(sobre(X,Y),do(empilhar(X,Y),S))
holds(limpo(Y),do(desempilhar(X,Y),S))
holds(mesa(X),do(desempilhar(X,Y),S))
holds(limpo(Y),do(mover(X,Y,Z),S))
holds(sobre(X,Z),do(mover(X,Y,Z),S))

Tão importante quanto descrever os efeitos causados pela ocorrência de uma ação, é estabelecer em que circunstâncias ela pode ocorrer, ou seja, que pré-condições devem estar satisfeitas para que a ação possa ser executada numa determinada situação (PEREIRA, 2002).

No cálculo de situações, as pré-condições de uma ação são estabelecidas através de sentenças na forma $possible(\alpha, \sigma) \leftarrow holds(\beta_1, \sigma) \wedge \cdots \wedge holds(\beta_n, \sigma)$, denominadas *axiomas de pré-condições*, que estabelecem que é possível executar a ação α na situação σ se suas pré-condições $\beta_1, \cdots, \beta_n$ estão satisfeitas nessa situação. Por exemplo, os axiomas a seguir descrevem as pré-condições para as ações *empilhar*, *desempilhar* e *mover*, respectivamente.

possible(empilhar(X,Y),S) ← holds(limpo(X),S) ∧
holds(limpo(Y),S) ∧
holds(mesa(X),S) ∧
¬ equal(X,Y).

possible(desempilhar(X,Y),S) ← holds(limpo(X),S) ∧
holds(sobre(X,Y),S).

possible(mover(X,Y,Z),S) ← holds(limpo(X),S) ∧
holds(limpo(Z),S) ∧

holds(sobre(X,Y),S) ∧
¬ equal(X,Y).

4.2.3 Problema do quadro

Além de descrever o que muda, é necessário descrever também aquilo que permanece inalterado. Por isso, a necessidade de *axiomas de quadro* (SHANAHAN, 1997). Por exemplo, para a ação $mover(X, Y, Z)$[1] é necessário codificar um axioma de quadro para cada fluente que permanece inalterado após a execução da ação $mover(X, Y, Z)$:

holds(sobre(V,W),do(mover(X,Y,Z),S)) ←
possible(mover(X,Y,Z),S) ∧
holds(sobre(V,W),S) ∧
¬ equal(V,X).

holds(mesa(V),do(mover(X,Y,Z),S)) ←
possible(mover(X,Y,Z),S) ∧
holds(mesa(V),S).

O problema do quadro[2] é a necessidade de se manter uma enorme quantidade de axiomas de quadro para garantir a persistência dos fluentes que não são afetados por uma ação. Por exemplo, num domínio com m ações e n fluentes, são necessários (m x n) axiomas de quadro.

Para resolver o problema do quadro (SHANAHAN, 1997) no cálculo de situações, é necessário substituir todos os axiomas de quadro por um único axioma genérico, independente de domínio, denominado *axioma de quadro universal*. Esse axioma estabelece que os fluentes persistem, a menos que sejam afetados pela execução de uma ação.

$$holds(\beta, do(\alpha, \sigma)) \leftarrow possible(\alpha, \sigma) \wedge holds(\beta, \sigma) \wedge \neg affects(\alpha, \beta) \qquad (4.1)$$

[1] A ação $mover(X, Y, Z)$ significa que um objeto qualquer X será movido de Y para Z
[2] Do inglês *Frame Problem*

O predicado $affects(\alpha, \beta)$ descreve que ações afetam determinados fluentes.

4.2.4 Planejamento no cálculo de situações

De acordo com Lin e Reiter (1997), para utilizar o cálculo de situações como ferramenta para solucionar problemas de planejamento, basta codificar os axiomas do cálculo de situações em um programa lógico, utilizando a linguagem PROLOG (STERLING; SHAPIRO, 1994).

Na linguagem PROLOG, dados um programa lógico e uma cláusula objetivo, o interpretador tem como meta descobrir um ramo fechado da árvore de busca e uma substituição para as variáveis da cláusula objetivo. Quando isto ocorre o interpretador apresenta esta substituição como a resposta à questão apresentada pela cláusula objetivo ao programa lógico. Internamente, o interpretador pesquisa os ramos da árvore fazendo uma busca em profundidade, dando prioridade aos ramos mais à esquerda. Isto é obtido considerando as cláusulas na ordem em que elas aparecem no programa lógico (DOETS, 1994).

4.2.5 Restrições do planejamento no cálculo de situações

O uso do cálculo de situações, ou seja, o uso do paradigma de programação declarativo, para solucionar problemas de planejamento em IA, fornece um ambiente de programação com alto nível de abstração. O desenvolvedor precisa preocupar-se apenas com a codificação do conhecimento do domínio do problema, fazendo uso dos axiomas do cálculo de situações, sem se preocupar em como implementar o mecanismo de inferência. Esta característica implica na redução da complexidade do desenvolvimento do sistema de planejamento, consequentemente reduzindo o tempo e o custo de desenvolvimento. Porém, é importante salientar, que o cálculo de situações possui sérias limitações quanto ao desempenho computacional e à representação de ações.

O principal problema do paradigma de programação declarativo é a eficiência computacional, tempo[3] e espaço[4], que uma solução adotando este paradigma

[3]Entende-se por eficiência de tempo, o tempo que determinada solução computacional necessita para resolver determinado problema.

[4]Estende-se por eficiência de espaço, a quantidade de memória necessária para solucionar determinado problema.

possui (SEBESTA, 2001).

As limitações que o cálculo de situações possui quanto à representação de ações, importantes para este domínio, são:

i. não representa ações concorrentes, informação do tipo: *"Enquanto a Fernanda trabalha, o Gaspar dorme"* não é possível de ser representada, e;

ii. não representa ações com duração, todas as ações são consideradas atômicas.

Em ambos os casos existem trabalhos propondo extensões para o Cálculo de Situações. Baier e Pinto (1998) propõem um arcabouço que representa tanto ações concorrentes como ações com duração. Miller e Shanahan (1994) propõem um outro arcabouço que trata apenas de ações com duração. Lin e Shoham (1992) e Alferes, Li e Pereira (1994) propõem arcabouços distintos que permite o Cálculo de Situações representar ações com duração. Entretanto, a utilização e implementação destes arcabouços em problemas de domínios reais não se mostrou trivial.

4.3 Representação Strips

A representação STRIPS (STanford Research Institute Problem Solver) foi proposta por Fikes e Nilsson (1971), no início da década de 1970, como uma alternativa ao Cálculo de Situações. Desde então, essa representação tem sido amplamente utilizada na descrição de ações nos sistemas de planejamento.

Neste formalismo, um estado é representado por um conjunto de literais que denotam as propriedades (fluentes) que valem no mundo e uma ação é representada por um operador que transforma um determinado estado em outro, através da adição ou remoção de literais no conjunto que representa este estado.

4.3.1 Operador Strips

Um operador STRIPS *Op* é uma 4-upla (NILSSON, 1998; RUSSEL; NORVIG, 2003):

$$Op = \langle N, P, A, D \rangle \tag{4.2}$$

onde,

- N é uma descrição da ação, ou seja, o nome da ação;
- P é uma lista de literais positivos, chamados de pré-condições da ação;
- A é uma lista de literais, chamados de efeitos positivos da ação;
- D é uma lista de literais, chamados de efeitos negativos da ação.

Dado um sistema de planejamento utilizando operadores STRIPS, define-se um plano como sendo uma seqüência finita destes operadores. Cada plano $\Sigma = (\alpha_1, \ldots, \alpha_N)$ define uma seqüência de modelos do mundo $M_0, M_1, \ldots, M_N$, onde M_0 é o modelo inicial do mundo e os demais modelos são definidos como:

$$M_i = (M_{i-1} \setminus D_{\alpha_i}) \cup A_{\alpha_i} \qquad (i = 1, \ldots, N) \tag{4.3}$$

Ou seja, a descrição do mundo em M_i é o que havia em M_{i-1} exceto os literais excluídos pela ação (D_{α_i}) mais os literais adicionados pela ação (A_{α_i}).

Quando uma ação é executada ela muda a descrição do mundo. Todos os literais contidos na lista de efeitos positivos são adicionados na descrição do estado, enquanto que todos os literais contidos na lista de efeitos negativos são removidos (LIFSCHITZ, 1990).

Para produzir a descrição do estado resultante da aplicação da ação, deve-se primeiro remover do estado anterior todos os literais contidos em D então, adicionar todos os literais contidos em A. Todos os literais não mencionados em D continuam válidos no estado resultante. Esta característica, chamada de STRIPS *assumption*, permite eliminar o problema do quadro (NILSSON, 1998).

Para a execução da ação α_i na descrição do mundo M_{i-1} é necessário que as pré-condições (P_{α_i}) de α_i sejam válidas em M_{i-1}:

$$M_{i-1} \models P_{\alpha_i} \qquad (i = 1, \ldots, N) \tag{4.4}$$

Não existe variável explícita do estado do mundo (situação). Tudo o que for declarado na pré-condição do operador se refere à situação imediatamente antes da ação a ser executada. E tudo o que for declarado no efeito se refere à situação imediatamente depois da ação a ser executada (RUSSEL; NORVIG, 2003).

A representação STRIPS descreve o estado inicial do mundo com um conjunto completo de literais *ground* (RUSSEL; NORVIG, 2003; WELD, 1994). Por exemplo, para o exemplo da figura 4.3 uma possível representação do estado inicial seria: $[mesa(a), mesa(b), limpo(a), sobre(c, b), limpo(c)]$. A mesma regra aplica-se a representação do estado final (objetivo).

Para que a descrição do estado inicial e do estado final possam ser consideradas como completas, todos os literais *grounds* não representadas são consideradas como falsos[5] (WELD, 1994).

Após realizado o planejamento, o algoritmo deve retornar uma descrição do tipo:

$$\Sigma = (mover(c, a, mesa), mover(b, mesa, c), mover(a, mesa, b)) \tag{4.5}$$

4.3.2 Planejamento baseado em estados do mundo

Um caminho simples para construir um planejador é transformar o problema de planejamento em um problema de pesquisa em um espaço de estados do mundo (situação), representado por um grafo (figura 4.5). Nessa figura, cada vértice do grafo representa um estado do mundo e cada aresta uma ação. O plano é o caminho a partir do estado inicial ao estado objetivo (WELD, 1994).

A vantagem de se utilizar um grafo que represente estados do mundo é a possibilidade da utilização de algoritmos de busca para encontrar a solução do problema de planejamento.

[5]Premissa do Mundo Fechado (REITER, 1980)

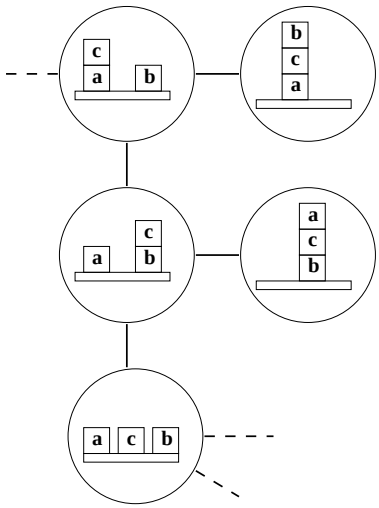

Figura 4.5: Espaço de estados para o exemplo do mundo dos blocos.

4.3.2.1 Planejamento regressivo e progressivo

Basicamente o funcionamento do planejamento progressivo busca para frente (*forward*) começando pelo estado inicial e a cada iteração verificando se o estado corrente é o estado objetivo (que contêm os objetivos do agente). Se não for o estado objetivo, o algoritmo aplica ao estado corrente uma nova ação (a mais apropriada) e anexa ao final do plano. Se o algoritmo não encontrar um conjunto de passos de ações que aplicado ao estado inicial chegue ao estado objetivo, o algoritmo retorna falha.

Contrário ao planejamento progressivo, os algoritmos regressivos buscam para trás (*backward*) partindo do estado objetivo e tentando alcançar o estado inicial, adicionando a nova ação no início do plano.

Um fator positivo dos algoritmos regressivos em relação aos algoritmos progressivos é que os estados que são objetivos tipicamente contêm poucos literais com poucos operadores aplicáveis, tornando o fator de ramificação dos algoritmos regressivos menor do que o fator de ramificação dos algoritmos progressivos, fazendo com que a eficiência de um algoritmo regressivo seja bem maior do que a eficiência de um algoritmo progressivo (figura 4.6).

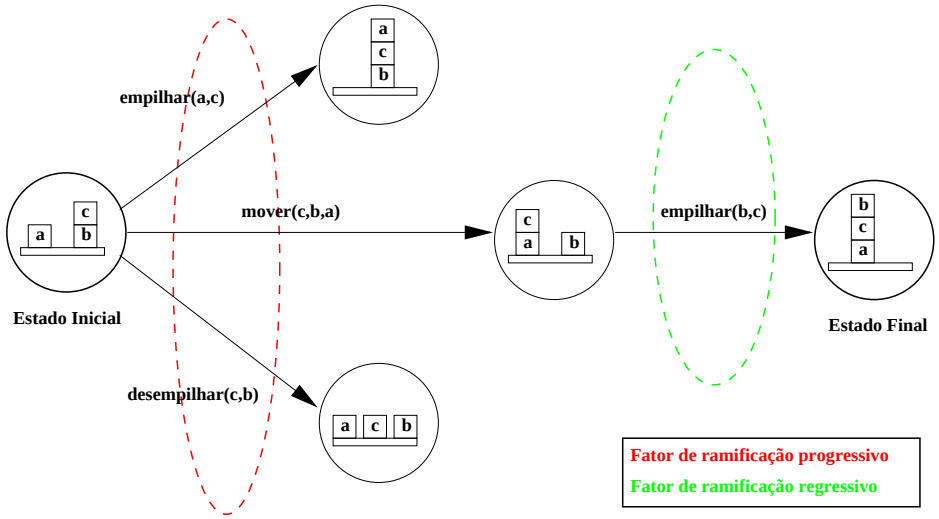

Figura 4.6: Fator de ramificação dos algoritmos progressivos versus algoritmos regressivos.

4.3.3 Planejamento baseado em espaços de planos

Segundo Weld (1994), o grafo que representa um espaço de planos é composto por vértices que representam um plano parcial e arestas que denotam operações que são adicionadas ao plano refinando ou modificando o plano. O estado inicial representa um plano nulo (sem ações) e o plano final representa todas as ações que devem ser tomadas para que o agente alcance o seu objetivo (RUSSEL; NORVIG, 2003). Trata-se de uma busca por um plano desejado ao invés de uma situação desejada.

Parte-se de um plano inicial (parcial), e aplica-se dois (2) tipos de operadores (operador de refinamento e operador de modificação) até chegar a um plano final (completo). Por exemplo, para o problema representado na figura 4.6 tem-se o grafo da figura 4.7.

Um plano solução é um plano que o agente pode executar e que garante realizar o objetivo, pode ser um plano totalmente instanciado e totalmente ordenado[6], ou um plano totalmente instanciado e parcialmente ordenado[7] (correspondendo a um conjunto de planos totalmente ordenados).

[6] *Total Oder Plan*: representa planos onde todos os passos são totalmente ordenados. Lista simples com todos os passos um atrás do outro.

[7] *Partial Order Plan*: representa planos onde alguns passos são ordenados e outros não.

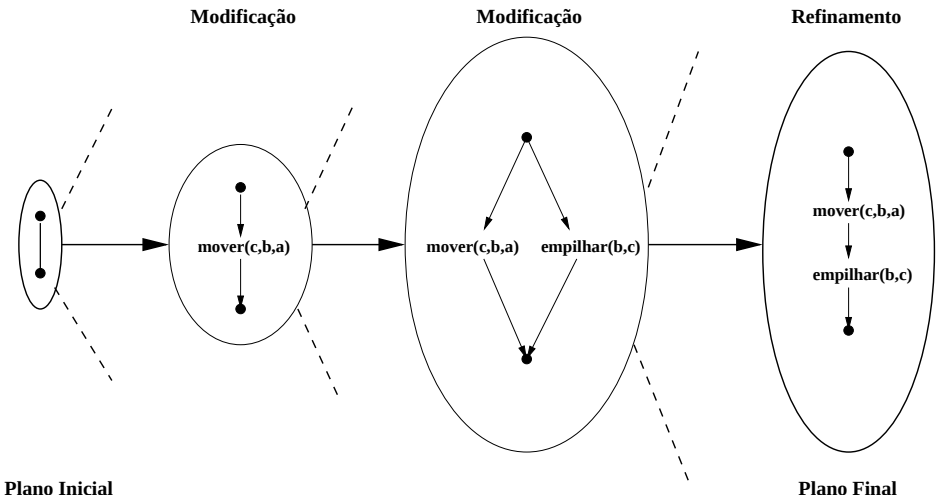

Figura 4.7: Espaço de planos para o exemplo do mundo dos blocos.

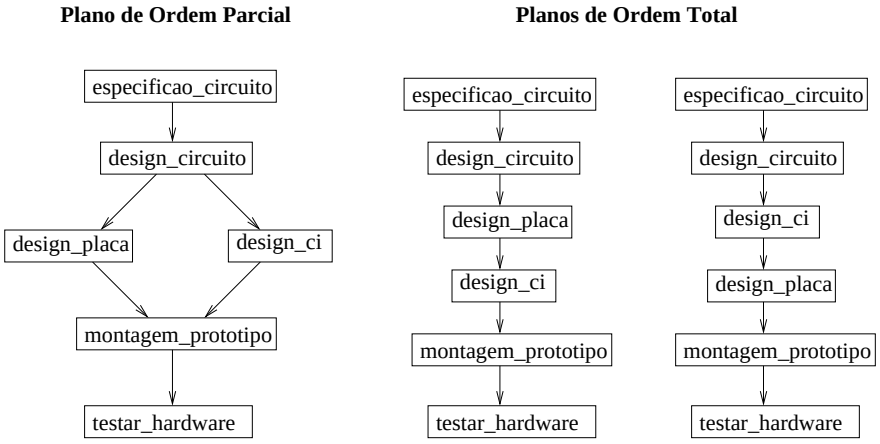

Figura 4.8: Linearização de planos.

A transformação de uma plano parcialmente ordenado em um ou vários planos totalmente ordenados é chamado de linearização de planos. Na figura 4.8 é possível visualizar o plano parcialmente ordenado e os planos totalmente ordenados de um exemplo onde o objetivo é desenvolver o hardware de um computador portátil. Os operadores deste exemplo são os apresentados na tabela 4.1.

4.4 Planejamento de ordem parcial

No planejamento de ordem parcial (WELD, 1994), um plano Σ é representado por uma tripla:

Tabela 4.1: Descrição dos operadores do domínio

Operador	**Pré-condições**	**Efeitos**
especificacao_circuito	requisitos	circuito_especificado
design_circuito	circuito_especificado	circuito_desenhado
design_placa	circuito_desenhado	placa_desenhada
design_ci	circuito_desenhado	ci_desenhado
montagem_prototipo	placa_desenhada ci_desenhado	prototipo_montado
testar_hardware	prototipo_montado	hardware_pronto

$$\Sigma = \langle \omega, \beta, \gamma \rangle \tag{4.6}$$

onde,

- ω é um conjunto de *passos do plano*, correspondendo a um operador/ação do problema;
- β é um conjunto de *restrições de ordenação* sobre ω: $S_i < S_j$, que deve ser lido "S_i *ocorre antes de* S_j", que significa: S_i deve ocorrer antes de S_j mas não necessariamente imediatamente antes, e;
- γ é um conjunto de *vínculos causais*: $(S_i \xrightarrow{Q} S_j)$, que deve ser lido "$S_i$ *realiza* Q *para* S_j". Os vínculos causais servem para registrar o propósito dos passos no plano, onde o propósito de S_i é realizar a pré-condição Q de S_j. Também chamados por alguns autores de intervalos de proteção ou ligações causais.

Por exemplo, sendo $\omega = \{A1, A2, A3\}$ e $\beta = \{A1 < A3, A2 < A3\}$. Estas restrições especificam um plano em que $A3$ é necessariamente a última ação a ser executada, mas não diz com que ação iniciar, se com a $A1$ ou com a $A2$. Note que este conjunto de restrições ordenadas é *consistente* porque pode ser satisfeita.

Vínculos causais são usados para detectar quando uma nova ação é introduzida ao plano e interfere as decisões passadas. Esta ação é chamada de *ameaça*.

Uma ação é uma *ameaça* ao plano, se quando adicionada ao plano ela interfere nas decisões passadas. Mais precisamente, suponha que $\langle \omega, \beta, \gamma \rangle$ é um plano e

($Ap \xrightarrow{Q} Ac$) é um vínculo causal em γ. Sendo At uma ação diferente em ω, sabe-se que At ameaça ($Ap \xrightarrow{Q} Ac$) quando os seguintes critérios são satisfeitos: $\beta \cup \{Ap < At < Ac\}$ é consistente, e; At possui $\neg Q$ como um efeito.

Para prevenir ameaças, o algoritmo de planejamento deve checar cada ameaça e tomar medidas evasivas. Por exemplo, o algoritmo pode adicionar uma restrição de ordenação adicional para assegurar que At é executado antes de Ap. Este método de proteção de ameaça é chamado de *demoção*[8], adicionando a restrição simétrica $Ac < At$ é chamada de *promoção*[9].

A cada iteração o planejador *adicona passos* ou *modifica a ordem* de passos e instanciação de variáveis. O plano final deve ser *completo* (toda pré-condição é realizada por algum passo do plano e uma pré-condição é realizada se e somente se ela é efeito de um passo e nenhum passo intermediário a desfaz) e *consistente* (não há contradições nas ordenação das atividades) (RUSSEL; NORVIG, 2003; WELD, 1994).

4.4.1 Algoritmo Pop

No algoritmo POP (*Partial Order Planning*), apresentado na figura 4.9, a idéia está em identificar uma atividade com pré-condição não satisfeita, introduzir uma atividade cujo efeito é satisfazer esta pré-condição, atualizar o conjunto de restrições de ordenação, atualizar o conjunto de vínculos causais e verificar se há ameaças e corrigir o plano se for o caso (WELD, 1994; BARRET; WELD, 1994; NGUYEN; KAMBHAMPATI, 2001).

O algoritmo POP é correto, completo, sistemático (sem repetição), não determinístico e a inserção de uma ação só é considerada se atender uma pré-condição não atingida.

4.4.2 Exemplo de planejamento de ordem parcial

Nesta seção será apresentado um exemplo cujo o objetivo é ilustrar o funcionamento do algoritmo POP. Neste exemplo, o estado inicial é formado pelo literal *requisitos*, o objetivo é *hardware_pronto* e o conjunto de ações são as descritas

[8]Do inglês *Demotion*
[9]Do inglês *Promotion*

Entrada: POP(Θ, Δ, Γ, $\langle\omega, \beta, \gamma\rangle$), onde:
Θ = conjunto de ações, Δ = estado inicial e Γ = objetivos.
Saída: Plano parcialmente ordenado $\Sigma = \langle\omega, \beta, \gamma\rangle$.
1. **Finalização**: se Γ = vazio então devolve $\langle\omega, \beta, \gamma\rangle$.
2. **Seleção do objetivo**: escolher uma pré-condição Q de A_{need}, onde $A_{need} \in \omega$.
3. **Seleção da ação**:
   escolher uma ação que adiciona Q
   (ou uma nova ação instanciada a partir de Θ, ou uma ação já existente em ω);
   se nenhuma ação adiciona Q então devolve *falha*;
   $\gamma' = \gamma \cup \{A_{add} \stackrel{Q}{\rightarrow} A_{need}\}$; $\beta' = \beta \cup \{A_{add} < A_{need}\}$;
   se A_{add} é uma nova ação instanciada então
   $\gamma' = \gamma \cup \{A_{add}\}$ e $\beta' = \beta \cup \{A_i < A_{add} < A_f\}$
4. **Modificar lista de objetivos**:
   $\Gamma' = \Gamma - \{\langle Q, A_{nedd}\rangle\}$;
   se A_{add} é uma nova ação então para cada $Q_i \in A_{add}$ adicionar $\langle Q_i, A_{add}\rangle$ em Γ'.
5. **Proteção de ligações causais**:
   para toda ação A_t que pode ameaçar uma ligação causal $A_p \stackrel{Q}{\rightarrow} A_c \in \gamma$ escolha uma ordenação consistente, entre:
   adicionar em $\beta' : A_t < A_p$ (Demoção), ou;
   adicionar em $\beta' : A_c < A_t$ (Promoção);
   se nenhuma ordenação é consistente então devolve *falha*.
6. **Invoque** POP($\Theta, \Delta, \Gamma', \langle\omega', \beta', \gamma'\rangle$).

Figura 4.9: Algoritmo POP

na tabela 4.1. Inicialmente, o algoritmo é chamado com o plano vazio ilustrado na figura 4.10 onde $\Gamma = \{\langle hardware\_pronto, af\rangle\}$.

Como o conjunto de pré-requisitos Γ não está vazio, o algoritmo precisa selecionar um deles para ser satisfeito. Portanto, o algoritmo adiciona a ação *testar_hardware* para satisfazer o pré-requisito *hardware_pronto*, como ilustrado na figura 4.11.

Com a adição da ação *testar_hardware*, a estrutura é devidamente modificada

$\Gamma' = \{\langle prototipo\_montado, testar\_hardware\rangle\}$
$\omega' = \{ai, af, testar\_hardware\}$
$\beta' = \{ai < af, ai < testar\_hardware, testar\_hardware < af\}$
$\gamma' = \{testar\_hardware \stackrel{hardware\_pronto}{\rightarrow} af\}$

Esta estrutura será fornecida como entrada para a próxima iteração do algoritmo. Na próxima iteração do algoritmo, será selecionada uma ação que deve satisfazer o pré-requisito *prototipo_montado* (figura 4.12).

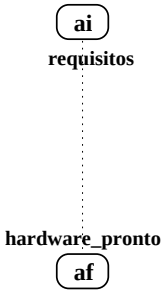

Figura 4.10: Plano parcialmente ordenado inicial.

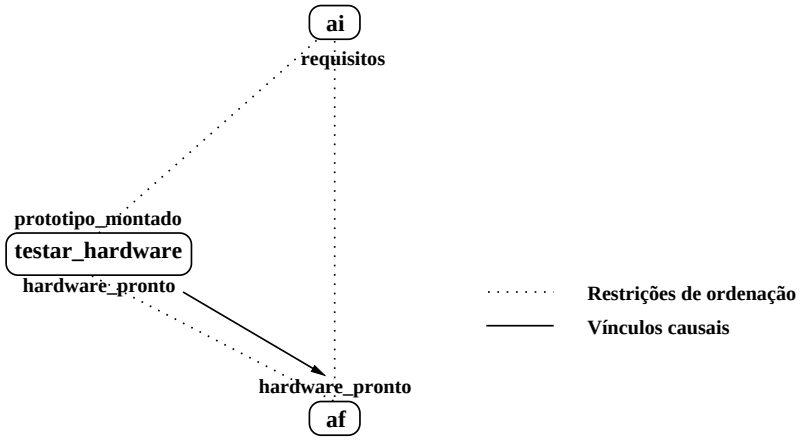

Figura 4.11: Plano parcialmente ordenado com uma ação adicionada.

Com a adição da ação *prototipo_montado*, a estrutura é devidamente modificada para:

$$\Gamma' = \{\langle placa\_desenhada, montagem\_prototipo\rangle, \langle ci\_desenhado, montagem\_prototipo\rangle\}$$
$$\omega' = \{ai, af, testar\_hardware, montagem\_prototipo\}$$
$$\beta' = \{ai < af, ai < testar\_hardware, testar\_hardware < af,$$
$$ai < montagem\_prototipo, montagem\_prototipo < af\}$$
$$\gamma' = \{testar\_hardware \overset{hardware\_pronto}{\rightarrow} af,$$
$$montagem\_prototipo \overset{prototipo\_montado}{\rightarrow} testar\_hardware\}$$

Como não existe nenhuma ação com efeito negativo neste exemplo, o algoritmo não executa nem promoção e nem demoção de nenhuma das ações. O algoritmo é executado iterativamente até que $\Gamma = vazio$. Neste momento ele irá retornar o plano parcialmente ordenado ilustrado na figura 4.13.

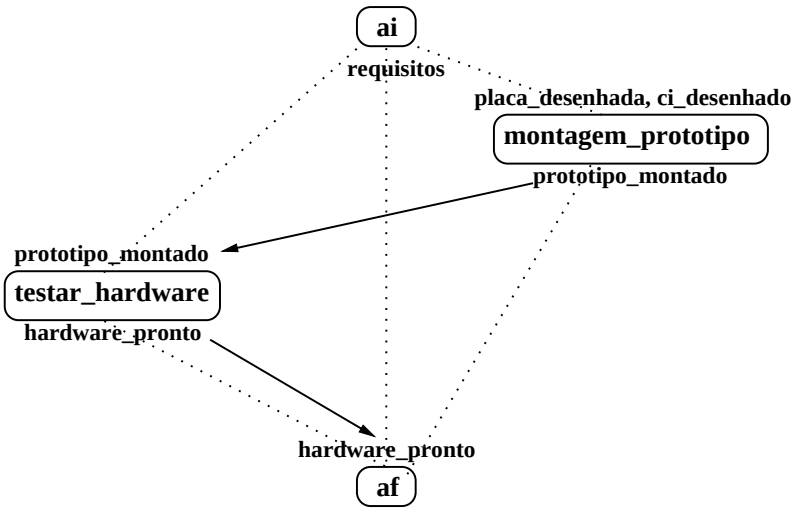

Figura 4.12: Plano parcialmente ordenado com duas ações adicionadas.

4.5 Considerações

Apesar da facilidade para modelar um domínio usando o cálculo de situações, existem várias características do mesmo que prejudicam a implementação de soluções para problemas reais usando tal formalismo, tais como, a incapacidade de representar ações concorrentes, com duração e a baixa eficiência computacional atrelada ao paradigma de programação declarativo. Exemplos de tentativas utilizando o cálculo de situações aplicado ao domínio de gestão de projetos podem ser vistos em (BARTH; GOMI, 2002a; BARTH; GOMI, 2002b). Estes trabalhos apresentam uma extensão do cálculo de situações que permite representar ações que consomem recursos humanos e materiais. O uso dessa extensão do cálculo de situações é exemplificado através de uma aplicação de planejamento de projetos na área de telecomunicações.

Os algoritmos de ordem parcial são os algoritmos, que de maneira geral, tem mostrado maior eficiência computacional para resolver os problemas de planejamento (BARRET; WELD, 1994; NGUYEN; KAMBHAMPATI, 2001). Além disso, para representar ações concorrentes, pode-se utilizar a estrutura de planos parcialmente ordenados, onde as ações que não estão totalmente ordenadas, podem ser manipuladas como ações que são executadas em paralelo (em vermelho na figura 4.14).

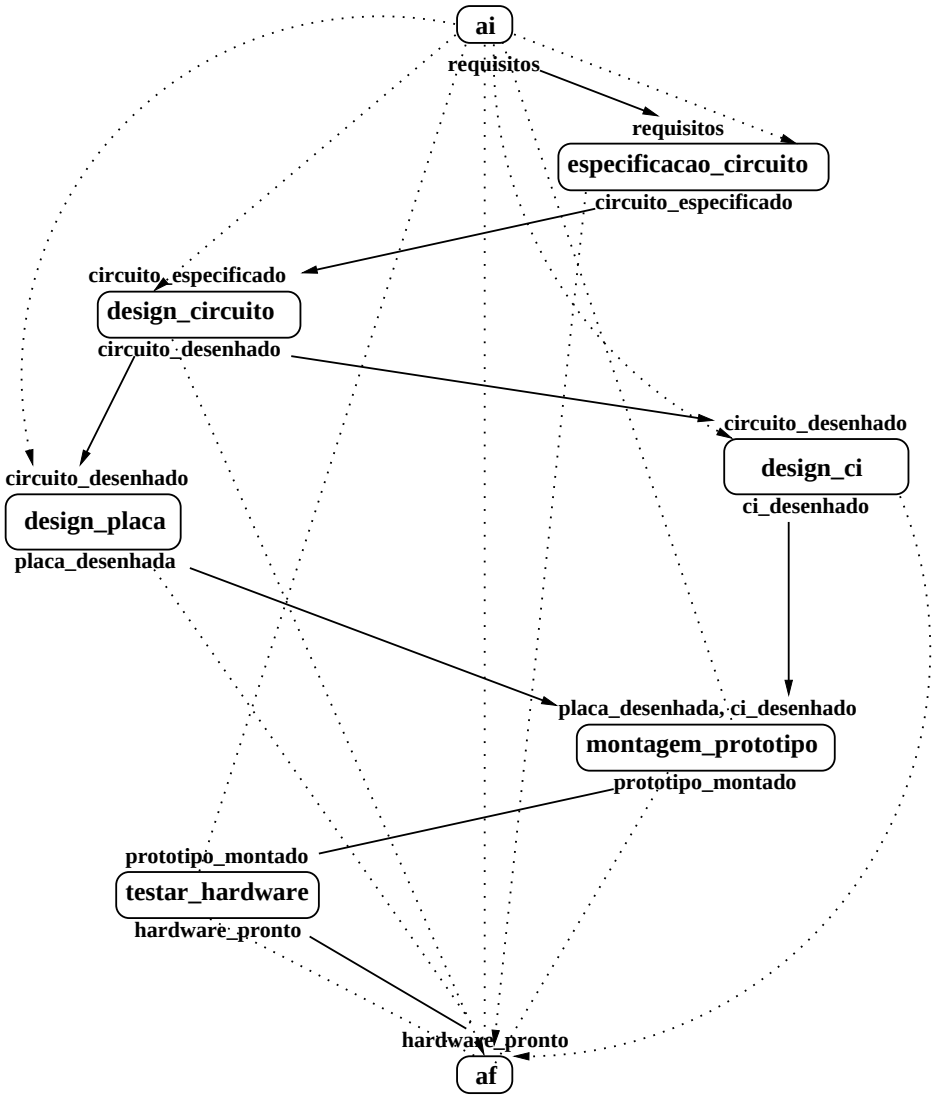

Figura 4.13: Plano parcialmente ordenado final.

Desta meneira, percebeu-se que a utilidade dos algoritmos de ordem parcial é maior do que a utilidade do cálculo de situações para resolver os problemas deste domínio, ou seja, de planejamento em gestão de projetos. Portanto, optou-se pela escolha do algoritmo POP como mecanismo de inferência no sistema que será desenvolvido.

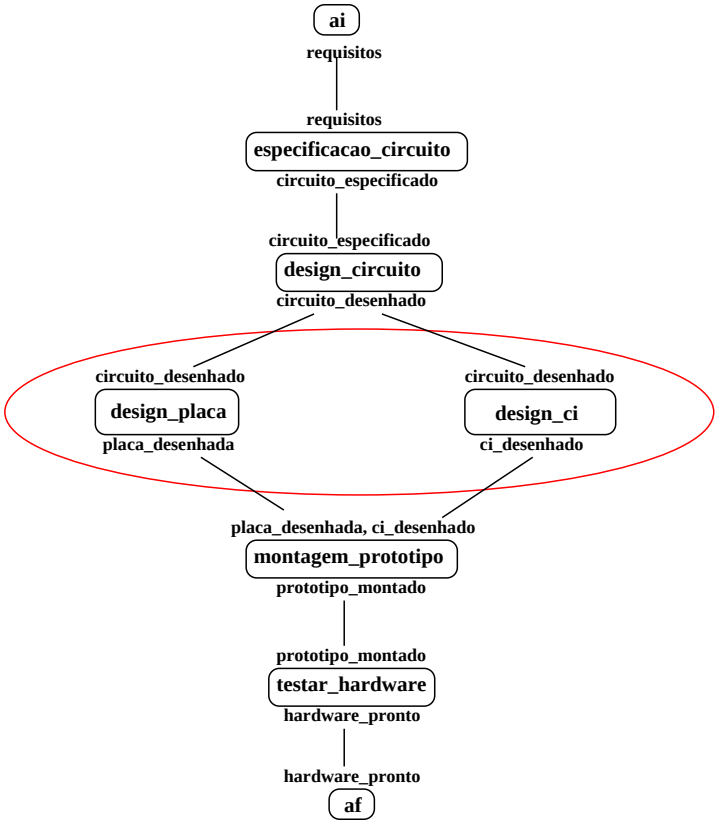

Figura 4.14: Plano parcialmente ordenado.

5 Descrição do Sistema *Atena*

Neste capítulo são apresentadas a descrição do contexto e dos requisitos que guiaram o desenvolvimento do sistema *Atena*[1], da arquitetura do sistema e da maneira como o sistema foi implementado.

5.1 Contexto

Na figura 5.1 é possível visualizar uma adaptação da sistemática de trabalho apresenta por (SRIVASTAVA; KAMBHAMPATI; DO, 2001). Esta nova sistemática de trabalho adiciona o sistema *Atena* como mais uma ferramenta que o gerente de projetos pode utilizar no processo de desenvolvimento de um plano de projeto.

Quando os projetos viáveis são passados aos gerentes de projetos, os mesmos podem fazer uso do sistema *Atena*, da mesma maneira que fazem uso da ferramenta de escalonamento, para obter possíveis soluções alternativas para um determinado problema.

5.2 Requisitos do Sistema

O sistema deve atuar sobre uma base de conhecimento que descreve o conjunto de atividades que uma empresa deve executar no âmbito de um projeto. O sistema recebe como entrada uma descrição da situação atual e uma descrição dos objetivos a serem atingidos e devolve uma ou mais redes de atividades possíveis, que descrevem o conjunto de atividades que devem ser executadas, os recursos necessários (pessoas, máquinas, etc) e as relações de dependência entre as atividades.

[1] O apêndice I apresenta a justificativa do nome *Atena*.

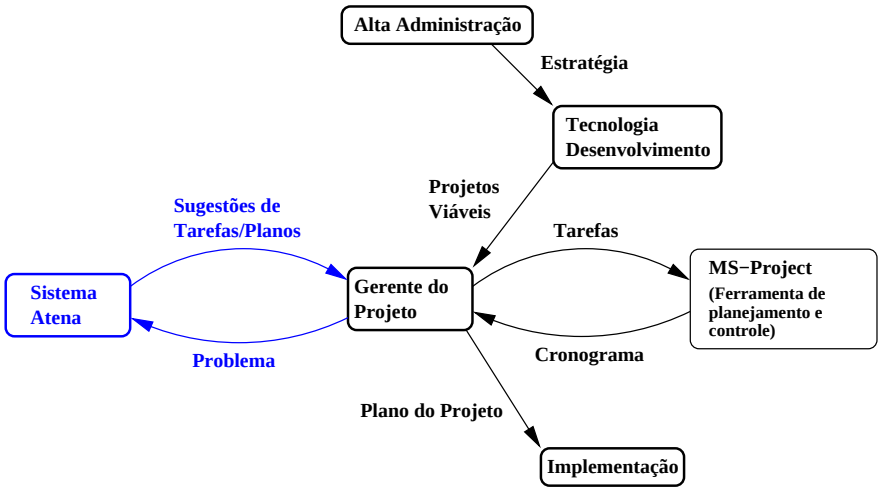

Figura 5.1: Contexto em que está inserido o sistema Atena.

Mais precisamente, as perguntas que o sistema deverá ser capaz de responder são:

i. Dada a descrição de um produto, qual o conjunto de atividades que deverá ser executada para desenvolver o produto?

ii. Quais são os recursos necessários para a execução das atividades?

iii. Qual a seqüência e a estimativa da duração das atividades?

O desenvolvimento deste sistema fundamenta-se em três (3) requisitos:

i. os planos retornados devem ser consistentes com os objetivos fornecidos;

ii. as informações descritas acima devem estar contidas nos planos retornados, e;

iii. o tempo para solução do problema deve ser satisfatório. Segundo Yang (1997), um sistema de planejamento que é capaz de construir um bom plano rapidamente pode ser chamado de um planejador eficiente.

5.3 Arquitetura do Sistema

A arquitetura do sistema é composta por um *núcleo*, por um componente de *controle* e por um componente de *interface com o usuário* (figura 5.2).

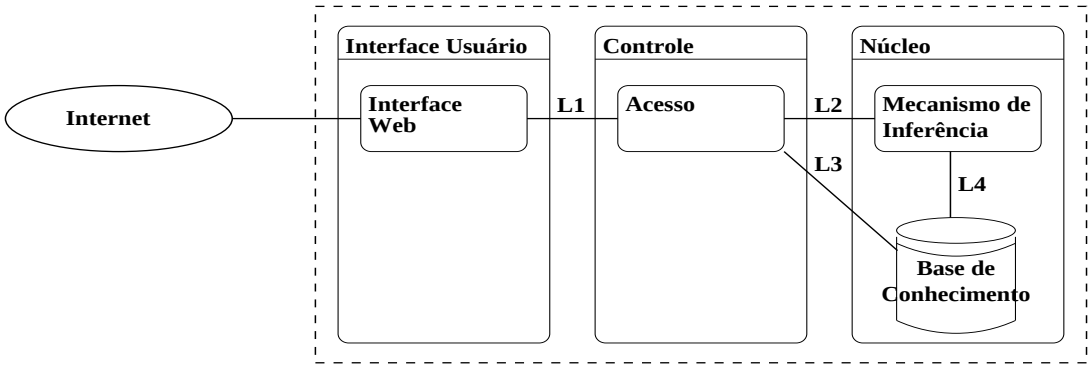

Figura 5.2: Arquitetura do sistema

O componente de *interface com o usuário* possibilita a comunicação do usuário com o sistema, permitindo:

- criar novas bases;
- cadastrar, remover e alterar atividades;
- cadastrar e remover recursos, e;
- solicitar propostas de planos.

Todas estas informações e requisições são encaminhadas para o componente de *controle* (figura 5.2 - linha L1).

O componente de *controle* implementa rotinas que controlam o acesso ao núcleo do sistema e o algoritmo de conversão do plano parcialmente ordenado para uma rede de atividades. Este componente se comunica com o *núcleo* enviando solicitação de novos planos (figura 5.2 - linha L2) e cadastrando, removendo ou alterando atividades e recursos (figura 5.2 - linha L3).

O componente *núcleo* é a base para o funcionamento do sistema. Formado pelo *mecanismo de inferência (MI)* e pela *base de conhecimento (BC)*. O *mecanismo de inferência* é responsável pela realização do raciocínio baseada nas informações representadas na *base de conhecimento* (figura 5.2 - linha L4).

A *base de conhecimento (BC)* é responsável por armazenar o conhecimento sobre as atividades que a empresa é capaz de executar. O conhecimento é codificado usando uma estrutura adaptada a partir da notação STRIPS e da ontologia TOVE.

Na próxima seção é apresentada a forma como são codificadas as atividades de projetos na base de conhecimento. Nesta estrutura, as pré-condições, efeitos positivos e efeitos negativos da atividade são determinados com base nos predicados que definem os estados terminais de uma atividade de empresa, ou seja, com base nos seus recursos.

5.3.1 Estrutura de representação de atividades

Usando a notação STRIPS e a definição de atividade segundo a ontologia TOVE, foi desenvolvida uma estrutura que define como uma atividade de projeto deve ser representada na base de conhecimento deste sistema.

A notação STRIPS define uma estrutura que permite com que algoritmos de planejamento possam manipular informações sobre ações (pré-condições, efeitos positivos e negativos), enquanto que a ontologia TOVE, fornece a estrutura de uma atividade de projeto, baseada em recursos que são usados, consumidos, liberados e produzidos.

Optou-se em utilizar a estrutura da notação STRIPS atribuindo um significado a cada elemento do operador, com base no que está descrito na ontologia TOVE de atividade. Desta forma, pode-se definir o operador que irá compor a base de conhecimento como:

$$Op = \langle N, P, A, D, E \rangle \tag{5.1}$$

Onde:

i. N é o nome da atividade;

ii. P é um conjunto de recursos, *usados* ou *consumidos*, que são pré-condições para a execução da atividade N;

iii. A é um conjunto de recursos *produzidos* pela atividade N;

iv. D é um conjunto de recursos *consumidos* pela atividade N, e;

v. E é um número natural que determina o esforço realizado para a execução da atividade N.

```
Op(energizar_campo,
   [tecnico,piso_instalado],
   [campo_energizado],
   [piso_instalado],
   32
  ).
```

Figura 5.3: Exemplo de operador STRIPS codificado na base de conhecimento

Considerando α como um elemento pertencente ao conjunto de atividades e R como um elemento pertencente ao conjunto de recursos, pode-se definir formalmente o significado dos elementos do operador como:

- $N = \alpha$;
- $P = \{R \mid use(R, \alpha) \cup release(R, \alpha) \cup consume(R, \alpha)\}$;
- $A = \{R \mid produce(R, \alpha)\}$;
- $D = \{R \mid consume(R, \alpha)\}$;
- $E = \{X \mid X \in \mathbb{N}\}$

A semântica deste operador segue as regras semânticas do operador STRIPS.

Como exemplo da codificação de uma atividade na base de conhecimento, na figura 5.3 é possível visualizar a codificação da atividade apresentada na figura 3.3.

Onde, *energizar_campo* é o nome da atividade, o recurso *técnico* é *usado* e *liberado*, o recurso *piso_instalado* é *consumido*, o recurso *campo_energizado* é *produzido* e 32 horas é o esforço necessário para a execução da atividade *energizar_campo*.

Uma vez definida a linguagem de representação das atividades, o segundo passo é a escolha do algoritmo que irá interpretar o conhecimento existente na base. O algoritmo de planejamento escolhido é apresentado na próxima seção.

5.3.2 Mecanismo de Inferência

Optou-se pela utilização do algoritmo POP para a implementação do mecanismo de inferência devido aos seguintes fatores:

i. eficiência dos algoritmos de ordem parcial;

ii. fácil implementação dos mesmos, e;

iii. possibilidade de representação de ações concorrentes usando os planos parcialmente ordenados.

As informações contidas em um plano parcialmente ordenado são suficientes para a construção de um modelo de rede de atividades, porém não estão estruturadas da melhor forma. Na próxima seção é apresentado o algoritmo de conversão de um plano parcialmente ordenado em uma rede de atividades.

5.3.3 Conversão de um plano parcialmente ordenado em uma rede de atividades

Para transformar as informações de um plano parcialmente ordenado em um modelo de rede de atividades, deve-se aplicar um algoritmo de conversão que tem como entrada um plano parcialmente ordenado (passos do plano, restrições de ordenação e vínculos causais) e como saída uma rede de atividades (conjunto de atividades, conjunto de recursos, relação de seqüência entre as atividades e conjunto de números naturais representando a duração das atividades).

A idéia do algoritmo de conversão está em listar, para todas as atividades pertencentes ao conjunto de passos do plano (ω), seus respectivos predecessores através das informações contidas no conjunto de restrições de ordenação (γ), no formato $S_i < S_j$, e não atribuir nenhum predecessor para cada elemento em S_j que não estiver em S_i. Após listada todas as atividades e seus respectivos predecessores o algoritmo realiza uma consulta na base de conhecimento para estabelecer qual a duração e respectivos recursos utilizados por cada atividade.

Depois que o algoritmo for executado, os valores gerados são utilizados para construir uma representação visual da rede de atividade. Por exemplo, para o plano parcialmente ordenado apresentado na figura 4.14 tem-se uma rede de atividade equivalente que pode ser visualizada na figura 5.4.

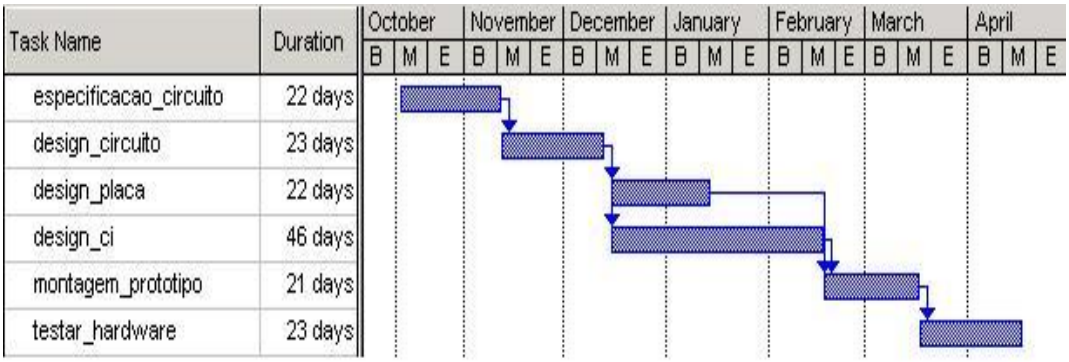

Figura 5.4: Rede de atividade equivalente ao plano da figura 4.14.

5.4 Implementação do Sistema

Nesta seção será descrito a forma como foi implementado o *mecanismo de inferência*, o componente de *controle* e a *interface com o usuário*.

O *mecanismo de inferência* foi implementado utilizando a linguagem PROLOG (STERLING; SHAPIRO, 1994), que acessa as informações contidas na *base de conhecimento* codificadas no formato visto anteriormente.

O componente de *controle* foi implementado em JAVA e utilizou a API JPL (DUSHIN, 1999) para possibilitar a comunicação dos componentes implementados em JAVA com o *núcleo* implementado em PROLOG. O componente de *controle* possui dois sub-componentes:

- *AcessoBase*: possui um conjunto de classes (*Atividade*, *EstadoEnabled*, $EstadoCaused$ e *Recurso*) responsáveis por acessar a base de conhecimento, adicionando, alterando e removendo dados (figura 5.5);
- *AlgoritmoPlanejador*: possui um conjunto de classes (POP, $Plano$, $Tradutor$ e $DadosProject$) responsáveis por solicitar planos ao *mecanismo de inferência* e converter os planos retornados em redes de atividades equivalentes (figura 5.6). As classes responsáveis pelo processo de geração de planos são as classes POP e $Plano$ e as classes responsáveis por converter os planos retornados em redes de atividades são as classes $Tradutor$ e $DadosProject$.

Na figura 5.7 é possível visualizar o diagrama de interação entre objetos da ação planejar. Quando o usuário informa o estado inicial e os objetivos que pretende alcançar para o sistema, a $InterfaceWeb$ cria um objeto da classe POP

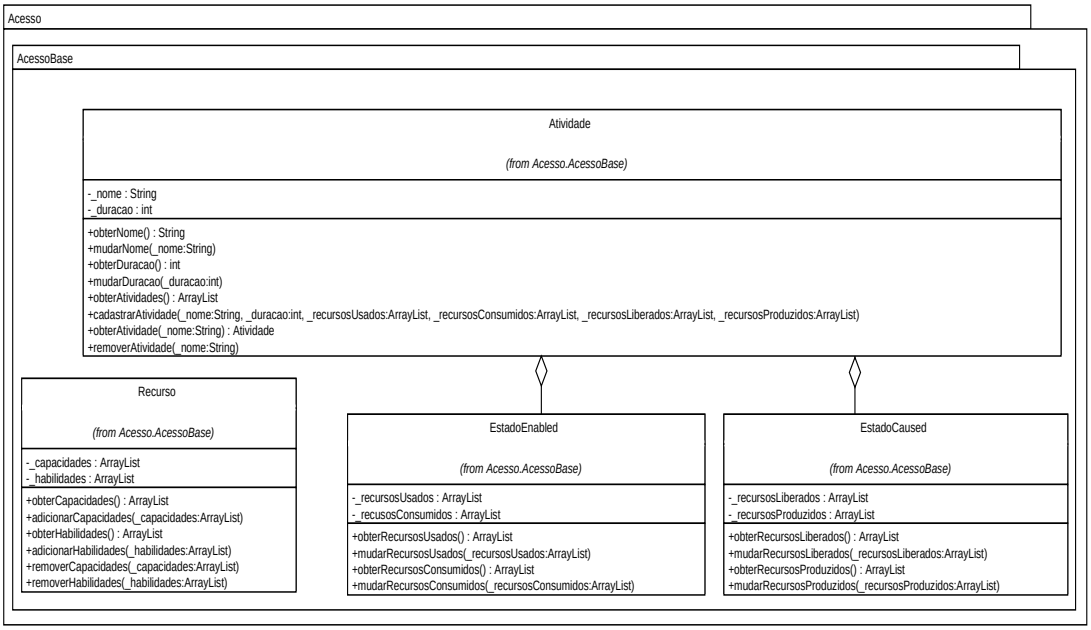

Figura 5.5: Diagrama de classes do componente AcessoBase.

e depois envia uma mensagem solicitando um plano. O objeto da classe POP retorna um conjunto de planos possíveis para a classe $InterfaceWeb$. Para cada plano retornado a classe $InterfaceWeb$ solicita para a classe $Tradutor$ uma rede de atividades equivalente. Cada rede de atividades é gravada em um arquivo texto, que depois pode ser exportado para o *MS-Project* (MICROSOFT CORPORATION, 2002), permitindo ao gerente de projetos a edição das redes de atividades.

A transfomação de um plano parcialmente ordenado em uma rede de atividades formatada no padrão do *MS-Project* é realizada através da transferência dos dados de um plano parcialmente ordenado para os atributos codificados na classe $DadosProject$.

O formato de arquivo do *MS-Project* foi escolhido porque o *MS-Project* é uma das ferramentas mais utilizadas na área de gestão de projetos.

Com o intuito de disponibilizar o sistema para o maior número possível de pessoas, o componente de *interface com o usuário* foi implementado em JAVA e possui uma versão Web (figura 5.8).

Para ilustrar e validar a funcionalidade deste sistema, no capítulo seguinte são descritos exemplos, juntamente com os resultados experimentais obtidos.

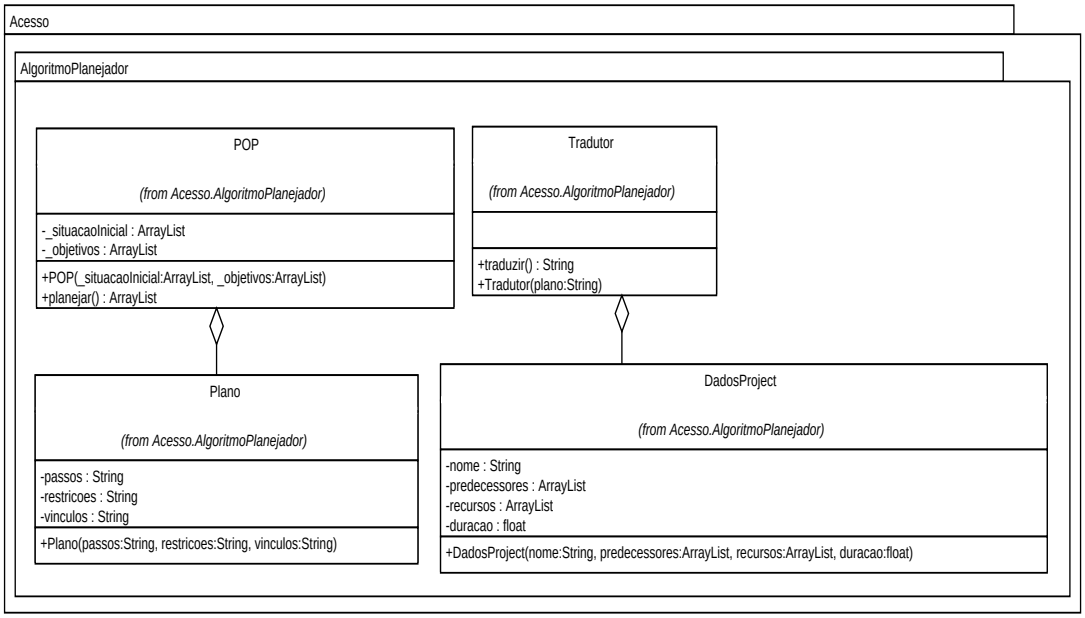

Figura 5.6: Diagrama de classes do componente AlgoritmoPlanejador.

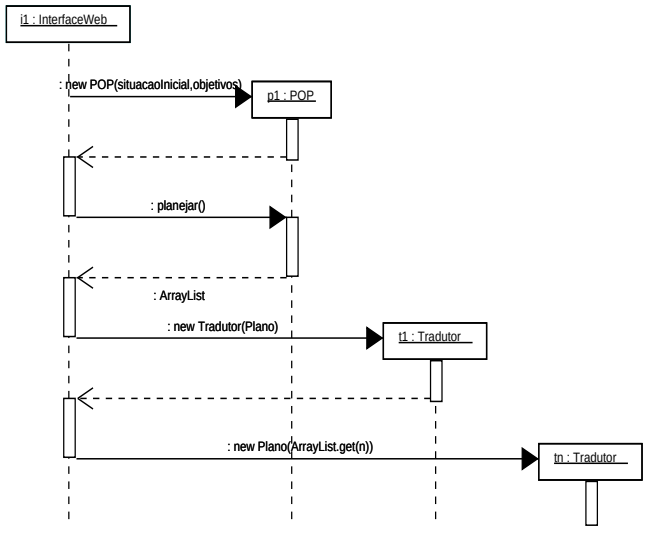

Figura 5.7: Diagrama de interação entre objetos da ação Planejar

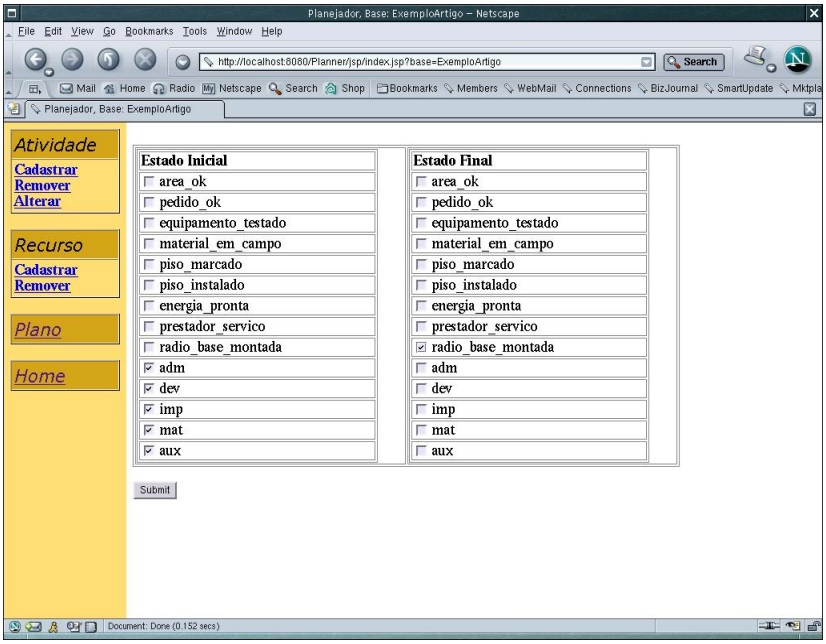

Figura 5.8: Interface com o usuário

6 Resultados Experimentais

Neste capítulo é apresentado o método de avaliação do sistema, a descrição dos testes e a avaliação dos resultados.

6.1 Método de avaliação

Com o intuito de avaliar a funcionalidade do sistema, foram realizados diversos testes com informações de projetos reais executados por uma empresa da área de telecomunicações. Estes experimentos foram realizados de acordo com a seguinte sistemática:

i. coleta de informações sobre projetos já executados pela organização;

ii. seleção dos projetos com informações completas sobre todas as atividades executadas durante o projeto;

iii. identificação dos atributos de cada atividade: recursos, nome, esforço e relação de precedência;

iv. cadastro de todas as atividades na base de conhecimento;

v. execução de consultas ao sistema, e;

vi. comparação das respostas fornecidas pelo sistema com os planos reais.

Os resultados dos experimentos foram avaliadas de acordo com os três requisitos estabelecidos no capítulo anterior:

i. os planos retornados devem ser consistentes com os objetivos fornecidos;

ii. os planos devem representar o conjunto de atividades que deverá ser executado para desenvolver o produto, as relações de precedência, os recursos necessários para a execução das atividades e a estimativa da duração das atividades, e;

iii. o tempo para solução do problema deve ser satisfatório.

Na próxima seção é apresentado o contexto em que a organização que cedeu os planos atua e um exemplo simplificado de projeto desta organização.

6.2 Descrição dos testes

6.2.1 Contexto

A organização que forneceu o conjunto de projetos para testes é uma empresa que atua no mercado de telecomunicações, desenvolvendo e implantando centrais e estações rádio-base para telefonia móvel, sistemas de comunicação, desenvolvimento de aparelhos de telefones, entre outras atividades.

Os projetos utilizados nos testes realizados estão relacionados ao desenvolvimento e implantação de estações rádio-base para telefonia móvel. Entre vários cronogramas de projetos, num total de 92, foram selecionados todos os cronogramas que estavam completos, ou seja, todos aqueles que continham todas as informações sobre os recursos necessários, precedência e duração das atividades, totalizando 41 cronogramas de projetos. Entre estes, foram selecionados 10 cronogramas que refletiam todos os tipos de projetos analisados.

Para melhor visualizar o tipo de projeto executado pela empresa, na próxima seção é apresentado um exemplo da utilização do sistema desenvolvido com dados de um projeto onde o objetivo é desenvolver uma estação rádio base.

6.2.2 Exemplo simplicado de projeto

Considere uma empresa com um conjunto de equipes - por exemplo, que realizem pré-montagem (ADM), desenvolvimento (DEV), implantação (IMP), manutenção (MAT) e serviços auxiliares (AUX) - e capaz de executar as atividades descritas na tabela 6.1.

Tabela 6.1: Descrição das atividades

Atividade	**Recursos**			**Esforço**
	Utilizados	**Consumidos**	**Produzidos**	
procurar_area2	ADM		area_ok	16 h.
pedido_prefeitura2	ADM area_ok		pedido_ok	8 h.
fazer_testes_fabrica3	DEV		eq_testado	24 h.
colocar_material_campo1	AUX pedido_ok eq_testado	eq_testado	material_campo	8 h.
fazer_marcacao_piso2	IMP		piso_marcado	16 h.
fazer_instalacao_piso4	material_campo piso_marcado IMP	piso_marcado	piso_instalado	32 h.
energizar_campo4	piso_instalado IMP		energia_pronta	32 h.
fazer_marcacao_piso5	AUX		piso_marcado	40 h.
contratar_montagem20	prestador_servico ADM		radio_base_montada	160 h.

A primeira ação para viabilizar o uso do sistema é inserir informações sobre os recursos disponíveis na empresa e sobre o conjunto de atividades que a empresa é capaz de executar (tabela 6.1).

Depois que as informações tenham sido devidamente armazenadas na base de conhecimento, os gerentes de projetos podem utilizar o sistema, solicitando planos para os projetos . Por exemplo, considere um projeto onde o objetivo é construir uma estação rádio base. O gerente deste projeto pode utilizar o sistema informando a situação do início do projeto (que expressa o conjunto de todas as equipes da empresa {ADM,DEV;IMP,MAT,AUX}) e a situação que quer alcançar, ou seja, a estação rádio base montada (figura 5.8).

Dado a situação inicial e a situação objetivo, o sistema irá acessar a base de conhecimento à procura de um conjunto de atividades, que ordenadas, possibilitam o cumprimento dos objetivos.

Este sistema pode retornar um ou mais planos possíveis para cada caso. Para o caso descrito nesta seção são três as possíveis soluções (figura 6.1).

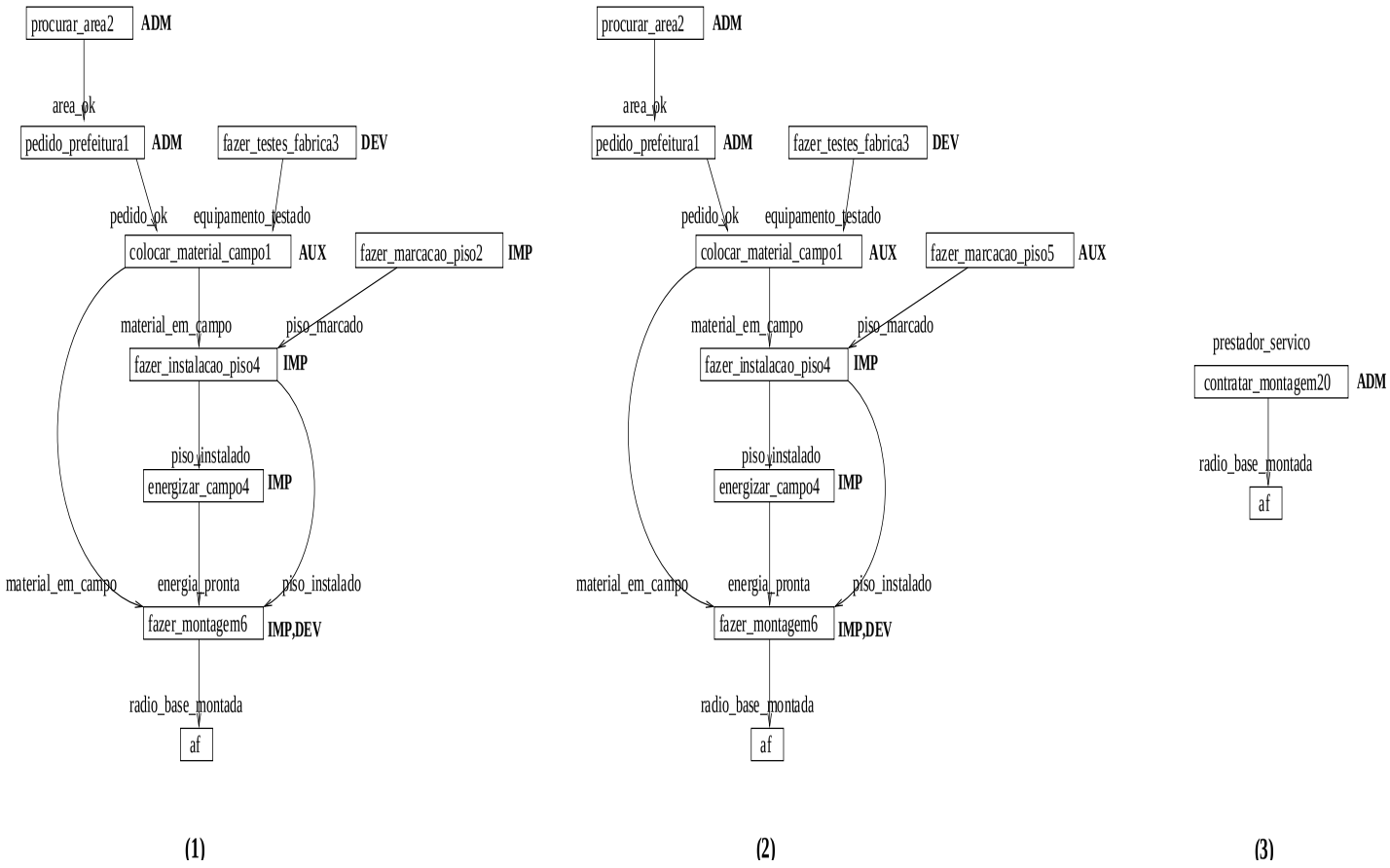

Figura 6.1: Planos parcialmente ordenados

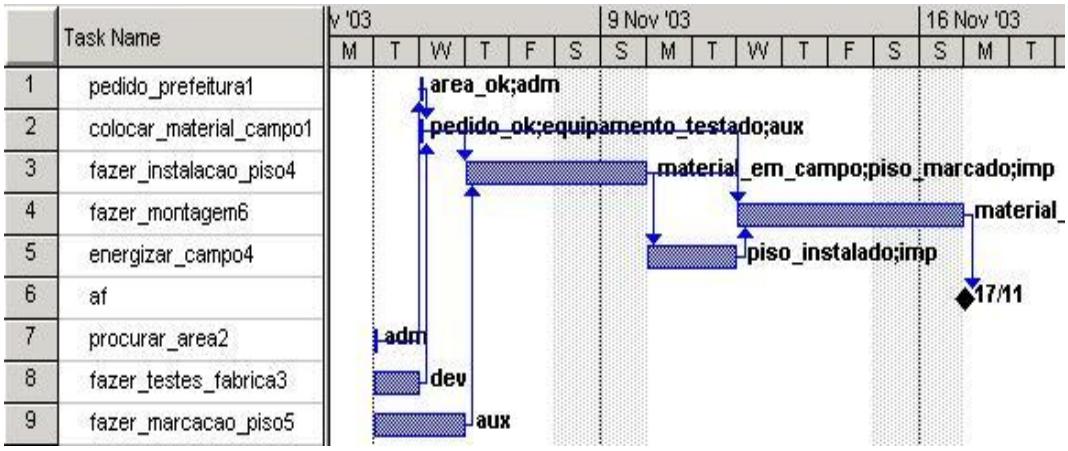

Figura 6.2: Rede de atividades do plano (2) da figura 6.1

A diferença entre os planos (1) e (2) da figura 6.1 é que no plano (1) a atividade $fazer\_marcacao\_piso$ possui uma duração de 2 dias, enquanto que no plano (2) a mesma atividade possui uma duração de 5 dias, isto porque a segunda atividade utiliza um grupo de recursos menos especializado. O plano número (3) demonstra que uma das opções da empresa para alcançar o objetivo determinado, além da própria empresa desenvolver o projeto, é terceirizar o projeto.

Aplicando o algoritmo de conversão sobre o plano (2) da figura 6.1 é possível gerar o modelo de rede de atividade da figura 6.2. O resultado final é convertido para o formato do *Microsoft Project*, para permitir que o gerente de projetos possa eventualmente modificar o plano proposto (figura 6.2).

6.3 Resultados

Conforme mencionado anteriormente, para realizar os testes foram utilizados 10 cronogramas que refletiam todos os tipos de projetos fornecidos pela empresa. Para cada projeto cadastrado na base de conhecimento, foram realizadas consultas questionando um plano possível para aquele projeto. Além destas consultas, foram realizadas consultas formadas a partir da conjunção de consultas anteriores.

Com o intuito de avaliar os resultados alcançados neste trabalho, as próximas seções discutem separadamente cada critério estabelecido no capítulo 5, ou seja, a precisão do sistema, os elementos que devem estar contidos nos planos retornados e o tempo necessário para o sistema retornar uma resposta.

6.3.1 Precisão do sistema

Na tabela 6.2 é possível visualizar os resultados obtidos na execução dos testes. Nesta tabela, as linhas significam as consultas realizadas e as colunas representam os atributos utilizados na análise das respostas e da performance do sistema, onde:

i. *ConjAtiv*: retorna 1 se o conjunto de atividades do plano real for diferente do plano fornecido pelo sistema e 0 caso contrário;

ii. *SeqAtiv*: retorna 1 se a sequência das atividades do plano real for diferente do plano fornecido pelo sistema e 0 caso contrário;

iii. *ConjRec*: retorna 1 se o conjunto de recursos do plano real for diferente do plano fornecido pelo sistema e 0 caso contrário;

iv. *Dur*: retorna 1 se a duração do plano real for diferente do plano fornecido pelo sistema e 0 caso contrário.

v. *NrAtiv*: significa o número de atividades do projeto;

vi. *Tempo*: significa o tempo necessário em segundos para retornar a resposta;

vii. *Espaço*: significa o número de nodos que foram abertos para alcançar a resposta, e;

viii. *NrOper*: significa o número de atividades cadastradas na base de conhecimento.

Para quantificar o erro do sistema foi utilizada a seguinte equação:

$$erro = \frac{1}{n} \sum_{i=n}^{n} \frac{(ConjAtiv(i) + SeqAtiv(i) + ConjRec(i) + Dur(i))}{4} \tag{6.1}$$

onde n é o número de consultas realizadas.

A precisão do sistema é denotada pela seguinte equação:

$$acc = 1 - erro \tag{6.2}$$

Tabela 6.2: Síntese dos resultados obtidos

Consultas	**Análise das Respostas**					**Performance do Sistema**		
	ConjAtiv	*SeqAtiv*	*ConjRec*	*Dur*	*NrAtiv*	*Tempo*	*Espaço*	*NrOper*
C_1	0	0	0	1	15	0.11	34	196
C_2	0	0	0	1	15	0.09	34	196
C_3	0	0	0	1	22	0.42	52	196
C_4	0	0	0	1	22	0.37	51	196
C_5	0	0	0	1	22	0.74	51	196
C_6	0	0	0	1	27	0.47	72	196
C_7	0	0	0	1	26	3.07	60	196
C_8	0	0	0	1	23	0.73	54	196
C_9	0	0	0	1	23	0.75	54	196
C_{10}	0	0	0	1	26	2.90	60	196
$C_1, \cdots, C_3$	0	0	0	1	52	11.62	520	196
$C_1, \cdots, C_4$	0	0	0	1	74	442.73	660	196
$C_1, \cdots, C_5$	0	0	0	1	96	8450.95	750	196

Utilizando as equações definidas em 6.1 e 6.2, calculou-se que a precisão do sistema é de 75%. Percebe-se que o erro do sistema está sempre relacionado com a duração dos projetos. Este erro ocorre porque as informações sobre o atraso e avanço das atividades não é representado no sistema.

O atraso de uma atividade corresponde a um atraso para o início da atividade, mesmo que todas as suas pré-condições já tenham sido satisfeitas. O avanço significa o contrário.

De qualquer maneira, a comparação dos resultados obtidos pelo sistema desenvolvido com as redes de atividades propostas pelos gerentes de projetos, mostra que os resultados obtidos são coerentes com os projetos reais.

6.3.2 Estrutura dos planos retornados

Os planos retornados apresentam todas as informações solicitadas pelos requisitos do sistema: o conjunto de atividades que deverá ser executada para desenvolver o produto, os recursos necessários para a execução das atividades, a seqüência das atividades e a duração das atividades.

Na figura 6.3 é apresentada uma rede de atividades de um projeto real executado pela empresa, enquanto que na figura 6.4 é apresentado uma rede de atividades equivalente proposta pelo sistema. Através destas duas figuras é possível

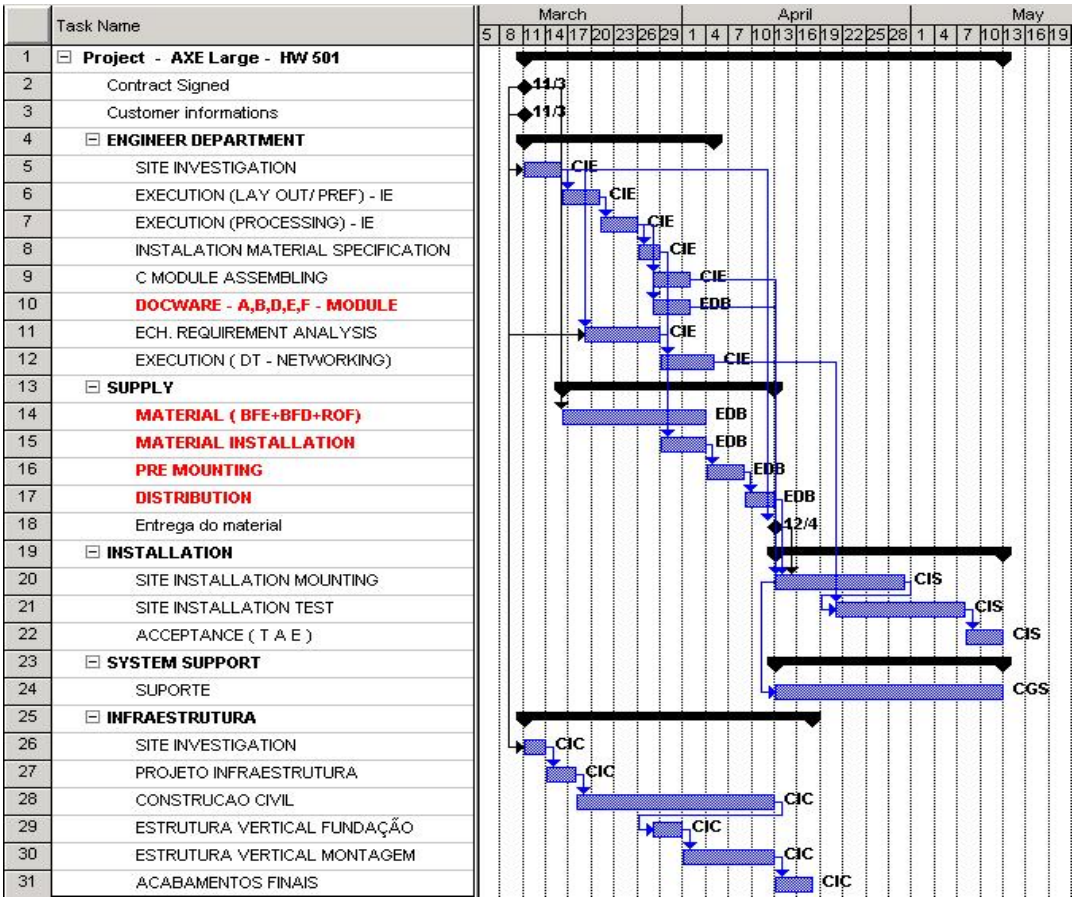

Figura 6.3: Cronograma de um projeto executado

verificar que o sistema é capaz de retornar diagramas de redes com todas as informações necessárias e que a solução encontrada pelo sistema é coerente com o projeto real.

Comparando a solução proposta pelo sistema (figura 6.4) com o rede de atividades real (figura 6.3), pode-se verificar que a solução retornada pelo sistema sofre de alguns problemas de legibilidade.

Talvez o problema de legibilidade possa ser resolvido adotando algum algoritmo hierárquico de planejamento (EROL; HENDLER; NAU, 1994) ou adaptando o algoritmo atual. Se adotado um algoritmo hierárquico de planejamento, a resposta do sistema possa ser estruturada de maneira hierárquica, assim como o cronograma da figura 6.3.

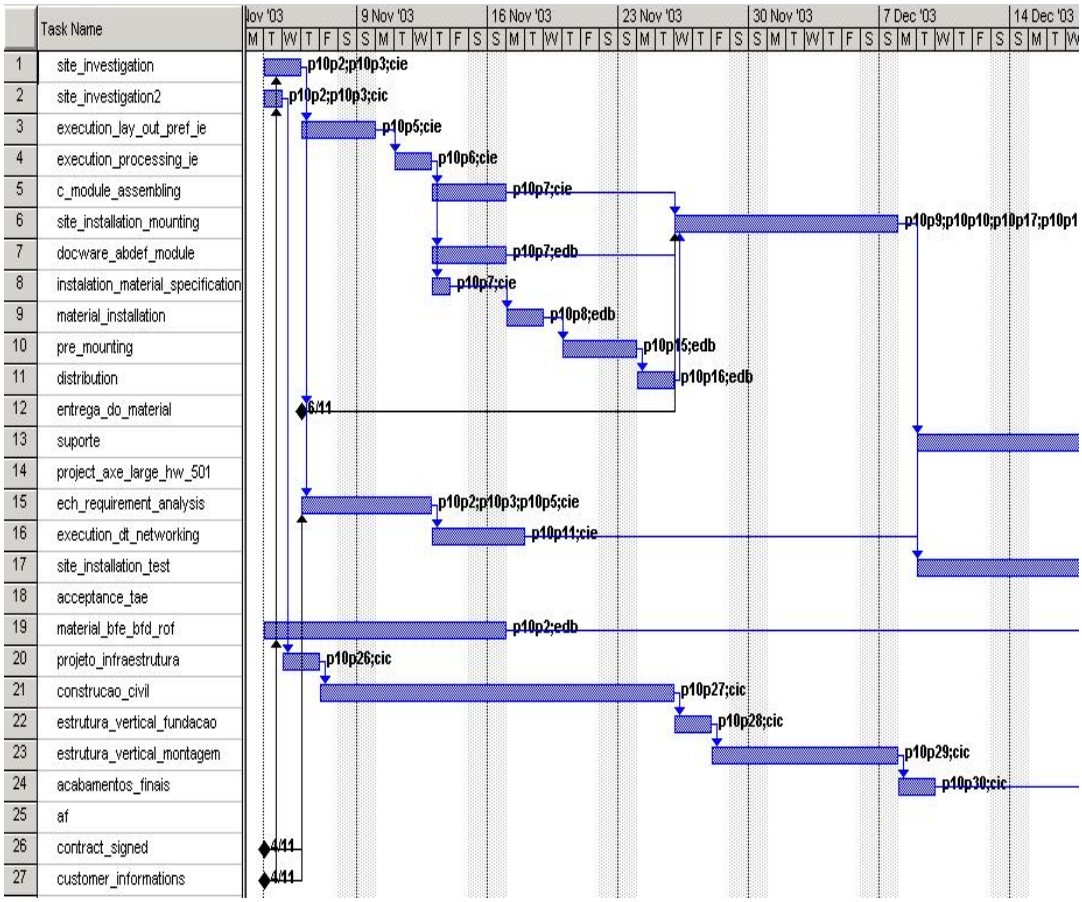

Figura 6.4: Cronograma retornado pelo sistema

6.3.3 Performance do sistema

A performance de um sistema pode ser medida levando-se em consideração o tempo e espaço de memória que o sistema utilizou para propor uma determinada solução. Neste contexto, quanto menor for o tempo e o espaço de memória ocupado, melhor será a sua performance.

Para medir a performance deste sistema, foram realizados diversos testes (tabela 6.2), onde foram medidos o número de atividades de cada diagrama de rede proposto, o tempo e o espaço de memória que o sistema utilizou para propor uma solução. O resultado desta medição pode ser visualizado no gráfico da figura 6.5.

Analisando o gráfico da figura 6.5 é possível constatar que para projetos com um número maior que 90 atividades o sistema começa a degradar, passando da casa dos segundos para a cada dos 80 minutos.

De fato, esta característica de exponencialidade está presente em quase todos

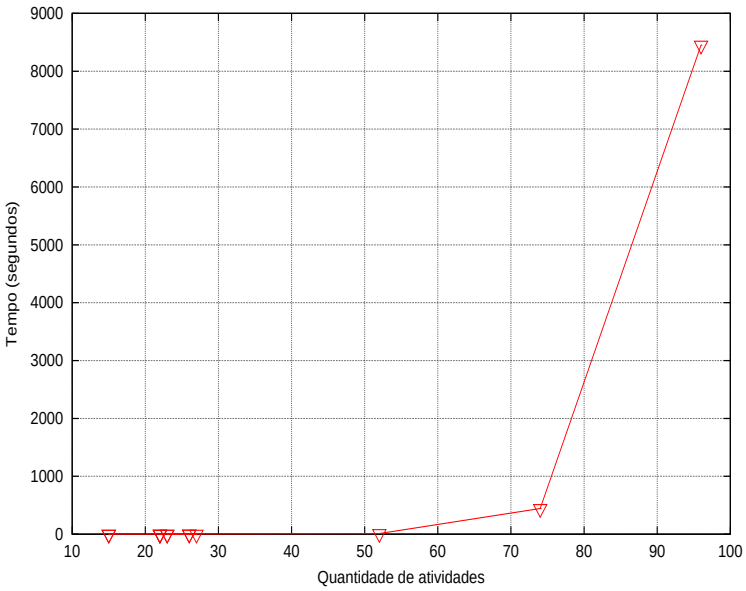

Figura 6.5: Análise da performance do sistema

os domínios de problemas de planejamento. Talvez, uma maneira para melhorar a performance do sistema é realizando algumas otimizações na implementação ou adotando outras abordagens na implementação do mecanismo de inferência.

7 Considerações Finais

Neste trabalho foram apresentados conceitos, características e dificuldades da tarefa de planejamento na área de gestão de projetos. Também foram analisado conceitos e ferramentas que possibilitam a implementação de sistemas capazes de propor planos de projetos. No decorrer do trabalho foi possível destacar as características, vantagens e desvantagens do Cálculo de Situações e dos algoritmos de ordem parcial.

Esta dissertação poderá ser útil para pessoas que queiram conhecer conceitos básicos sobre planejamento na área de gestão de projetos e planejamento na área de inteligência artificial (IA), além de servir como exemplo da aplicabilidade dos algoritmos de planejamento da IA em problemas reais.

Ao longo deste trabalho, constatou-se que, apesar da facilidade para modelar um domínio usando o Cálculo de Situações, existem várias características desse formalismo que prejudicam a implementação de soluções para problemas reais, tais como: a incapacidade de representar ações concorrentes, com duração e a baixa eficiência computacional atrelada ao paradigma de programação declarativo. Em contrapartida, o algoritmo POP mostrou-se adequado para compor o arcabouço para a solução do problema. Os algoritmos de ordem parcial, são os algoritmos, que de maneira geral, tem mostrado maior eficiência computacional para resolver problemas de planejamento e a estrutura de planos parcialmente ordenados mostrou-se extremamente simples para ser manipulada e convertida em um diagrama de redes.

A comparação dos resultados obtidos pelo sistema desenvolvido com as redes de atividades propostas pelos gerentes de projetos, mostra que os resultados obtidos são coerentes com os projetos reais.

Os planos retornados apresentam todas as informações solicitadas pelos re-

quisitos do sistema: o conjunto de atividades que deverá ser executada para desenvolver o produto, os recursos necessários para a execução das atividades, a seqüência das atividades e a duração das atividades.

Na avaliação da performance do sistema, constatou-se que para projetos com um número maior que 90 atividades o sistema começa a degradar. Para projetos com um número menor que 90 atividades, o sistema retorna as possíveis respostas em questão de segundos. Alternativas para melhorar a performance do sistema são: realizar algumas otimizações na implementação ou adotar outras abordagens na implementação do mecanismo de inferência.

Também foi possível testar o uso de ontologias na reutilização do conhecimento para a construção de novos sistemas baseados em conhecimento. Neste trabalho, as definições de atividade, recursos, pré-condições e efeitos da ontologia TOVE foram utilizadas para atribuir significado aos objetos que estão representados nos operadores da base de conhecimento.

Durante o desenvolvimento e a avaliação do sistema, constatou-se a importância do processo de aquisição de conhecimento, seja na forma manual ou automática. Se a aquisição de conhecimento for realizada de maneira correta, o sistema irá propor um ou vários planos corretos para um determinado problema, caso contrário, o sistema não apresentará soluções úteis. É por isso que, uma das sugestões para trabalhos futuros é o aperfeiçoamento do sistema adicionando módulos de aquisição automática sobre as informações de atividades de projetos. Um módulo de aquisição automática de conhecimento pode reduzir as chances de erro durante o processo de aquisição de conhecimento.

De qualquer forma, acredita-se que o sistema desenvolvido neste trabalho, possa servir como uma ferramenta para que uma empresa possa estruturar o conhecimento sobre projetos já executados, disponibilizando-o aos membros da organização. Isso permite minimizar alguns dos problemas relacionados a tarefa de planejamento na área de gestão de projetos: falta de conhecimento para realizar a tarefa de planejamento, complexidade inerente dos projetos e falta de soluções alternativas.

Além da sugestão para trabalho futuro já enumerada anteriormente, outras sugestões são:

i. desenvolver uma interface com o usuário mais amigável, por exemplo, utilizando uma interface gráfica com diagramas equivalentes ao da ontologia TOVE;

ii. avaliar a utilidade de algoritmos hierárquicos neste problema, e;

iii. utilizar uma abordagem que una técnicas de planejamento com técnicas para solucionar problemas de escalonamento, como descrito por Laborie (2003).

Referências Bibliográficas

AARUP, M. et al. *OPTIMUM-AIV: A knowledge-based planning and sheduling system for spacecraf AIV*. San Matro, California: Morgan Kaufmann, 1994.

ALFERES, J. J.; LI, R.; PEREIRA, L. M. Concurrent actions and changes in the situation calculus. *Proc. of IBERAMIA*, McGraw-Hill, p. 93–104, 1994.

AYLETT, R. S. et al. Ai planning: solutions for real world problems. *Knowledge-Based Systems*, v. 13, p. 61–69, April 2000.

BAIER, J.; PINTO, J. Non-instantaneous actions and concurrency in the situation calculus. *In 10th European Summer School in Logic, Language and Information*, 1998.

BARRET, A.; WELD, D. S. Partial-order planning: evaluating possible efficiency gains. *Artificial Intelligence*, n. 67, p. 71–112, May 1994.

BARTH, F. J.; GOMI, E. S. An extension of situation calculus applied to project management. In: *4th Argentine Symposium on Artificial Intelligence*. Santa Fé, Argentina: [s.n.], 2002. v. 4, p. 231–241.

BARTH, F. J.; GOMI, E. S. Extensão do cálculo de situações aplicada à gestão de projetos. In: ABE, J. M.; FILHO, J. I. S. (Ed.). *Advances in Logic, Artificial Intelligence and Robotics. Proceedings of the Congress of Logic Applied to Technology - LAPTEC'2002*. [S.l.: s.n.], 2002. II, p. 139–146.

BRANDãO, J. de S. *Mitologia Grega*. [S.l.]: Editora Vozes, 2000.

CURRIE, K.; TATE, A. O-plan: the open planning architecture. *Artificial Intelligence*, v. 52, n. 1, p. 49–86, 1991 1991.

DOETS, K. *From Logic to Logic Programming*. [S.l.]: Massachusetts Institute of Technology, 1994. ISBN 0-262-04142-1.

DUSHIN, F. *JPL: A Java Interface to Prolog*. http://www.swi-prolog.org/, 1999.

ENTERPRISE INTEGRATION LABORATORY. *TOVE Ontology Project*. http://www.eil.utoronto.ca/enterprise-modelling/tove/index.html, 2002.

EROL, K.; HENDLER, J.; NAU, D. S. Htn planning: Complexity and expressivity. *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seatle, Washington, USA, v. 2, p. 1123–1128, 1994.

FALBO, R. de A. *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 1998.

FIKES, R.; NILSSON, N. J. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, v. 2, n. (3/4), p. 189–208, 1971.

FOX, M. S.; GRüNINGER, M. Enterprise modelling. *AI Magazine*, p. 109–121, 1998.

FREITAS, F. L. G. de. Anais do xxiii congresso da sociedade brasileira de computação. In: _____. [S.l.]: Sociedade Brasileira de Computação, 2003. cap. Ontologias e Web Semântica, p. 1–52.

GENESERETH, M. R.; NILSSON, N. J. *Logical Foundations of Artificial Intelligence*. Palo Alto: Morgan Kaufmann Publishers, Inc., 1987. ISBN 0-934613-31-1.

GOMI, E. S. Gestão de projetos. Apostila da disciplina Práticas de Eletrônica e Eletricidade II. Escola Politécnica da Universidade de São Paulo. Outubro 2002.

GOMI, E. S. Planejamento e controle de projetos. Notas de aula do Curso de Gestão de Projetos. Escola Politécnica da Universidade de São Paulo. Setembro 2002.

GRüNINGER, M.; FOX, M. S. An activity ontology for enterprise modelling. In: *Workshop on Enabling Technologies - Infrastructures for Collaborative Enterprises*. West Virginia University: [s.n.], 1994.

GRUBER, T. R. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human and Computer Studies*, v. 43, n. 5/6, p. 907–928, 1995.

HEISSERMAN, J.; CALLAHAN, S.; MATTIKALI, R. A design representation to support automated design generation. In: *Proceedings of the Sixth International Conference on Artificial Intelligence in Design*. [S.l.: s.n.], 2000. p. 545–566.

JACKSON, P. *Introduction to Expert Systems*. 3. ed. Harlow, England: Addison-Wesley, 1998.

KORTENKAMP, D. A day in an astronaut's life: Reflections on advanced planning and scheduling technology. *IEEE Intelligent Systems*, p. 8–11, March/April 2003.

LABORIE, P. Algorithms for propagating resource constraints in ai planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, n. 143, p. 151–188, 2003.

LIEBOWITZ, J. *Knowledge Management Handbook*. [S.l.]: CRC Press, 1999.

LIFSCHITZ, V. On the semantics of strips. In: ALLEN, J.; HENDLER, J.; TATE, A. (Ed.). *Readings in Planning*. [S.l.]: Morgan Kaufman, 1990. p. 523–531.

LIN, F.; REITER, R. Rules as actions: A situation calculus semantics for logic programs. *Journal of Logic Programming*, v. 31, n. 1-3, p. 299–330, 1997.

LIN, F.; SHOHAM, Y. Concurrent actions in the situation calculus. *In Proc. of AAAI*, p. 590–595, 1992.

MCCARTHY, J.; HAYES, P. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, v. 4, p. 463–502, 1969.

MICROSOFT CORPORATION. *Microsoft Project*. http://www.microsoft.com/office/project/default.asp, 2002.

MILLER, R.; SHANAHAN, M. Narratives in the situation calculus. *Journal of Logic and Computation*, v. 4, n. 5, p. 513–530, October 1994.

MORENO, M. D. R.; KEARNEY, P. Integrating ai planning techniques with workflow management system. *Knowledge-Based Systems*, v. 15, p. 285–291, July 2002.

NGUYEN, X.; KAMBHAMPATI, S. Reviving partial order planning. In: *Proc. IJCAI-01*. Seattle, WA: [s.n.], 2001. p. 459–464.

NILSSON, N. J. *Artificial Intelligence: A New Synthesis*. San Francisco, California: Morgan Kaufmann Publishers, Inc, 1998.

O'LEARY, D. E. Enterprise knowledge management. *IEEE Computer*, p. 54–61, march 1998.

PANKASKIE, M.; WAGNER, M. Use of clips for representation and inference in a clinical event monitor. In: *Proceedings of the 1997 American Medical Informatics Association Anual Fall Symposium*. [S.l.: s.n.], 1997. p. 193–197.

PEREIRA, S. L. *Planejamento Abdutivo no Cálculo de Eventos*. Dissertação (Dissertação de Mestrado) — Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, Abril 2002.

PMI. *A Guide to the Project Management Body of Knowledge*. Maryland, USA, 2000.

PRESSMAN, R. S. *Engenharia de Software*. 5. ed. [S.l.]: McGraw-Hill, 2002.

REITER, R. A logic for default reasoning. *Artificial Intelligence*, v. 13, p. 81–132, 1980.

REZENDE, S. O. (Ed.). *Sistemas Inteligentes: Fundamentos e Aplicações*. [S.l.]: Editora Manole, 2003.

RUSSEL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. 2. ed. [S.l.]: Prentice-Hall, 2003. ISBN 0-13-790395-2.

SEBESTA, R. W. *Concepts of Programming Languages*. 5. ed. [S.l.]: Pearson Addison Wesley, 2001.

SHANAHAN, M. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. [S.l.]: The MIT Press, 1997. ISBN 0-262-19384-1.

SRIVASTAVA, B.; KAMBHAMPATI, S.; DO, M. B. Planning the project management way: Efficient planning by effective integration of causal and resource reasoning in realplan. *Artificial Intelligence*, v. 131, p. 73–134, September 2001.

STERLING, L.; SHAPIRO, E. *The Art of Prolog*. 2. ed. Cambridge, Massachusetts: The MIT Press, 1994. ISBN 0-262-19338-8.

WELD, D. S. An introduction to least commitment planning. *AI Magazine*, v. 15, n. 4, p. 27–61, 1994.

YANG, Q. *Intelligent Planning: a decomposition and abstraction based approach*. Berlin, Germany: [s.n.], 1997.

Apêndice I – Atena

> "Com o fito de evitar derramamento de sangue heleno, *Ulisses, inspirado por Atená*, imaginou o genial estratagema do cavalo de madeira, introduzido na cidade, pejado de guerreiros, que saquearam Tróia." (BRANDãO, 2000)

> "Atená é antes de mais nada a deusa da inteligência, da razão, do equilíbrio apolíneo, do espírito criativo e, como tal, preside às artes, à literatura e à filosofia de modo particular, à música e a toda e qualquer atividade do espírito. Deusa da paz, é a boa *conselheira do povo e de seus dirigentes*." (BRANDãO, 2000)

> "A ave predileta de Atená era a *coruja*, símbolo da reflexão que domina as trevas." (BRANDãO, 2000)

Atená era quem dava conselhos, inspirava e auxiliava Ulisses em todas as suas "empreitadas".

O sistema *Atena* fornece "conselhos" aos gerentes de projetos.