Combinando Bancos de Dados e Raciocínio Baseado em Casos para Apoio à Decisão em Planejamento Ambiental

Daniel dos Santos Kaster

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Combinando Bancos de Dados e Raciocínio Baseado em Casos para Apoio à Decisão em Planejamento Ambiental

Daniel dos Santos Kaster[1]

Fevereiro de 2001

Banca Examinadora:

- Profa. Dra. Claudia Maria Bauzer Medeiros (Orientadora)
- Profa. Dra. Ana Carolina Salgado
  Centro de Informática - Universidade Federal de Pernambuco
- Prof. Dr. Alexandre Xavier Falcão
  Instituto de Computação - Universidade Estadual de Campinas
- Profa. Dra. Ariadne M. Brito Rizzoni Carvalho
  Instituto de Computação - Universidade Estadual de Campinas (Suplente)

[1] O autor é bacharel em Ciência da Computação pela Universidade Estadual de Londrina.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Kaster, Daniel dos Santos

K156c Combinando bancos de dados e raciocínio baseado em casos para apoio à decisão em planejamento ambiental / Daniel dos Santos Kaster -- Campinas, [S.P. :s.n.], 2001.

K155c Orientadora : Claudia Maria Bauzer Medeiros

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Sistema de informação geográfica. 2. Inteligência artificial. 3. Sistema de recuperação da informação. I. Medeiros, Claudia Maria Bauzer. II. Universidade Estadual de Campinas. Instituto de Computação . III. Título.

Combinando Bancos de Dados e Raciocínio Baseado em Casos para Apoio à Decisão em Planejamento Ambiental

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Daniel dos Santos Kaster e aprovada pela Banca Examinadora.

Campinas, 05 de Abril de 2001.

Profa. Dra. Claudia Maria Bauzer Medeiros
(Orientadora)

Profa. Dra. Heloisa Vieira da Rocha
(Co-orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 12 de março de 2001, pela Banca Examinadora composta pelos Professores Doutores:

Prof. Dr. Ana Carolina Salgado
UFPe

Profa. Dra. Claudia Maria Bauzer Medeiros
IC - UNICAMP

Prof. Dr. Alexandre Xavier Falcão
IC - UNICAMP

"Mesmo que eu tivesse o dom da profecia, e conhecesse todos os mistérios e toda a ciência; mesmo que eu tivesse toda a fé, a ponto de transportar montanhas, se não tiver amor, não sou nada."

(1 Coríntios 13, 2)

Resumo

O planejamento ambiental faz uso de Sistemas Espaciais de Apoio à Decisão (*Spatial Decision Support Systems - SDSS*)) para resolução de problemas. Estes sistemas fornecem ambientes integrados que permitem aos usuários lidar com dados e modelos em tarefas de análise e simulação. Entretanto, eles geralmente provêm modelos genéricos que precisam ser especializados para adequar-se a situações particulares. Uma vez que este processo requer considerável esforço e perícia, é crucial permitir que os planejadores beneficiem-se de experiências de outros especialistas.

O objetivo desta dissertação é desenvolver mecanismos para auxiliar especialistas de planejamento ambiental a solucionar problemas de forma incremental. A solução aqui apresentada consiste em acoplar Raciocínio Baseado em Casos (*Case-Based Reasoning - CBR*) ao sistema espacial de apoio à decisão WOODSS (*WOrkflOw-based spatial Decision Support System*), desenvolvido no laboratório LIS do Instituto de Computação da UNICAMP. O sistema WOODSS interage com um Sistema de Informação Geográfica e provê facilidades para manipulação de modelos, documentando-os em termos de *workflows* científicos. O foco deste trabalho é especificar e implementar novos módulos de armazenamento e recuperação de modelos para o WOODSS, utilizando técnicas de CBR.

As principais contribuições desta pesquisa são: (a) o levantamento de requisitos para o uso de CBR para apoio à decisão no contexto ambiental; (b) o desenvolvimento de algoritmos para gerenciamento de modelos fundamentados em CBR; e (c) a extensão do sistema WOODSS, tornando-o mais efetivo para a resolução de problemas a partir de casos precedentes.

Abstract

Environmental planning takes advantage of Spatial Decision Support Systems (SDSS) for problem solving. These softwares supply integrated frameworks which permit users to deal with data and models in analysis and simulation tasks. However, they usually provide generic models which need to be specialized to fit particular situations. Since this process requires considerable effort and expertise, it is crucial to allow planners to profit from others' experience.

The goal of this dissertation is to develop mechanisms to help environmental planners to solve problems incrementally. The solution presented here consists in coupling Case-based Reasoning (CBR) to the WOODSS spatial decision support system (WOrkflOw-based spatial Decision Support System), developed at the LIS laboratory at the Institute of Computing, UNICAMP. WOODSS interacts with a Geographical Information System and provides model handling facilities, documenting them by means of scientific workflows. The focus of this work is on specifying and implementing new model storing and retrieval modules for WOODSS, using CBR techniques.

The main contributions of this research are: (a) requirement eliciting for using CBR in environmental decision support; (b) development of model management algorithms founded on CBR; and (c) extension of the WOODSS system, making it more suitable for problem solving from precedent cases.

Agradecimentos

Primeiramente, à Deus pelo dom da vida e por conduzir sempre os meus passos com amor supremo, sendo o motivo maior de toda felicidade e existência. O meu tudo.

À minha família por tão grande exemplo e apoio em todas as etapas da minha vida. Este berço acolhedor, mantido pelo amor e pela fidelidade de vocês, pai e mãe, que colhem já os frutos desta bênção. Aos irmãos e irmã, cunhadas e sobrinhas, que juntos formamos esta união, tenham certeza que esta conquista não é só do "cebolinha".

À minha noiva, Isabella, e à sua família pelo apoio e acolhimento. Isa, o seu carinho e a sua compreensão, apesar da distância e da saudade, foram e são fundamentais para mim. Agradeço e dedico este trabalho como uma pequena prova do meu amor por você.

À minha orientadora e amiga Claudia, pelo incentivo, pelos inúmeros ensinamentos e também pelas cobranças, que têm sido fundamentais para a minha formação profissional e pessoal. Estendo estas palavras também à professora e co-orientadora Heloisa.

Aos amigos de casa, André e Curitiba, que fizeram deste tempo longe de casa uma acolhedora e divertida convivência. Agradeço por tudo o que fizeram por mim e contem comigo para o que precisarem. Estou certo de que vamos lembrar destes anos por muito tempo.

A todos os amigos do IC, especialmente à "galera do fut", pelo companheirismo, pelo suporte e pelos momentos agradáveis que passamos juntos. Que a nossa amizade se perpetue por muitos anos.

Aos companheiros do grupo de banco de dados pela ajuda incondicional que recebi de todos sempre que precisei, pelas sugestões de melhoria e pelas críticas construtivas. Agradeço em especial ao Marco, pelo trabalho de implementação.

Ao professor Jansle, da FEAGRI, e aos pesquisadores da Embrapa de Londrina, especialmente ao meu pai, obrigado pelo suporte técnico.

Ao Instituto de Computação da UNICAMP e à Universidade Estadual de Londrina, seus professores e funcionários, obrigado pela oportunidade.

Este trabalho foi desenvolvido com auxílio parcial do CNPq e faz parte do projeto SAI (Sistemas Avançados de Informação) do PRONEX II-MCT.

Conteúdo

Lista de Figuras

Lista de Algoritmos

Capítulo 1

Introdução e Motivação

Uma das áreas em que há grande demanda de soluções de apoio à decisão é o planejamento ambiental, que visa explorar racionalmente os recursos naturais [Fed93]. No contexto da agricultura, por exemplo, envolve determinar o que plantar, onde e quando plantar, como preparar a terra, que técnicas de controle de pragas utilizar e como realizar a irrigação.

Em planejamento ambiental, os processos naturais são usualmente expressados através de modelos, que aproximam e simplificam a realidade, fornecendo uma representação mais estruturada dos fenômenos em estudo [Ste93]. A tomada de decisão neste domínio é inerentemente complexa pois os dados são tipicamente espaço-temporais e requerem um tratamento especial, desde esquemas de armazenamento e operadores específicos até mecanismos de apresentação e análise [DG90, Sno86].

Sistemas Ambientais de Apoio à Decisão (*Environmental Decision Support Systems - EDSS*) e Sistemas Espaciais de Apoio à Decisão (*Spatial Decision Support Systems - SDSS*) são ferramentas que fornecem interfaces e pacotes específicos de software que facilitam a interação do usuário com modelos de análise e dados para gerar e avaliar soluções alternativas nos domínios ambiental e espacial [RY97, CWP95, Den91, SW89]. Geralmente, EDSS e SDSS utilizam Sistemas de Informação Geográfica (SIG) para executar tarefas de gerenciamento e análise de dados espaciais [RY97, Kee97]. SIG são sistemas de informação que permitem coletar, armazenar e recuperar informações baseadas em sua localização espacial e explorar relações entre conjuntos de dados espaciais [Mag91, CCH+96]. Um SDSS aplicado ao domínio ambiental pode ser considerado um EDSS, e será tratado neste texto como um SDSS ambiental.

Um fator chave deste tipo de *software* é a sua habilidade em prover modelos de fenômenos ambientais. Embora SDSS ambientais sejam ferramentas úteis para a construção de soluções no planejamento ambiental, eles oferecem, em geral, modelos genéricos que demandam um considerável esforço de utilização. A codificação de modelos ambientais genéricos tende a ser pouco estruturada e insuficientemente documentada, dificultan-

do a colaboração e a troca de experiências entre grupos de especialistas e resultando em esforços repetidos.

Uma proposta de solução para estes problemas é o sistema WOODSS (*WOrkflOw-based Spatial Decision Support System*) [Sef98, SMRY99], um SDSS ambiental em desenvolvimento no LIS-IC/UNICAMP. O WOODSS interage com um SIG e oferece mecanismos para a manipulação de modelos, documentados sob a forma de um tipo específico de *workflow*: o *workflow* científico [WWVM96, WVMP98]. Estes modelos são armazenados em um banco de modelos para serem reutilizados total ou parcialmente em situações subseqüentes, evitando gastar tempo e recursos com a construção de soluções redundantes. O WOODSS constitui assim uma base para a construção de um ambiente computacional para apoio à decisão no contexto ambiental.

Contudo, problemas ambientais em geral envolvem um conjunto de restrições complexas, dificultando a identificação dos modelos mais adequados para as diferentes situações encontradas. Desta forma é preciso oferecer ao responsável pela tomada de decisão mecanismos que auxiliem na escolha dos modelos corretos. Esta dissertação está voltada a estender o WOODSS com ferramentas que permitam aos usuários recuperar os modelos relevantes para solucionar seus problemas. Estas ferramentas são baseadas na noção de Raciocínio Baseado em Casos (*Case-Based Reasoning - CBR*).

CBR é uma técnica de inteligência artificial que consiste em construir soluções baseando-se em soluções para problemas similares. Mais especificamente, o ciclo de processamento de CBR consiste em: (1) recuperar os casos mais similares ao caso de entrada; (2) reutilizar a solução do caso recuperado, adequando-a para a nova situação; (3) revisar a solução gerada; e (4) reter o novo caso em um banco de casos, para posterior reutilização.

Esta dissertação concentra-se em acoplar CBR ao WOODSS visando permitir a construção de soluções de forma incremental, através da combinação de soluções pré-existentes para problemas semelhantes, para o caso de aplicações agro-ambientais. Nesta abordagem, casos são instâncias de implementação de modelos ambientais em um SIG com um conjunto de metadados associado.

O ponto de partida para o desenvolvimento do trabalho foi um estudo de famílias de problemas agro-ambientais, levantando os requisitos necessários para documentação e indexação de casos, visando antecipar reutilizações futuras e possibilitar a identificação de semelhanças entre situações. Com base nestes requisitos foi proposto um esquema de análise de similaridade entre casos, que diferencia as características dos casos em dois níveis: os seus objetivos e as suas restrições a estes objetivos. Estas características são materializadas em descritores de indexação dos tipos palavras-chave e predicados, formados por valores numéricos e textuais.

A partir do esquema de análise de similaridade proposto foram desenvolvidos algoritmos, baseados em CBR, de recuperação e retenção de casos para o WOODSS. O WO-

ODSS foi completamente recodificado para incorporar estes algoritmos e tornar-se mais aberto a expansões futuras. Esta nova versão do WOODSS possibilita uma recuperação mais precisa, facilitando a identificação dos casos mais relevantes para um novo processo decisório.

A originalidade desta pesquisa reside nos seguintes aspectos: (1) a abordagem aqui adotada visa assistir o usuário na tarefa de construção de modelos ambientais e não envolve a utilização de modelos definidos formalmente, como em outras aplicações; (2) o domínio enfocado é muito mais abrangente do que os sistemas inteligentes em geral, o que dificulta a determinação de regras de similaridade; e (3) existem pouquíssimos trabalhos que combinam as tecnologias aqui utilizadas (CBR, SIG e bancos de dados) para apoio à decisão no domínio agro-ambiental.

As principais contribuições desta dissertação são:

- Levantamento das pré-condições necessárias para a aplicação de CBR para apoio à decisão no contexto ambiental, acoplando este conceito a SIG e bancos de dados;
- Especificação de algoritmos de recuperação e retenção de casos utilizando uma nova proposta de análise de similaridade; e
- Extensão do sistema WOODSS, tornando-o mais efetivo para a manipulação de modelos, através do acoplamento de mecanismos baseados nestes algoritmos, permitindo a resolução de problemas a partir de precedentes.

O restante da dissertação está organizado da seguinte forma. O Capítulo 2 resume os principais conceitos necessários para o entendimento do texto, introduzindo o WOODSS e realizando uma revisão bibliográfica sobre CBR. O Capítulo 3 apresenta a metodologia adotada para o estudo do problema e a descrição da proposta de acoplamento de CBR ao WOODSS nas tarefas de recuperação e armazenamento de modelos. O Capítulo 4 discute alguns aspectos da implementação realizada no WOODSS para permitir a inserção de CBR e para torná-lo mais aberto, visando sua expansão paulatina. O Capítulo 5 dá alguns exemplos de interação do usuário com o sistema em uma aplicação real do domínio de planejamento agrícola. Finalmente, o Capítulo 6 apresenta as conclusões e contribuições da dissertação, propondo algumas extensões ao modelo conceitual proposto e à implementação realizada.

Capítulo 2

Conceitos básicos e revisão bibliográfica

Este capítulo introduz os conceitos básicos ao entendimento desta dissertação. A seção 2.1 apresenta o sistema WOODSS, desenvolvido no LIS-IC/UNICAMP, que é a base para a implementação do mecanismo de recuperação de modelos ambientais propostos nesta dissertação. A seção 2.2 faz um breve apanhado de raciocínio baseado em casos (CBR), concentrando-se em aspectos de recuperação e retenção de casos. A seção 2.3 apresenta alguns trabalhos correlatos encontrados na literatura, no que tange a SDSS ambientais e utilização de CBR em apoio à decisão ambiental.

2.1 O sistema WOODSS

O sistema WOODSS (*WOrkflOw-based spatial Decision Support System*) [Sef98, SMRY99] é baseado em dois conceitos principais: no uso de *workflows* científicos para documentação de processos decisórios no domínio ambiental; e na premissa de que, neste domínio, os processos decisórios envolvem seqüências de atividades que visam a obtenção de mapas como produto final.

O processo decisório ambiental pode ser descrito como uma iteração de quatro passos, que são baseados na metodologia descrita por Pires [Pir97]:

1. Planejamento: envolve a definição dos objetivos da aplicação, da área de estudo e dos modelos a serem utilizados;

2. Inventário: consiste na determinação e coleta dos dados a utilizar;

3. Desenvolvimento: corresponde à implementação, utilizando um SDSS ambiental ou alguma ferramenta similar, dos modelos selecionados usando os dados levantados na

fase de inventário; e

4. Avaliação: compreende a interpretação dos resultados e a tomada de decisão.

Os primeiros três passos deste processo correspondem à geração de um conjunto de mapas, e a etapa de avaliação à análise destes mapas para a tomada de decisão e calibragem dos modelos gerados. Sob a perspectiva de DSS, os mapas resultantes da execução de um modelo são usados para realizar diagnósticos ambientais e detalhar as opções a serem tomadas para solucionar o problema atacado. O processo como um todo é complicado pelo fato de que os modelos variam de acordo com a região geográfica à qual o problema se aplica e pela heterogeneidade dos dados (coletados por diferentes dispositivos, em distintas escalas geográficas e em unidades espaciais e temporais não homogêneas).

Para exemplificar, considere-se o seguinte problema de decisão adaptado de [Sef98] que consiste em encontrar uma área adequada para a instalação de uma fábrica no estado de São Paulo. A área escolhida deve satisfazer os dois critérios seguintes:

(a) Apresentar uma declividade menor que 3 graus; e

(b) Estar fora de uma área de 250 metros ao redor de reservas ambientais.

A resolução deste problema no SIG Idrisi pode se realizar segundo três passos: um para cada critério mencionado e um passo combinando os resultados anteriores. Para isso consideram-se dois mapas de entrada, que representam o relevo e o uso da terra da região escolhida.

(a) Para o cálculo da declividade é preciso desenvolver um mapa de declividades (em graus) da área geográfica a analisar e em seguida efetuar uma reclassificação para criar uma imagem booleana que represente as áreas de declividades maiores e menores que 3 graus. Para a criação do mapa de declividades é preciso usar no Idrisi o módulo *Surface* – opção *Slopes* – e para a reclassificação o módulo *Reclass*.

(b) A área de 250 metros ao redor das zonas de reservas é calculada em três passos: (1) reclassificação de um mapa para criar uma imagem booleana correspondente às áreas de reservas; (2) cálculo da distância de cada ponto da área em estudo à reserva mais próxima; e (3) reclassificação do mapa de distâncias em distâncias maiores e menores que 250 metros. Nos passos (1) e (3) é usado o módulo *Reclass* e no passo (2) o módulo *Distance* do Idrisi.

(c) Finalmente, as duas imagens geradas nos passos (a) e (b) são combinadas em uma operação de sobreposição gerando o mapa solução (módulo *Overlay* do Idrisi).

A figura 2.1 mostra como este processo decisório pode ser documentado por meio de um *workflow*. Neste *workflow*, atividades são representadas por caixas e as dependências são arcos entre as caixas, que representam o fluxo de dados (neste caso, mapas).

As seções subseqüentes apresentam o conceito de *workflows* científicos e descrevem as principais características do WOODSS, bem como suas limitações que motivaram este

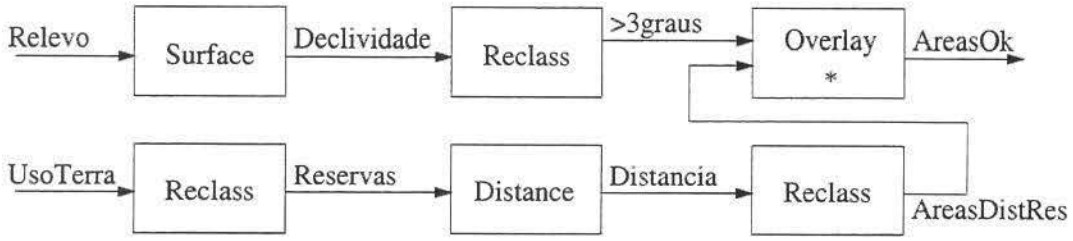

Figura 2.1: Workflow correspondente ao processo de solução do problema.

trabalho.

2.1.1 *Workflows* científicos

Um *workflow* pode ser definido como uma seqüência de passos necessários para atingir um determinado objetivo. Cada passo deste processo é chamado atividade ou tarefa, e pode ser executado por um ou mais agentes. Um agente ou executor é uma pessoa ou componente de software capaz de executar uma ou mais destas tarefas. Através de um papel é possível descrever um determinado (tipo de) agente de acordo com um conjunto pré-estabelecido de habilidades ou conhecimento de contexto necessários à execução de uma tarefa [Bar96]. O objetivo de um sistema de *workflows* é auxiliar na especificação, execução, monitoramento e coordenação de um fluxo de trabalho.

Como ressaltado em [WWVM96], a seqüência de passos de um processo decisório pode ser descrita por meio de um *workflow*, onde cada passo é representado por uma atividade. As informações de entrada e saída (dados, condições de execução, dispositivos a usar, entre outros) e os relacionamentos entre os diferentes passos do processo são expressados pelo fluxo do *workflow*.

Workflows científicos são extensões de sistemas de *workflow* tradicionais, especialmente definidos para documentar procedimentos e experimentos científicos [WWVM96, WVMP98]. A documentação de trabalhos científicos requer um tratamento especial pois este tipo de trabalho é caracterizado por um grande grau de flexibilidade e incerteza e pela ocorrência de exceções.

Workflows científicos estendem sistemas de *workflow* tradicionais nos seguintes aspectos:

- Incompletude: *workflows* científicos podem ser executados até mesmo quando incompletos, sendo construídos progressivamente durante sua execução. *Workflows* convencionais, ao contrário, precisam ser totalmente definidos antes de serem executados;

- Reutilização parcial: *workflows* científicos diferem dos tradicionais pois são conside-

rados blocos em construção para especificação de experimentos. Assim, *workflows* parciais podem ser utilizados para estruturar novos workflows;

- Modificação dinâmica: *workflows* científicos permitem alterar dinamicamente sua especificação inicial, retrocedendo a uma atividade anterior, restabelecendo seu contexto e realizando uma reexecução, possivelmente tomando um novo curso de ação;
- Execução de processos inválidos: no domínio científico, os processos decisórios são baseados no mecanismo de tentativa e erro. *Workflows* científicos são flexíveis o suficiente para suportar este tipo de abordagem;
- Especificação a partir da instância: *workflows* convencionais são especificados para serem executados freqüentemente, onde cada execução é uma instância. *Workflows* científicos, por outro lado, podem ser executados uma única vez, como em tentativas sem sucesso. Uma vez que um *workflow* científico pode ser definido dinamicamente, sua especificação é realizada a partir da instância, ao invés da especificação definir a instância, como em *workflows* tradicionais.

Segundo Seffino et. al. [SMRY99], o processo decisório ambiental pode ser naturalmente caracterizado como um experimento científico, onde o objetivo é produzir um mapa, representando um cenário alternativo, que irá indicar como resolver o problema. Este mapa é gerado a partir da execução de uma seqüência parcialmente ordenada de atividades, seguida de ajustes sucessivos, usando funções de um SIG. Assim, o processo de geração de mapas, e conseqüentemente o processo decisório associado, pode ser apropriadamente modelado como um *workflow* científico.

2.1.2 Visão geral do WOODSS

O WOODSS visa oferecer mecanismos para manipulação e gerenciamento de modelos e suportar a geração de soluções, ou seja, a criação de mapas (passos 1 a 3 do processo decisório ambiental – vide início da seção 2.1). Para alcançar este objetivo, ele interage com um SIG capturando interações dos usuários em atividades de geração de mapas e documentando-as como *workflows* científicos.

Workflows científicos são usados no WOODSS com os seguintes objetivos:

- Para documentação do processo decisório, com isto auxiliando futuras tomadas de decisão;
- Como especificações de modelos de simulação de processos ambientais; e
- Como especificações parametrizadas de procedimentos de decisão, que podem ser reutilizadas e adaptadas para situações semelhantes.

No WOODSS, um *workflow* é uma instância de implementação em um SIG de um modelo genérico (matemático ou empírico), considerando as características espaciais e temporais do problema abordado. Desta forma, pode ser considerado como um modelo em um nível operacional, na linguagem do SIG, que descreve a seqüência de passos utilizada e o fluxo de dados entre estes passos. A cada modelo são associados metadados, que descrevem a semântica do modelo, contextualizando o algoritmo usado no domínio da aplicação, o que é essencial para reutilizações futuras.

Estes modelos são armazenados em um banco de dados, também chamado Banco de Modelos. Sendo assim, os usuários podem consultar o Banco de Modelos, recuperar os modelos mais relevantes, e modificá-los para gerar novas soluções. Os modelos podem ser modificados através da inserção, remoção ou refinamento de atividades e da modificação de dependências e metadados. A apresentação de processos como *workflows* oferece uma visão clara e representativa, onde o usuário pode analisar o processo decisório como um todo, facilitando o controle do processo e o planejamento dos próximos passos. Além disso, o WOODSS oferece mecanismos que facilitam a execução das soluções geradas no SIG.

A figura 2.2 mostra que o usuário pode trabalhar diretamente com um SIG para gerar mapas. Este trabalho é armazenado pelo WOODSS, sob a forma de *workflows* científicos, no banco de modelos. O usuário pode igualmente manipular estes *workflows* (via WOODSS) e solicitar uma execução no SIG. Estes dois tipos de interação do usuário correspondem, respectivamente, à criação e execução de modelos (na figura, interação 1) e à documentação e manipulação de modelos (interação 2), ambas necessárias ao processo decisório.

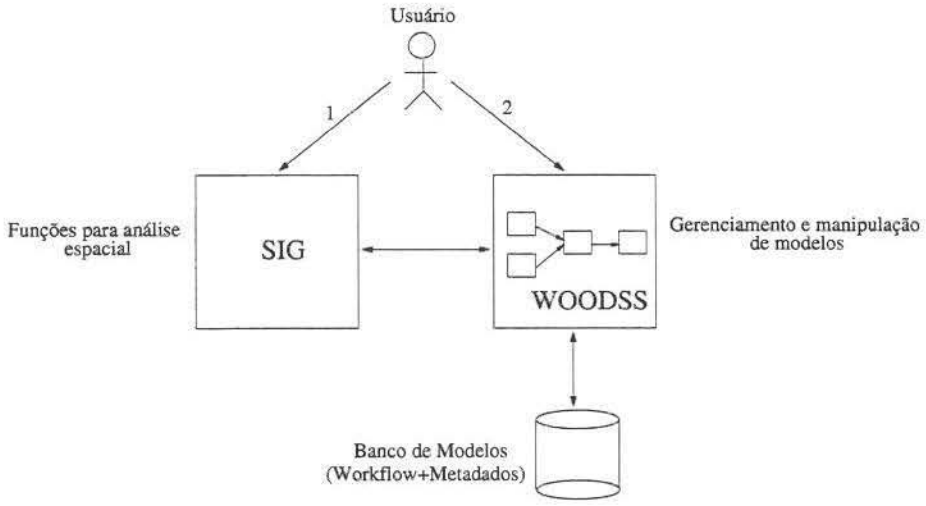

Figura 2.2: Interação do WOODSS com um SIG.

Em sua versão original [Sef98], a arquitetura do WOODSS era formada por cinco módulos: Interface, Monitor, Atualizações, Consultas e Gestor de *workflows* (vide figura 2.3). O módulo Monitor captura as atividades realizadas pelo usuário no SIG e converte esta informação para o formato utilizado pelo gestor de *workflows*. O Gestor de *workflows* gerencia o banco de modelos (codificados como *workflows* científicos). O módulo de Consultas é responsável pela navegação do usuário no banco de modelos. O módulo de Atualizações permite inserir atividades manuais e informações sobre as atividades monitoradas e os dados usados. Por último, a Interface realiza a comunicação dos demais módulos do WOODSS com os usuários.

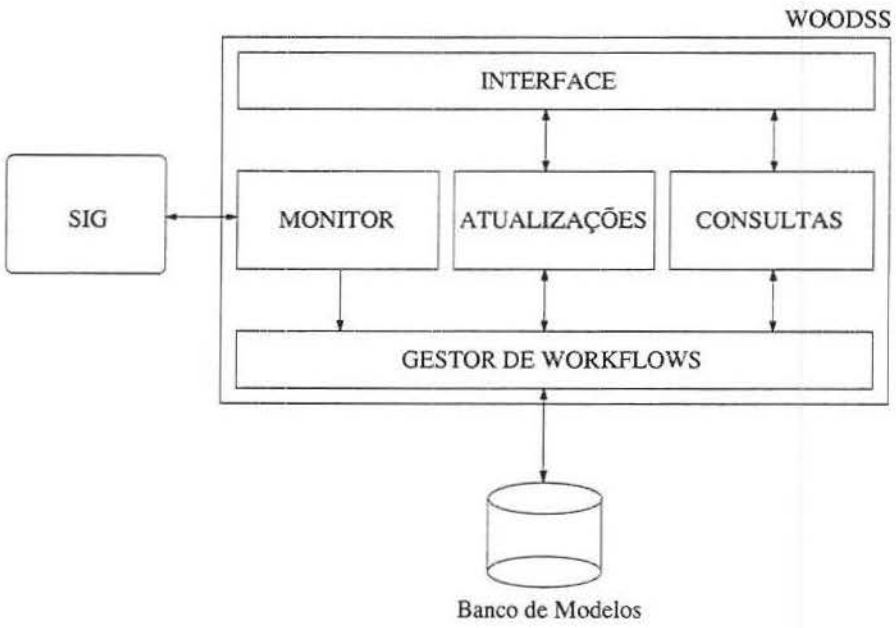

Figura 2.3: Arquitetura original do WOODSS.

2.1.3 Limitações do WOODSS quanto à recuperação de modelos

Algumas limitações do sistema original de recuperação do WOODSS mostraram a necessidade de um novo esquema, capaz de identificar, de forma mais efetiva, os modelos que possam vir a ser relevantes para a solução de um novo problema. Esta constatação motivou o desenvolvimento dos mecanismos descritos nesta dissertação.

O mecanismo de recuperação original do WOODSS previa a recuperação de *workflows* a partir de consultas com casamentos exatos de palavras-chave. A primeira limitação deste esquema é não levar em consideração o conhecimento do domínio para calcular estes casamentos, mas apenas a sintaxe das palavras-chave. Como conseqüência, correspondências simples, como sinônimos e abstrações de conceitos, não são identificadas. Esta

limitação é agravada pelo fato de que supõe-se que vários usuários deverão compartilhar o sistema, provavelmente documentando seus modelos sob diferentes pontos de vista e com vocabulários distintos. Desta forma, inúmeras correspondências semânticas entre os modelos não são capturadas devido às nuances do vocabulário utilizado. Além disso, o casamento entre dois valores é binário (verdadeiro ou falso), o que não reflete o desejo do usuário de encontrar o modelo "mais similar".

Outra limitação é que esta abordagem não pressupõe uma ponderação das características dos modelos quanto à sua relevância. Com isto, características irrelevantes para um determinado problema podem interferir consideravelmente no conjunto de modelos recuperado, retornando muitos modelos sem interesse para o usuário.

A união destes dois problemas tende a forçar o usuário a buscar os modelos desejados puramente através de tentativa-e-erro. O usuário iterativamente vai fornecendo conjuntos de palavras-chave inerentes à situação abordada até encontrar um resultado razoável. Este processo, além de cansativo, pode não levar o usuário a recuperar o modelo mais adequado, especialmente quando há vários modelos referentes a um mesmo tipo de problema, com diferenças sutis, porém fundamentais.

Finalmente, a versão original do WOODSS não fornecia esquemas que permitissem aos usuários descrever de forma adequada os seus modelos. O conjunto de atributos descritivos era bastante incompleto, induzindo a descrição em texto corrido, dificultando o seu processamento para extração do conteúdo. Além disso, as formas de interação do usuário para documentar seus modelos não ofereciam qualquer auxílio, sendo apenas caixas de texto a serem preenchidas. Desta forma, este esquema de documentação dos modelos não incentivava os usuários a fazerem uma boa documentação, resultando em problemas para posterior reutilização.

Visando suprir estas limitações, esta dissertação concentrou-se em aplicar o conceito de Raciocínio Baseado em Casos (apresentado na seção a seguir) para armazenamento e recuperação de modelos no WOODSS. Para isto, foi necessária uma reestruturação global do WOODSS para acomodar o novo mecanismo e para facilitar a tarefa de documentação de modelos, descrita no capítulo 4).

2.2 Raciocínio Baseado em Casos

O Raciocínio Baseado em Casos (*Case-Based Reasoning - CBR*) é um modelo de raciocínio que consiste em solucionar um problema relembrando problemas semelhantes anteriores e adaptando as soluções encontradas para estes problemas à nova situação [RS89].

O CBR é fundamentado no raciocínio natural humano e há evidências de que as pessoas utilizam CBR em seu raciocínio cotidiano [Kol93]. Por exemplo, pessoas em processo de aprendizado freqüentemente procuram relembrar situações prévias para deduzir como

realizar uma determinada tarefa no novo contexto. Por exemplo, os médicos utilizam extensivamente casos anteriores para interpretar os resultados de exames realizados e para selecionar o tratamento mais adequado para um paciente de acordo com seu diagnóstico [KJ91].

O princípio de CBR está baseado no modelo cognitivo proposto por Schank, denominado teoria da memória dinâmica [Sch82]. Esta teoria foi estimulada pelo desejo de compreender como as pessoas lembram-se de informações relevantes para a resolução de problemas. Segundo Schank, a memória humana é dinâmica pois é alterada à medida que novas experiências são encontradas. Os questionamentos provenientes destas experiências e a maneira como foram encontradas as respostas são acrescentados à memória.

Tais experiências são entendidas em CBR como *casos*. Um caso encerra lições aprendidas em um determinado contexto, que podem ser aplicadas a situações recorrentes. Para acomodar as novas experiências, a memória é reorganizada para refletir as lições aprendidas. Os relacionamentos entre experiências são ajustados e os aspectos comuns entre casos são aglutinados na memória através de generalizações e casos compostos, conhecidos como *esquemas* [Kol93].

Nesta estrutura, os casos gravados na memória representam experiências individuais, diferentes do que seria comum ou do que seria esperado. Os esquemas, por outro lado, acomodam as experiências comuns e esperadas que dão uma idéia geral de como determinado tipo de problema pode ser resolvido. O termo "memória" refere-se a algum espaço de endereçamento que armazena os casos de forma a permitir sua recuperação posterior – arquivos, memória principal ou, no contexto da dissertação, estruturas em bancos de dados.

Seguindo este raciocínio, a teoria de Schank [Sch82] propõe estruturas baseadas no conceito de esquemas, denominadas pacotes de organização de memória (*Memory Organization Packets - MOP*). A memória é organizada de forma hierárquica, sendo os MOP a base para esta estrutura, funcionando como repositórios de conhecimento genérico.

Os casos representam conhecimento específico do domínio, que pode ser mais relevante para a resolução de um problema do que um conhecimento genérico que exigiria uma inferência muito maior. Usando como exemplo o domínio de aplicação da dissertação, as entidades de pesquisa agropecuária fornecem tabelas prontas de recomendação de quantidades de fertilizantes para cada tipo de cultura. Uma opção alternativa às tabelas seria o fornecimento de diretrizes para determinar as recomendações adequadas. Embora as duas abordagens ofereçam os dados necessários para determinar as quantidades corretas de fertilizantes, a primeira provê um conhecimento operacional que pode ser usado mais diretamente.

Existem dois tipos básicos de implementação de sistemas CBR: sistemas totalmente automáticos e sistemas de recuperação de informação baseados em casos [KJ91]. Siste-

mas totalmente automáticos resolvem problemas de forma autônoma e têm mecanismos de interação com o mundo para avaliar os resultados de suas decisões. Sistemas de recuperação de informações baseados em casos, por sua vez, subsidiam pessoas que resolvem os problemas, como uma extensão da memória do usuário, a quem cabe realizar o raciocínio e tomar as decisões. É do segundo tipo de implementação que estamos tratando neste trabalho.

As seções subsequentes abordam os conceitos mais importantes referentes à noção de CBR.

2.2.1 CBR e tecnologias de inteligência artificial

Embora seja uma teoria recente, CBR tem sido utilizado sob várias óticas, sendo considerado uma metodologia de raciocínio e não uma nova tecnologia [Wat99]. A definição de raciocínio baseado em casos indica *o que* um sistema deve fazer e não *como* deve fazer.

O princípio de um sistema CBR é resolver um problema tentando reutilizar explicitamente uma solução de um problema passado. Este princípio é materializado em um sistema CBR através da implementação de tecnologias diversas, como raciocínio baseado em regras [LFK97, SV95], raciocínio baseado em modelos [Kot89, BH94], algoritmos genéticos [WI97, SH99], lógica fuzzy [Che97], redes neurais [Den96] e outras.

Contudo, existem algumas características conceituais de CBR que podem justificar e motivar sua utilização sob tecnologias de inteligência artificial. As principais características são: obtenção de conhecimento, modelagem a partir de instâncias, robustez e método natural de raciocínio.

Obtenção de conhecimento: uma característica muito importante de CBR é o acoplamento entre os processos de resolução de problemas e de aprendizado. A comunidade de pesquisa em aprendizado de máquina tem sido uma grande propulsora dos métodos de CBR, considerando até mesmo CBR como uma linha de aprendizado de máquina [AP94]. A razão é que o aprendizado em CBR é uma tarefa natural, proveniente da aquisição de novos casos.

A aquisição de conhecimento em sistemas baseados em regras, por exemplo, exige alterar regras. A alteração de uma regra pode resultar na necessidade de ajustes em várias outras para resolver conflitos, provocando um efeito cascata que pode ser muito grande.

Sob este prisma, CBR não somente denota um método particular de raciocínio, mas também um paradigma de aprendizado de máquina que possibilita a aquisição de conhecimento através da atualização da memória de casos após a resolução de um problema [AP94].

Modelagem a partir de instâncias: em CBR o conhecimento do domínio é modelado extensivamente através dos casos armazenados na memória [AP94]. Este conhecimento é construído a partir dos conhecimentos específicos fornecidos pelos casos. Desta forma, CBR permite lidar com domínios complexos, onde não existe um modelo formal que descreve o comportamento do domínio, facilitando a acomodação de exceções e anomalias.

Robustez: sistemas baseados em casos são robustos no que se refere a descrições incompletas de problemas. Um sistema CBR ocupa-se em encontrar um caso armazenado que satisfaça parcialmente as restrições apresentadas e gera uma solução alternativa [KJ91]. Já sistemas baseados em regras, por exemplo, não podem resolver problemas que não se encaixam exatamente em nenhuma regra.

Método natural de raciocínio: é geralmente mais fácil para especialistas articular soluções relembrando soluções para problemas encontrados do que realizando inferências sobre um conjunto potencialmente grande de regras. Relembrar situações vividas também permite evitar erros cometidos anteriormente.

Além disso, o fato de basear-se em soluções que tiveram sucesso em situações anteriores fornece justificativas razoavelmente satisfatórias para as decisões tomadas [Kol93]. Em contrapartida, por exemplo, o processo interno de raciocínio de uma rede neural é invisível ao usuário, sendo guiado pelos pesos associados às conexões entre os neurônios [Wat97]. Assim, não é fácil justificar um resultado de uma inferência em uma rede neural.

2.2.2 Ciclo de processamento de CBR

O ciclo básico de processamento de CBR pode ser descrito genericamente como: dado um problema, obter as soluções anteriores relevantes, adaptá-las para o problema atual e armazenar o novo caso, juntamente com sua solução. Aamodt e Plaza [AP94] apresentam este ciclo considerando quatro grandes processos (os 4 *RE*): *REcuperar, REutilizar, REvisar e REter* (figura 2.4):

1. **Recuperar:** analisar o caso de entrada extraindo descritores relevantes e recuperar, utilizando estes descritores, os casos mais similares ao caso de entrada;
2. **Reutilizar:** construir novas soluções a partir de soluções recuperadas e/ou partes destas, através de ajustes e adaptações;
3. **Revisar:** avaliar e testar a solução construída para determinar sua corretude, utilidade e robustez;
4. **Reter:** acrescentar o novo caso à memória;

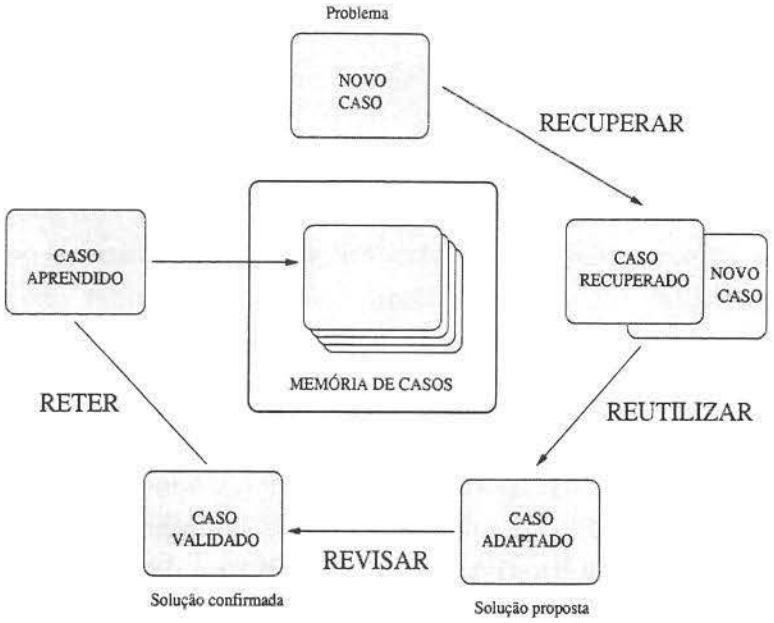

Figura 2.4: Ciclo básico de processamento em CBR.

Uma descrição inicial de um problema define um novo caso. Baseando-se nesta descrição, a memória de casos é pesquisada em busca de casos úteis para a situação atual. O caso mais relevante recuperado é combinado com o novo caso, realizando as adaptações necessárias para a reutilização. O caso gerado a partir desta adaptação é denominado *caso solução*. O caso solução passa por uma etapa de revisão, onde é testado em um ambiente real ou em um modelo de simulação, ou ainda analisado por um especialista. Uma vez validado, o caso é retido para futuras reutilizações.

Esta dissertação visa utilizar CBR para selecionar os modelos mais relevantes para uma situação de planejamento ambiental, deixando para o usuário a responsabilidade de reutilizar adequadamente os modelos. Sendo assim, apenas para organizar as idéias contidas no restante deste capítulo, os processos de reutilização e revisão serão introduzidos primeiramente, e, em seguida, os processos de recuperação e retenção serão apresentados com maior ênfase.

Reutilização

Em linhas gerais, o processo de reutilização concentra-se em dois aspectos: identificar as diferenças entre o caso recuperado e o novo caso, e quais partes do caso recuperado podem ser transferidas para o novo caso.

Exemplos triviais de reutilização são situações em que as características do novo caso correspondem às de um armazenado. Nesta situação, basta reutilizar a solução do caso

antigo. Tarefas de classificação (identificar a qual das classes existentes um caso pertence, considerando suas características) são exemplos típicos desta forma de reutilização. Por exemplo, um sistema para identificar tipos de solos baseando-se em suas características poderia ser modelado como um sistema CBR de classificação.

Quando não existe uma solução direta para o novo caso é preciso adaptar a solução que mais se aproxima de uma solução factível. Para suportar o processo de adaptação, Watson destaca, em [Wat99], diversas tecnologias: sistemas baseados em regras, algoritmos genéticos, programação funcional, técnicas baseadas em restrições, entre outras.

Existem dois tipos principais de adaptação: (1) adaptação derivacional e (2) adaptação transformacional. A *adaptação derivacional* consiste em recuperar o método de solução do caso mais relevante e re-instanciá-lo no novo contexto. Os casos devem conter informações sobre o método de solução utilizado, incluindo os operadores usados, os objetivos considerados e as alternativas geradas. Este tipo de "adaptação" assemelha-se a uma tarefa de classificação, onde o método de solução do caso antigo é transferido para o novo caso e re-executado.

A *adaptação transformacional* consiste em modificar a solução do caso recuperado adequando-a à nova situação. A reutilização transformacional não considera como um problema foi resolvido, mas sim a equivalência de soluções. Este tipo de adaptação exige um amplo conhecimento do domínio para definir transformações capazes de adequar uma solução antiga a um novo panorama. Devido à dificuldade de definir estas transformações, os sistemas que utilizam este tipo de adaptação são aplicados a domínios restritos e relativamente bem conhecidos.

Revisão

A fase de revisão consiste em avaliar o caso solução gerado na fase de reutilização e, se a solução gerada não for satisfatória, procurar repará-la [Kol93].

Uma solução pode ser avaliada de várias formas: considerando o resultado da sua aplicação no ambiente real, através de simulações, ou pela análise de um especialista do domínio. Assim, o processo de avaliação deve estar acoplado a dispositivos e sensores reais, a mecanismos de simulação, ou depender do respaldo do usuário.

Se a solução não for satisfatória, pode ser abandonada ou corrigida. Em ambas as situações, os erros identificados devem ser apontados e justificados para evitar sua repetição. A solução errônea é iterativamente reparada e submetida a uma nova revisão até que seja corrigida ou sejam exauridas as possibilidades de reparo.

Recuperação e retenção

Os processos de recuperação e retenção de casos em CBR são fortemente relacionados. A efetividade da recuperação depende da forma como os casos foram armazenados.

No processo de retenção são definidas quais informações do novo caso serão armazenadas, de que forma integrar o novo caso à estrutura de memória e como indexar o caso para posterior recuperação. Antes de armazenar um novo caso, é necessário decidir se a relevância deste caso justifica o seu armazenamento. O processo de recuperação inicia com uma descrição parcial do problema e termina com o caso mais semelhante ao problema apresentado dentre os casos armazenados [Wat97].

Ambos os processos precisam pesquisar a memória de casos e identificar onde inserir o novo caso (retenção) ou recuperar os casos mais próximos (recuperação). Esta pesquisa é baseada nas características do caso de entrada e guiada por métricas de similaridade, que são dependentes do domínio da aplicação. A figura 2.5 apresenta um paralelo entre estes dois processos.

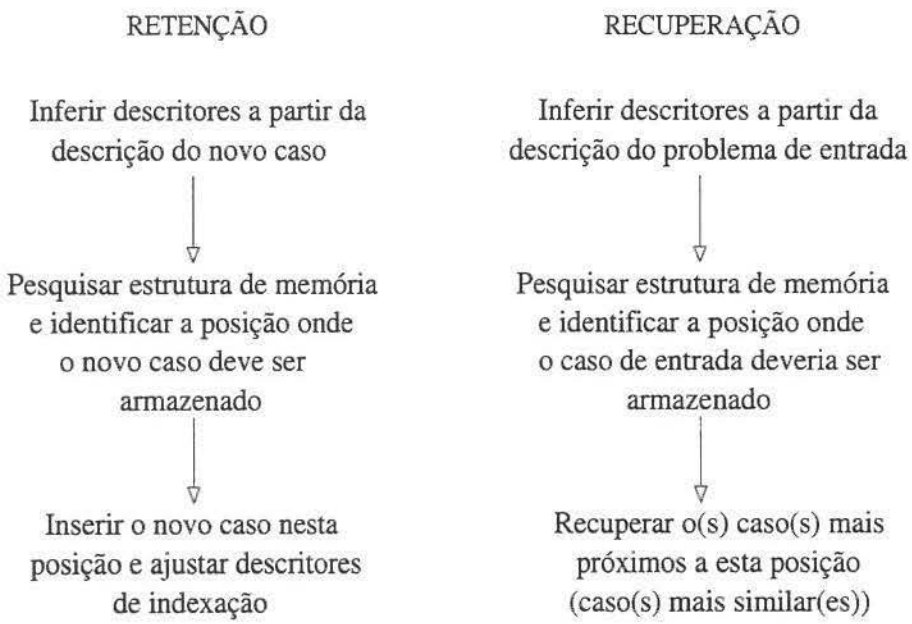

Figura 2.5: Paralelo entre os processos de recuperação e retenção de casos.

O primeiro passo para desenvolver um mecanismo eficaz de recuperação de casos é conhecer a semântica dos casos. Assim, os casos devem ser representados explicitando sua semântica. A representação dos casos fornece informações que serão utilizadas em todo o ciclo de processamento de CBR.

A partir desta representação são extraídos os descritores que são relevantes para que os casos sejam recuperados nas situações esperadas. Entretanto, não é uma tarefa fácil

escolher descritores que contextualizem o significado dos casos. Os seres humanos naturalmente inferem semelhanças entre situações e relembram informações úteis para abordar uma nova situação. Estas lembranças são muitas vezes inesperadas e é difícil explicar como tal lembrança foi alcançada.

Os descritores são entradas para os algoritmos de busca. A estrutura da memória de casos é construída incrementalmente, particionando conceitualmente o conjunto de casos baseando-se nos seus descritores. Os algoritmos de busca decidem o percurso na estrutura analisando a similaridade semântica entre descritores. Finalmente, o caso mais similar é recuperado ou o novo caso é armazenado.

As seções subsequentes discutem aspectos e diretrizes genéricas para os processos de recuperação e retenção introduzidos nesta seção. O primeiro aspecto apresentado é a representação de casos. Em seguida, são considerados a escolha de descritores e alguns métodos propostos. Fechando a seção, são apresentadas algumas técnicas para análise de similaridade entre casos e estruturas de memória de casos.

2.2.3 Representação de casos

Um caso representa um conhecimento específico do domínio. O conhecimento embutido em um caso são as lições que ele ensina e o contexto destas lições. Basicamente, a representação do conteúdo de um caso é composta de três partes: a descrição do problema, a solução desenvolvida e o resultado obtido com a solução construída [Kol93].

Descrição do problema

A descrição do problema apresenta uma situação e o estado do mundo no momento em que esta situação acontece. Existem três componentes principais da descrição do problema: (1) os objetivos a serem alcançados, (2) as restrições referentes a estes objetivos e (3) características complementares do problema.

Os objetivos refletem o propósito do caso e guiam o processo de solução considerando as restrições apresentadas. Podem ser divididos em sub-objetivos em uma abordagem de *divisão-e-conquista*. As restrições descrevem as circunstâncias que devem ser consideradas para alcançar um determinado objetivo. As características complementares do problema são informações úteis para alcançar os objetivos, porém não fundamentais. Enquanto os objetivos e as restrições são inter-dependentes, as características complementares apenas enriquecem a descrição do caso.

Solução desenvolvida

No processo de reutilização, o sistema CBR baseia-se na solução de outros problemas. Algumas informações, além da solução em si, podem auxiliar o processo de adaptação. Estas informações incluem: o conjunto de passos usados na construção da solução (método de solução); o conjunto de justificativas para as decisões tomadas no desenvolvimento da solução; as potenciais soluções não escolhidas e a razão pela qual não foram escolhidas; e as soluções descartadas e as justificativas do seu descarte.

O conjunto de passos de raciocínio possibilita o processo de adaptação derivacional. As justificativas de decisões no decorrer do processo de solução ressaltam as características que levaram ao raciocínio utilizado. As potenciais alternativas não escolhidas, as alternativas descartadas e as justificativas da não utilização destas alternativas podem ser úteis na construção de novas soluções, pois as novas situações em geral possuem restrições diferentes.

Resultado obtido

O resultado de uma solução indica se as expectativas do usuário foram satisfeitas com a solução construída. Estas informações podem auxiliar o sistema a alcançar determinados objetivos e a evitar futuros fracassos. Desta maneira, muitas vezes é importante armazenar também casos sem sucesso.

As informações contidas no resultado obtido incluem: em que aspectos a solução satisfez ou não a expectativa do usuário; a justificativa do resultado obtido; o que pode ser feito para evitar novos fracassos (antecipar potenciais problemas); e um apontador para outra solução com sucesso, caso a solução tenha fracassado.

2.2.4 Métodos para determinação de descritores

O problema fundamental em CBR é a recuperação de casos apropriados. Para desenvolver um mecanismo eficaz de raciocínio baseado em casos, idealmente é preciso construir estruturas e algoritmos que simulem o comportamento humano de inferência de semelhanças. Este problema é conhecido, na literatura de CBR, como o *problema da indexação* [Kol93].

O problema da indexação engloba dois subproblemas:

- Definir que características utilizar para indexar um caso de forma que ele possa ser recuperado nas circunstâncias adequadas; e
- Definir uma estrutura de organização para a memória de casos que permita que o processo de busca seja eficiente e preciso.

Descritores são lembretes que apontam para um determinado caso, indicando em que circunstâncias ele pode ser útil. Por exemplo, um caso de avaliação da aptidão agrícola das terras de Campinas poderia ter como descritores `avaliação da aptidão agrícola de terras` e `Campinas`, explicitando a tarefa que o caso encerra e a região onde foi realizada.

O processo de recuperação depende fortemente da qualidade dos descritores atribuídos aos casos. Uma vez que um mesmo caso pode ser relembrado de variadas formas é necessário definir descritores que garantam a recuperação dos casos corretos nas diversas situações que o sistema CBR enfrenta. Há várias propostas para extrair e organizar as características dos casos, antecipando o vocabulário e as circunstâncias de potenciais reutilizações futuras.

Mesmo quando um domínio já foi analisado em sua quase totalidade, um contra-exemplo pode ser descoberto, resultando em uma falha no vocabulário de descritores. Portanto, a intenção é definir um vocabulário que funcione na maior parte das vezes, embora não exista forma de garantir que um vocabulário seja livre de falhas.

Em geral, bons descritores vão além das características superficiais do caso. Desta forma, é preciso identificar as características funcionais do caso e da sua solução e compreender os inter-relacionamentos implícitos. Segundo diretrizes sugeridas por Kolodner [Kol93], bons descritores devem antecipar lições que podem ser úteis em raciocínios posteriores, apontando como alcançar determinados objetivos e/ou como evitar possíveis falhas.

Os sistemas CBR em geral possuem um conjunto pré-definido de descritores determinado pelo desenvolvedor do sistema. Um conjunto fixo e bem especificado de descritores facilita muito a implementação de um sistema CBR, embora seja impossível prever todos os possíveis descritores. Esta limitação é sentida quando o domínio não é bem conhecido. A seguir são apresentados dois métodos de determinação de descritores bastante utilizados em sistemas CBR [Kol93]: métodos baseados em justificativas e métodos baseados em diferenças.

Métodos baseados em justificativas

Métodos baseados em justificativas utilizam explicações de porque uma solução funcionou ou fracassou como base para escolher descritores. A hipótese que sustenta este método é que explicações tendem a predizer as lições que um caso pode ensinar. Desta forma, descritores extraídos deste tipo de explicações tendem a ser preditivos para cada situação em particular.

O processo de seleção de descritores baseado em explicações consiste em gerar uma explicação do sucesso ou fracasso de uma solução e escolher descritores a partir do conteúdo da explicação. Os passos deste processo são detalhados como:

1. Gerar uma explicação dos resultados da solução: as explicações podem incluir, por exemplo, as justificativas das decisões tomadas durante o processo de solução e porque uma falha aconteceu. As justificativas podem ser apontadas pelo usuário ou geradas automaticamente pelo sistema. Uma explicação pode ser desenvolvida de várias maneiras, incluindo prova de teoremas e explicações baseadas em acontecimentos prévios;

2. Selecionar os descritores relevantes da explicação: associar descritores aos objetivos e tarefas de raciocínio aos quais correspondem para permitir a recuperação do caso utilizando estas características;

3. Generalizar estes descritores.

Segundo Kolodner [Kol93], a maior vantagem deste método é que ele permite escolher indicadores melhores para contextualizar as lições trazidas pelos casos. Os descritores escolhidos são as características que foram responsáveis para alcançar um resultado (falha ou sucesso), que se encontradas novamente devem predizer o potencial resultado do caso armazenado. Este método é computacionalmente complexo e requer um amplo conhecimento do domínio da aplicação.

Métodos baseados em diferenças

O procedimento de seleção de índices baseado em diferenças concentra-se em extrair diferenças entre o novo caso e os casos armazenados. A filosofia deste método é que as peculiaridades de um caso são as características que induzem a sua lembrança. Desta forma, os descritores selecionados por este método são extraídos das anomalias e características particulares de um caso para distingui-lo dos demais.

Os passos do processo de seleção de descritores baseado em diferenças são:

1. Identificar o conjunto de características do novo caso que o diferenciam dos outros casos;

2. Extrair descritores que retratem estas diferenças, dentro do contexto do problema; e

3. Generalizar estes descritores.

Um problema deste método é que nem sempre os atributos que diferenciam casos uns dos outros são preditivos [Kol93]. Por exemplo, utilizar como descritor de um caso de recomendação de adubação o nome do agrônomo responsável pode diferenciá-lo dos demais, mas provavelmente não é um índice preditivo.

Por outro lado, ressaltar as diferenças entre os casos armazenados tende a diminuir a redundância e a guiar o processo de recuperação. Desta forma, os métodos usualmente são combinados para selecionar descritores preditivos e diferenciadores.

2.2.5 Análise de similaridade entre casos

Sistemas de CBR visam distinguir dentre os casos armazenados aquele (ou aqueles) que é potencialmente mais útil. Para isto, é necessário calcular a similaridade semântica entre casos. A tarefa de avaliação de similaridade entre casos pode ser dividida em: (1) encontrar correspondências, (2) calcular o grau de similaridade entre os descritores correspondentes e (3) ponderar os descritores dos casos de acordo com o contexto.

1. Encontrar correspondências: o primeiro passo do processo de análise de similaridade é identificar os descritores da nova situação e de casos armazenados que possuem a mesma regra funcional.

Em grande parte das vezes, descritores do tipo <atributo, valor> têm a mesma regra funcional quando seus atributos possuem o mesmo nome. Por exemplo, os descritores `temperatura=alta` e `temperatura=baixa` se referem ao mesmo fenômeno (temperatura), e podem ser comparados diretamente.

Igualdade de nomes de atributos, no entanto, não é o único critério para comparar descritores. Por exemplo, os descritores `temperatura=baixa` e `latitude=alta` podem indicar os mesmos fenômenos, pois regiões em altas latitudes possuem baixas temperaturas. Considere ainda um caso com o descritor `número de pessoas=10` e outro com os descritores `número de mulheres=4` e `número de homens=5`. Neste caso, os atributos do segundo caso deveriam ser mapeados em um único atributo somando os seus valores antes de realizar a comparação.

Para encontrar correspondências semânticas mais profundas entre descritores é necessário explorar o conhecimento do domínio, possivelmente sob a forma de uma rede conceitual, com abstrações e relacionamentos entre conceitos. A maioria dos sistemas CBR implementados não possui esta capacidade.

2. Calcular o grau de similaridade entre descritores: existem várias formas propostas para comparar a similaridade entre descritores de casos. Algumas das técnicas mais freqüentes são:

- Diferença: consiste em calcular a norma da diferença entre dois valores numéricos. Quanto menor o valor obtido, maior a similaridade entre os descritores. Este método pode ser aplicado a descritores não numéricos, desde que possam ser mapeados para valores numéricos;

- Faixas de valores: esta técnica particiona o domínio de um descritor em faixas qualitativamente iguais. A similaridade entre valores é calculada como a distância entre as faixas a que os valores pertencem. O principal problema de utilizar faixas de valores é a mudança abrupta de valor nas extremidades do intervalo.

- Lógica fuzzy: uma função fuzzy é usada para transformar um valor quantificável de um atributo em uma descrição qualitativa que pode ser comparada com a descrição qualitativa de outros atributos [Men95]. Funções fuzzy sofrem mudanças suaves nos limites entre valores qualitativos, ao contrário da técnica anterior. Um sistema implementado que utiliza lógica fuzzy na análise de similaridade entre casos é descrito em [Che97]. É um sistema CBR, criado na General Electric, para determinar quais colorantes utilizar para produzir cores específicas de plásticos.

- Microatributos: têm uma relação de agregação (*parte de*) com o atributo considerado. Valores do atributo `solo`, por exemplo, podem ser comparados considerando suas características, como `textura` e `rocha formadora`.

- Hierarquia de abstração: neste caso, um par de valores tem maior similaridade do que outro par se a *abstração comum mais específica* do primeiro par é mais específica do que a do segundo. Considerando na hierarquia apresentada na figura 2.6 os pares `<periquito, canário>` e `<periquito, cachorro>`, a abstração comum mais específica do primeiro é `pássaro` e a do segundo `animal`. Como `pássaro` é mais específico na hierarquia do que `animal`, o primeiro par possui maior similaridade. Um problema desta abordagem é que dois itens podem ser abstraídos de várias maneiras diferentes, gerando inconsistências nos valores de similaridade. Isto pode ser contornado ponderando as abstrações de acordo com as circunstâncias em questão.

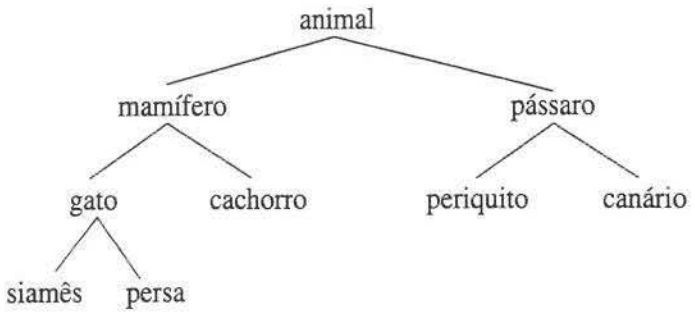

Figura 2.6: Exemplo de hierarquia de conceitos.

3. Ponderar a importância dos descritores: uma função de similaridade deve ponderar a importância de cada atributo para evitar que atributos irrelevantes comprometam o resultado. Um especialista do domínio pode atribuir estes pesos no momento da inserção do caso na memória. Ele deve ressaltar quais dimensões e combinações de dimensões fazem boas predições. Outra forma consiste em realizar uma avaliação estatística de um conjunto conhecido de casos para determinar quais dimensões melhor predizem diferentes resultados e/ou soluções. A estas dimensões são atribuídos maiores pesos de importância.

Algoritmo de vizinhos mais próximos

Um tipo de algoritmo de recuperação muito utilizado em CBR é o de vizinhos mais próximos (*k-Nearest Neighbors*) [Kol93]. Algoritmos de vizinhos mais próximos partem do pressuposto que cada caso é definido por um conjunto de n atributos (simbólicos ou numéricos). A cada atributo é associado um peso, representando sua relevância no problema. O cálculo é obtido através de uma função semelhante à função a seguir, que especifica o valor da similaridade entre o caso de entrada (E) e um caso armazenado (A), onde f é a função de similaridade entre atributos e w_i o peso da similaridade entre os descritores Ei e A_i no cálculo da similaridade total. O valor de similaridade é usualmente normalizado para valores entre 0 e 1 ou valores percentuais (onde os valores máximo de similaridade são 1 e 100%, respectivamente).

$$Similaridade(E, A) = \sum_{i=1}^{n} w_i \times f(E_i, A_i)$$

Este cálculo é repetido para todos os casos da memória de casos, gerando uma ordenação da similaridade com o caso de entrada. Outra limitação deste método é que as correspondências entre os atributos são fixas. Desta forma, correspondências mais profundas são desprezadas.

Embora o algoritmo de vizinhos mais próximos possua as limitações citadas, a maioria dos sistemas CBR utiliza alguma variante deste algoritmo [Wat97].

2.2.6 Estruturas de memória de casos

As estruturas de memória de casos propostas na literatura refletem esforços para simular a estrutura da memória humana e possibilitar que o processo de raciocínio seja o mais natural possível. Além disso, estas estruturas procuram particionar conceitualmente o conjunto de casos, para que casos irrelevantes possam ser descartados durante a busca. Esta seção apresenta duas estruturas de memória bastante influentes em CBR: o modelo da memória dinâmica e modelo de categoria e exemplar.

Modelo da memória dinâmica

O modelo de memória dinâmica é baseado na teoria da memória dinâmica de Schank [Sch82] e foi implementada no primeiro sistema CBR referenciado na literatura, o sistema CYRUS [Kol84].

Trata-se de uma estrutura hierárquica de *pacotes de organização de memória* – conceito proveniente da teoria de Schank. A idéia básica é organizar casos específicos que compartilham propriedades similares em uma estrutura mais genérica, denominada *episódio generalizado*, que possui três tipos diferentes de objetos: *normas*, *casos* e *índices*. Normas são descritores comuns a todos os casos do episódio generalizado e índices são descritores que diferenciam estes casos entre si. Um índice pode apontar para um episódio generalizado mais específico ou diretamente para um caso. Um índice é composto por dois termos: um atributo e um valor.

A figura 2.7 ilustra um episódio generalizado complexo (episódio 1), que engloba casos e um episódio generalizado mais específico (episódio 2). Os nós desta estrutura são episódios generalizados, casos, atributos de índice ou valores de índice. Um índice pode apontar para somente um caso ou um episódio. O esquema de indexação é redundante, pois existem múltiplos caminhos para um caso particular ou um episódio generalizado (caso1, na figura 2.7) .

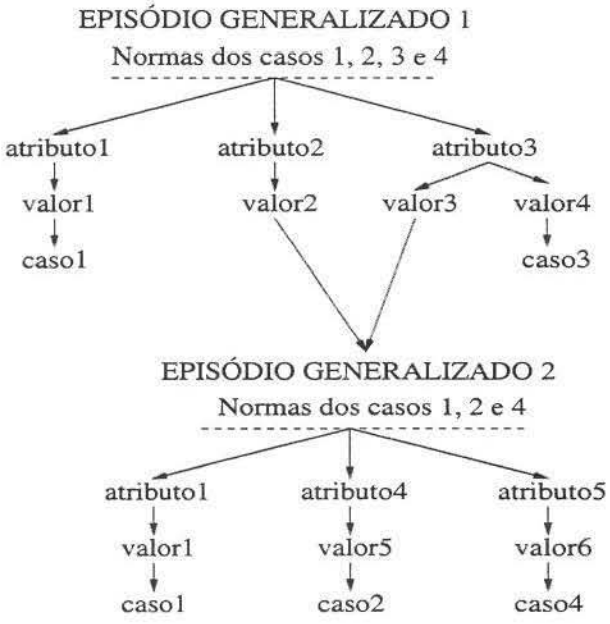

Figura 2.7: Uma estrutura de memória dinâmica.

Um caso é recuperado encontrando o episódio generalizado com a maior quantidade de normas em comum com o caso de entrada. Os índices abaixo deste episódio indi-

cam qual dos casos é o mais similar. O processo de retenção segue o mesmo princípio, transformando em índices os atributos diferenciadores do caso de entrada em relação ao episódio. Se existir outro caso que compartilhe algum destes descritores, um novo episódio é dinamicamente criado neste nível, e os casos são depositados abaixo dele.

A estrutura de memória dinâmica pode conduzir a uma explosão no número de descritores com o crescimento do conjunto de casos [Kol93]. A maioria dos sistemas que implementam esta estrutura de memória limita a escolha de descritores [AP94]. O sistema CYRUS, por exemplo, permite apenas um pequeno vocabulário de descritores.

Modelo de categoria e exemplar

Uma estrutura de memória de casos alternativa foi introduzida por Bareiss, no sistema PROTOS [Bar89], de diagnóstico de distúrbios auditivos. Os casos são referenciados como exemplares e são identificados como pertencentes a uma categoria pelo seu conjunto de descritores.

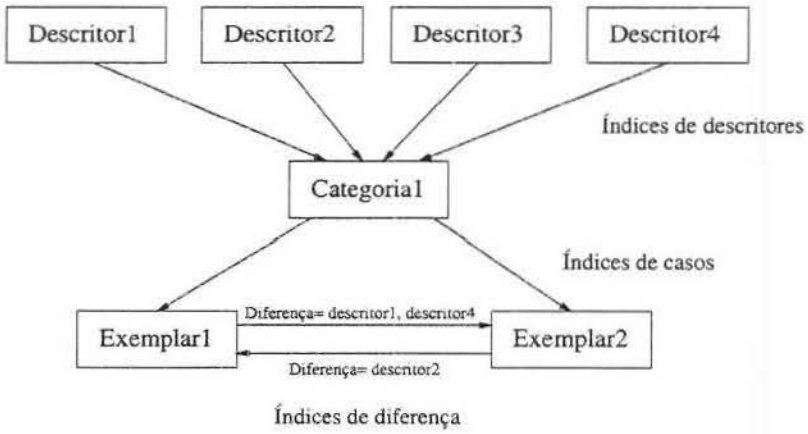

Figura 2.8: Estrutura de memória categoria e exemplar.

A figura 2.8 apresenta a idéia desta estrutura. A memória de casos é embutida em uma rede semântica que interrelaciona os descritores de casos, as diferentes categorias e os casos. Estes relacionamentos são expressos através de índices, que podem ser de três tipos:

- Índices de descritores: são apontadores (ou lembretes) de descritores para categorias ou casos;
- Índices de casos: apontadores de categorias para os casos associados; e
- Índices de diferença: são apontadores entre casos vizinhos que diferem em um pequeno conjunto de descritores.

O processo de busca de um caso nesta estrutura percorre os índices que correspondem aos descritores do caso de entrada, encontrando a categoria que compartilha mais descritores. O caso mais similar é encontrado percorrendo os índices de casos desta categoria. Os índices de diferença indicam soluções alternativas que suportam possíveis adaptações ou re-seleções de casos.

2.3 Trabalhos correlatos

Esta dissertação combina técnicas de inteligência artificial (CBR) e SDSS ambientais. Existem inúmeras propostas de utilização de mecanismos de inteligência artificial em aplicações ambientais (vide exemplos de revisões bibliográficas em [CP91, MD88]).

Mais especificamente na área agro-ambiental, algumas propostas recentes são, por exemplo [SB98, MSC99, JFU95]. Shafer e Brodahl [SB98] descrevem um sistema baseado em regras para apoio à decisão no domínio agrícola que simula o efeito de regras simples definidas pelo produtor frente a eventos climáticos e mudanças na relação solo-cultura no tempo e no espaço. Um editor específico permite que sejam inseridas novas regras para guiar as simulações, realizadas em um software de apoio à decisão denominado GPFFARM (*Great Plains Framework for Agricultural Resource Management*).

Matthews et. al. [MSC99] apresentam a implementação de um SDSS ambiental para tarefas de planejamento de uso de solo, incorporando módulos de avaliação de impacto ambiental. Este sistema combina um SIG com um KBS (*Knowledge Based System*), utilizando a noção de algoritmos genéticos.

Embora estas propostas utilizem técnicas de inteligência artificial para apoio à decisão ambiental, fornecem apenas suporte para a instanciação de modelos pré-definidos em situações diferentes. Jacucci et. al. [JFU95] apresentam uma abordagem mais direcionada à geração de modelos, adaptando-os a novas situações, guiando-se pelas diferenças entre as regiões de estudo usando algoritmos genéticos. Contudo, as adaptações suportadas por estas abordagens são restritas a variações induzidas por algumas poucas características climáticas da região em estudo.

Existem também algumas abordagens que utilizam CBR em aplicações de apoio à decisão ambiental, como [HB99, VB99, Has96]. Hastings [Has96] apresenta um sistema de notificação de alertas de infestações de gafanhotos combinando CBR e raciocínio baseado em modelos (*Model-Based Reasoning*). O processo decisório utilizado consiste em estimar a proporção de folhagem que será consumida pelos gafanhotos, avaliar se a infestação pode afetar as plantações e, caso isto ocorra, estimar os custos e benefícios das medidas a serem tomadas. O sistema, denominado CARMA (*CAse-based Range Management Adviser*), não utiliza SIG.

Verdenius e Broeze [VB99] apresentam um sistema baseado em CBR aplicado ao

domínio de tratamento de águas de esgoto utilizando plantas e microorganismos. Durante o processo de transformação de substâncias tóxicas, existe uma grande demanda de oxigênio. Para que a tarefa de purificação da água tenha um desempenho ótimo, uma técnica utilizada consiste em injetar doses de oxigênio na água, de acordo com a necessidade. A idéia do sistema consiste em gerenciar o nível de oxigênio e decidir, nas variadas situações, que medidas tomar. Este sistema também não utiliza SIG.

Uma aplicação que combina CBR e SIG é apresentada por Holt e Benwell em [HB99]. Nesta referência, os autores descrevem um sistema de classificação de solos denominado ZONATION. A proposta deste sistema reside no fato de que, segundo os autores, não existe uma fórmula genérica que pode ser aplicada para classificação de séries de solos para uma região genérica. O sistema permite aos especialistas realizar classificações baseadas em instâncias prévias, utilizando conhecimento específico do domínio.

Todos os trabalhos correlatos citados implementam modelos genéricos pré-definidos, utilizando técnicas de inteligência artificial. O enfoque da dissertação é auxiliar o usuário a desenvolver novos modelos e executá-los em um SIG, de acordo com sua necessidade, não sendo restrito apenas à execução de modelos genéricos com parâmetros diferentes. Além disso, a inserção de sistemas inteligentes em aplicações ambientais considera domínios restritos e bem definidos. Uma das maiores dificuldades da abordagem desta dissertação reside na abrangência do domínio enfocado e na conseqüente flexibilidade necessária dos mecanismos de gerenciamento de modelos.

Desta forma, a proposta descrita nesta dissertação é original, pois: (1) a abordagem adotada visa assistir o usuário na tarefa de construção de modelos ambientais e não envolve a utilização de modelos definidos formalmente, como nas aplicações citadas; (2) o domínio enfocado é muito mais abrangente do que os sistemas inteligentes em geral, o que dificulta a determinação de regras de similaridade; e (3) existem pouquíssimos trabalhos que combinam as tecnologias aqui utilizadas (CBR, SIG e bancos de dados) para apoio à decisão no domínio agro-ambiental.

2.4 Resumo

Este capítulo apresentou os conceitos necessários ao entendimento da dissertação. Inicialmente, descreveu o sistema WOODSS, usado como base para a implementação de CBR, como se vê nos capítulos 3 e 4. A seguir, apresentou uma revisão sobre CBR, enfatizando as fases de recuperação e retenção. Finalmente, citou alguns trabalhos correlatos.

O próximo capítulo descreve a proposta de como acoplar CBR ao WOODSS para apoio à decisão no domínio agro-ambiental. Esta proposta utiliza as noções de vizinhos mais próximos e determinação de descritores baseada em diferenças, utilizando hierarquias de abstração e diferenças de valores para a organização e comparação de descritores.

Capítulo 3

Raciocínio baseado em casos no WOODSS

A primeira etapa de um processo decisório ambiental é a etapa de planejamento, que compreende definir os objetivos da aplicação e a área de estudo e, com base nestes dados, escolher o modelo e o conjunto de fontes de dados a serem utilizados [Pir97]. A extensão do WOODSS, utilizando CBR, tem por objetivo auxiliar o usuário a encontrar correspondências semânticas entre um novo problema e os modelos armazenados visando escolher o modelo mais adequado à situação. Considerando cada modelo ambiental implementado um caso, a identificação de modelos recai no processo de recuperação de casos em CBR.

Como introduzido na seção 2.2, para desenvolver um mecanismo de recuperação de casos em CBR é necessário definir: (a) como representar os casos, (b) que descritores utilizar para indexá-los, (c) como calcular a similaridade entre casos e (d) em que estruturas de dados armazená-los.

Este capítulo apresenta os requisitos dos mecanismos de recuperação e retenção de modelos proposto para o WOODSS. A seção 3.1 apresenta o referencial teórico para a utilização de CBR nesta dissertação. A seção 3.2 mostra a metodologia empregada nesta pesquisa. A seção 3.3 especifica a proposta de representação de casos no WOODSS. A seção 3.4 discute a determinação de descritores para indexação dos casos. A seção 3.5 descreve o algoritmo de análise de similaridade entre casos. As seções 3.6 e 3.7 descrevem, respectivamente, os algoritmos propostos de retenção e recuperação de casos.

3.1 CBR em modelagem e apoio à decisão ambiental: motivações teóricas

A qualidade das decisões geradas em um sistema de apoio à decisão depende diretamente dos modelos utilizados na sua construção. Modelos simulam, ou imitam, as operações e reações de vários tipos de processos do mundo real. Processos ambientais são complexos, pois tipicamente possuem uma série de características inter-relacionadas, dependentes do tempo e do espaço (localização) [Ste93]. Desta forma, dificilmente podem ser expressados de forma exata, embora modelos matemáticos e estatísticos sejam largamente utilizados para o seu entendimento e previsão.

Frente a este panorama, os especialistas procuram aproximar a realidade, fornecendo uma representação estruturada dos fenômenos em estudo, construindo modelos fortemente baseados em resultados experimentais [Ste93]. O processo de construção de modelos é, portanto, fundamentalmente empírico e pode ser descrito como:

1. Observar o comportamento do fenômeno ambiental em estudo, identificando correlações entre as diferentes variáveis envolvidas;

2. Recorrer a conhecimentos prévios de fenômenos semelhantes para gerar hipóteses que possam justificar as correlações identificadas;

3. Desenvolver experimentos para testar as hipóteses geradas e retornar ao primeiro passo até que os resultados sejam justificados.

4. Generalizar as hipóteses confirmadas, materializando esta generalização na especificação de um modelo.

Analisando este processo genérico, é possível identificar características que motivam a utilização de CBR para a construção de modelos, e, consequentemente, seu uso nesta dissertação.

Em primeiro lugar, muitas vezes as hipóteses confirmadas são insuficientes para determinar as generalizações necessárias e construir um modelo completo. Algumas generalizações são factíveis, mas em geral as situações caracterizam-se como exceções e casos especiais. CBR aparece como uma alternativa, pois em CBR o conhecimento não é modelado formalmente, mas definido extensivamente, através de seu conjunto de instâncias [AP94]. Cada um destes modelos (instâncias) é conhecido como modelo de instância específica, e o modelo genérico fica implícito no conjunto [VB99]. Não existem desvios condicionais no modelo genérico, há um modelo de instância específica para cada situação. Desta maneira, CBR pode suportar o armazenamento de instâncias de processos aleatórios, bem

como seus estados em instantes diferentes, e a construção de modelos do domínio através destes conjuntos de instâncias.

Além disso, a aquisição de conhecimento em CBR faz parte do processo natural de construção de soluções. Assim, o conhecimento de sistemas ambientais e das técnicas de análise utilizadas, em contínua evolução, pode ser acomodado através do armazenamento de novos casos na memória de casos. Isto ajuda a contornar a complexidade dos processos ambientais e facilita a acomodação de exceções e anomalias, que podem ser tratadas de maneira análoga a situações comuns.

Em segundo lugar, os especialistas em modelagem ambiental baseiam-se em conhecimentos sobre fenômenos e comportamentos que tenham alguma semelhança com a nova situação (modelagem a partir de precedentes). Esta característica recai no princípio básico de CBR, que recorre a situações antigas procurando subsídios para enfrentar uma nova situação.

Finalmente, CBR pode ser utilizado para prover mecanismos que auxiliem os responsáveis por tomadas de decisão a escolher os modelos adequados para o seu problema. Esta é a motivação principal que levou à escolha de CBR na reestruturação do WOODSS. Rizzoli et al. [RY97] apresentam um conjunto de características genéricas desejáveis de um sistema de apoio à decisão ambiental. A necessidade de cada uma delas em um sistema particular irá depender da natureza do problema e dos usuários do sistema. Estas características são:

1. Capacidade de adquirir, representar e estruturar o conhecimento do domínio em estudo. Para isto, é preciso identificar o domínio da aplicação, estabelecer quais são as estruturas de dados necessárias para representar os componentes do domínio modelado e entender os relacionamentos entre eles;

2. Disponibilização de um banco de modelos, que armazena o conhecimento estruturado do domínio, permitindo a separação entre dados e modelos para facilitar a reutilização de modelos e a prototipagem;

3. Capacidade de lidar com dados espaciais;

4. Possibilidade de oferecer ajuda especialista no domínio de interesse, através de técnicas de inteligência artificial. Conhecimentos do domínio auxiliam o usuário a recuperar informações, preparar dados, selecionar modelos e interpretar resultados;

5. Oferecimento de mecanismos para auxiliar o usuário a formular o problema e escolher os métodos de solução.

No caso específico do WOODSS, nota-se que ele oferece inicialmente as três primeiras características sugeridas:

Item 1 Os *workflows* no WOODSS permitem representar e estruturar o conhecimento do domínio em estudo, representado através dos modelos. Além disso, com o mecanismo de captura de interações dos usuários com o SIG, o WOODSS permite adquirir conhecimentos automaticamente. Os relacionamentos entre as tarefas e sub-processos são apresentados naturalmente através dos workflows;

Item 2 O banco de modelos é provido pelo WOODSS; e

Item 3 O SIG acoplado ao WOODSS fornece as funções de manipulação e análise de dados espaciais.

Com a inserção de CBR, o WOODSS passa a ter características desejáveis de um SDSS ambiental até então não oferecidas:

Item 4 A utilização de CBR na recuperação e retenção de modelos torna o WOODSS sensível ao domínio e auxilia o especialista na formulação do problema e na escolha dos métodos de solução (modelos) para tarefas de planejamento ambiental; e

Item 5 Através do conhecimento do domínio embutido no conjunto de casos e do mecanismo de similaridade parcial entre casos, o sistema pode identificar semelhanças entre os requisitos do usuário e os modelos armazenados, retornando os mais adequados.

Sob esta ótica, o acoplamento de CBR ao WOODSS permite a construção de um SDSS ambiental adequado. As seções a seguir especificam a inserção de técnicas de CBR no WOODSS.

3.2 Metodologia empregada

A área de planejamento ambiental engloba uma série de subáreas. Alguns exemplos são: avaliação de impacto ambiental, avaliação de aptidão de terras, alocação de recursos, controle de erosão e de fontes poluidoras, recuperação de áreas degradadas e agricultura de precisão. Embora o WOODSS seja genérico o suficiente para apoiar estes problemas, o seu mecanismo de manipulação de modelos deve ser sensível ao contexto da aplicação.

Devido à abrangência da área, a metodologia utilizada para propor e especificar o mecanismo de recuperação do WOODSS baseado em CBR, foi a seguinte:

1. Realização de um estudo em largura de problemas de planejamento ambiental, visando levantar semelhanças e divergências entre os processos de modelagem e documentação de modelos de diferentes tipos de problemas;

2. Escolha de uma família representativa de problemas para um estudo mais detalhado;

3. Determinação dos requisitos para esta família de problemas, identificando suas necessidades operacionais e particularidades;

4. Validação dos requisitos levantados;

5. Especificação de novos mecanismos de recuperação e retenção para o WOODSS, baseados em CBR; e

6. Implementação do mecanismo proposto.

A família de problemas escolhida para estudo de caso foi a de *recomendação de fertilizantes e corretivos em taxa variável.* Esta família situa-se dentro da subárea de *Agricultura de Precisão.*

A Agricultura de Precisão (ou *tecnologia de taxa variável*) consiste no manejo diferenciado de áreas dentro de um campo de produção, visando racionalizar o uso de insumos e o consumo de energia e aumentar a produtividade [QDM00]. Os dados coletados são espacializados em mapas para representar a variabilidade da região em estudo. Estes mapas servem como base para a geração de mapas para aplicação localizada de insumos, que são fornecidos para implementos agrícolas especiais que efetivamente realizam a aplicação [SCH00]. O processo é realimentado com a avaliação final dos resultados. Em que pesem as principais tecnologias envolvidas neste sistema, ressaltam-se: SIG, sensoriamento remoto, e sistemas de posicionamento global (*GPS - Global Positioning System*).

Dentro da agricultura de precisão são encontradas várias famílias de problemas, incluindo recomendação e aplicação de fertilizantes e corretivos, controle de pragas e aplicação de defensivos, operações de preparo do solo, irrigação e planejamento de colheita. Dentre estas famílias de problemas, esta dissertação tomou como exemplo de estudo a que lida com recomendação de fertilizantes e corretivos. A próxima seção introduz este conjunto de problemas e justifica a sua escolha.

3.2.1 Recomendação de fertilizantes e corretivos em taxa variável

O solo sofre alterações constantes, tendo seus nutrientes consumidos pelas plantas ou perdidos por lixiviação e erosão. Embora uma parte dos nutrientes fique retida no solo,

ou seja liberada com o tempo através da decomposição orgânica, as exigências das culturas geralmente superam a disponibilidade natural de nutrientes no solo [LGM99]. As principais medidas para atingir e manter boas condições nutricionais para o cultivo são a calagem e a adubação.

Em solos com pH excessivamente ácido ocorre a diminuição na disponibilidade de alguns nutrientes fundamentais para o desenvolvimento das culturas e aumento demasiado de outros, atingindo níveis tóxicos às plantas. A *calagem* é a prática da aplicação e incorporação de calcário ou de qualquer outro material com o objetivo de elevar o pH do solo, neutralizando a acidez, e também de suprir alguns nutrientes [CNP00].

A *adubação* consiste em adicionar nutrientes ao solo para corrigir e evitar deficiências. O cálculo da quantidade de adubo necessário considera os teores dos nutrientes no solo e a quantidade esperada de extração pela cultura. A prática de adubação mais utilizada nas lavouras é a de macronutrientes primários (nitrogênio, fósforo e potássio – adubação NPK). Isto não significa que sejam mais importantes do que os outros nutrientes, mas que são consumidos em maiores quantidades pelas plantas, tornando-se escassos mais rapidamente e exigindo reposições maiores.

As recomendações de calagem e adubação são específicas de acordo com a cultura e a região, pois consideram as exigências nutricionais da cultura e as características climáticas da região, visando alcançar a *máxima produtividade econômica*, que é o ponto ótimo da relação entre o gasto com insumos e o retorno da venda da produção.

Existem várias fórmulas utilizadas para o cálculo da necessidade de calagem. No Brasil, cada região utiliza uma fórmula preferencial, pelas características ambientais da região (clima, tipo de solo) ou por influência dos laboratórios locais. O cálculo das quantidades de fertilizantes é baseado em tabelas geradas por institutos de pesquisa agropecuária, que indicam a quantidade de acordo com os teores dos nutrientes no solo e a produtividade esperada da cultura. A produtividade esperada da cultura é baseada nas colheitas passadas dos últimos anos. Quando não existe esta informação histórica, é feita uma estimativa considerando-se as características da área. A figura 3.1 mostra uma tabela de recomendação para a cultura da soja[1] no estado de São Paulo, fornecida pelo Instituto Agronômico de Campinas (IAC) [RCQF96].

O processo convencional de recomendação de fertilizantes e corretivos baseia-se na média das quantidades de nutrientes encontradas nas amostras de solo coletadas da área, desconsiderando a distribuição espacial dos nutrientes. A recomendação em taxa variável fundamenta-se em conjuntos de amostras georeferenciadas que mostram a distribuição

[1]No caso da soja e de algumas outras leguminosas e oleaginosas (por exemplo, amendoim e grão-de-bico), o nitrogênio é fornecido para a planta através da fixação simbiótica, que ocorre com bactérias do gênero Bradyrhizobium, que captam nitrogênio da atmosfera. Assim, em geral, não é utilizada a adubação nitrogenada, mas apenas a inoculação das sementes, que é a mistura de uma pequena quantidade de fertilizantes diretamente às sementes.

Produtividade esperada	P resina, mg /dm ^{3}				K^+ trocável, mmol /dm ^{3}			
	0-6	7-15	16-40	>40	0-0.7	0.8-1.5	1.6-3.0	>3.0
t/ha	P_2O_5 kg/ha				K_2O kg /ha			
1.5-1.9	50	40	30	20	60	40	20	0
2.0-2.4	60	50	40	20	70	50	30	20
2.5-2.9	80	60	40	20	70	50	50	20
3.0-3.4	90	70	50	30	80	60	50	30
3.5-4.0	*	80	50	40	80	60	60	40

* Não é possível obter esta produtividade em solos com teores muito baixos de P.

Figura 3.1: Tabela de recomendação de fertilizantes NPK para a soja em São Paulo.

dos nutrientes, permitindo recomendações localizadas mais precisas.

Este conjunto de problemas justifica a utilização de CBR, dada a dificuldade de identificar o caso mais adequado, visto a quantidade de variáveis e de combinações possíveis. Os principais fatores que correspondem a variáveis embutidas na recomendação de fertilizantes e corretivos são: a cultura, a região, a textura do solo, as aplicações anteriores de insumos, o estágio na rotação de culturas e o tipo de cultivo em questão. Em números, considerando 50 culturas, 10 regiões com recomendações diferentes e 5 variações da mesma cultura em uma mesma região (que não são números exagerados), tem-se um conjunto de 2500 casos. Note que para cada caso existe um histórico particular de aplicações e rotação de culturas que implica variações adicionais.

O tópico de recomendação de fertilizantes e corretivos já se encontra em um estágio maduro de pesquisa. Assim sendo, existe uma bibliografia extensa que serviu como base para este trabalho ([RCQF96, dFdSR94, CNP00, dPdT99, CPA97], dentre outras).

Finalmente, a Agricultura de Precisão é uma área de pesquisa de ponta, com uma grande demanda de soluções de apoio à decisão. Neste contexto, os resultados desta dissertação podem ser aplicados sob vários outros pontos de vista dentro desta área.

3.3 Representação de casos

Conforme citado na seção 2.2.3, a estrutura de um caso em CBR em geral é constituída por três componentes: o problema, a solução desenvolvida e o resultado obtido. No WOODSS não existe uma separação clara entre problema e solução. A solução desenvolvida é representada pelo *workflow*, a descrição do problema é a descrição deste mesmo *workflow*, e os resultados também fazem parte da descrição do *workflow*.

Esta dissertação propõe que os casos no WOODSS sejam representados por dois com-

ponentes: (A) o *workflow* e (B) a descrição associada. A partir deste ponto, o termo *modelo* será utilizado exclusivamente para modelos ambientais genéricos e o termo *caso* para instanciações de modelos genéricos, documentadas como *workflows* científicos no WOODSS.

Cada problema de planejamento ambiental possui uma forma particular de descrição. Analisando vários problemas do domínio, identificou-se um conjunto de informações descritivas genéricas (metadados) comuns a quase todos eles. Baseando-se nesta análise, a descrição dos casos proposta para o WOODSS é formada pelo seguinte conjunto de campos:

- Os objetivos do caso;
- Os modelos ambientais genéricos utilizados;
- Uma descrição da área de estudo;
- Uma descrição dos dados de entrada;
- Os efeitos das decisões tomadas sobre os mapas resultantes; e
- Os dados sobre o especialista ou órgão responsável.

Os *objetivos* resumem o problema atacado, incluindo as restrições impostas. O campo *modelos* introduz os modelos utilizados e cita suas referências. A *descrição* da área inclui a localização geográfica e as características ambientais da região (por exemplo, clima, relevo, solo, vegetação). A *descrição dos dados de entrada* basicamente elenca os dados utilizados, citando a metodologia para seu levantamento e a data da coleta. Os *efeitos* indicam o sucesso ou fracasso das medidas tomadas com base nos mapas resultantes do caso, as limitações identificadas e suas justificativas. Finalmente, os dados sobre o especialista ou órgão responsável têm caráter de referência.

3.4 Determinação de descritores

A proposta do WOODSS é possibilitar a construção de novos modelos baseando-se em adaptações de modelos prévios. Ao documentar um novo modelo, o usuário geralmente ressalta as características ou restrições faltantes (ou em excesso) no modelo de origem que levaram-no a modificá-lo. Este panorama sugere determinar descritores de indexação para os casos usando a idéia do *método de determinação de descritores baseado em diferenças*, descrito na seção (2.2.4).

Relembrando, os passos do processo de determinação de descritores deste método são:

- Identificar o conjunto de características do novo caso que o diferenciam dos casos já armazenados;
- Extrair descritores que retratem estas diferenças, dentro do contexto do problema; e
- Generalizar estes descritores.

A questão é: que informações extraídas dos casos poderiam ser bons descritores para indexá-los? Idealmente, o sistema deveria realizar este processo de forma automática. Contudo, a complexidade de tal abordagem é muito grande. Assim, no contexto deste trabalho, os usuários são os responsáveis por determinar os descritores dos casos. Esta seção define os tipos de descritores de casos propostos para o WOODSS.

Primeiramente, considere a componente *workflow* dos casos. Chegou-se à conclusão de que o fluxo de atividades não oferece grande subsídio para a interpretação da semântica do caso, se considerado independente do contexto da aplicação. Os exemplos apresentados na seqüência vêm suportar esta hipótese.

Considere que uma determinada cultura obtém maior produção quando implantada em solos de um tipo hipotético x e com teores de potássio (K^+) acima de um valor y. Deseja-se identificar quais áreas de uma dada região possuem estas características, ideais para tal cultura. Um modelo para solução deste problema consiste em (1) a partir do mapa de solos da região, selecionar as áreas onde o solo é do tipo x (em um SIG, operação de reclasssificação), (2) a partir do mapa da distribuição de K^+, selecionar as áreas com teor acima de y (reclassificação) e (3) combinar os mapas (binários) obtidos, buscando a interseção destas características (operação de sobreposição). A figura 3.2 apresenta o workflow resultante da implementação deste modelo no SIG Idrisi.

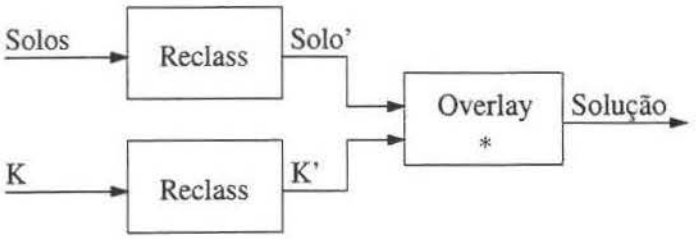

Figura 3.2: Exemplo de identificação de áreas ideais para uma determinada cultura.

Considere agora outro problema, para determinar a quantidade de fertilizantes potássicos a ser recomendada para uma determinada cultura, a partir de um tabela de recomendação de quantidades de fertilizantes idêntica à tabela da figura 3.1. Suponha que áreas com certo tipo de solo x devem receber o dobro da dose da tabela de recomendação. Um modelo para este problema seria (1) processar o mapa de teores de potássio, atribuindo

as doses recomendadas na tabela, (2) processar o mapa de solos atribuindo valor 2 (dobro) às áreas com solo do tipo x e valor 1 às demais áreas (reclassificação) e (3) combinar os dois mapas (sobreposição ponderada utilizando uma função de multiplicação). O *workflow* da figura 3.3 corresponde à especificação deste modelo.

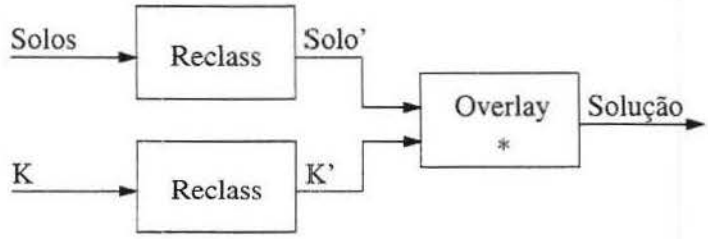

Figura 3.3: Exemplo de recomendação de quantidades de fertilizantes.

Note que os *workflows* das figuras 3.2 e 3.3 têm estruturas idênticas e invocam as mesmas funções do SIG, mas têm propósitos completamente diferentes. Ambos utilizam o mesmo fluxo de operações, mas chegam a resultados sem qualquer relação. Em alguns casos, a sucessão de determinadas operações pode indicar algum significado, mas este fato não é uma regra. Talvez uma análise detalhada de uma grande gama de modelos, com propósitos diversos, possibilite chegar a conclusões concretas neste sentido. Contudo, esta abordagem foge ao escopo desta dissertação devido à sua complexidade.

Relembrando, propõe-se documentar casos a partir do par <*workflow*, descrição>, a partir do qual se extraem descritores para indexá-los. Descartada a possibilidade do uso da "topologia"dos *workflows*, resta a opção de extrair descritores dos casos do seu componente *descrição*.

Analisando o conjunto de problemas de planejamento ambiental que o WOODSS pode apoiar, verificam-se dois níveis de diferenciação na descrição dos casos: o nível de *objetivos* e o nível de *restrições* associadas a estes objetivos. Considera-se que um fator é uma restrição a um objetivo quando limita o contexto em que um objetivo deve ser atingido. Por exemplo, "controle de pragas e aplicação de defensivos" é um objetivo global, enquanto as restrições típicas incluem fatores espaço-temporais e sócio-econômicos.

No nível de objetivos, as diferenças entre casos são mais explícitas. É visível que é pequena a similaridade entre casos de recomendação de fertilizantes em taxa variável e casos de análise da aptidão de terras. Entretanto, no nível de restrições estas diferenças já não são tão claras, exigindo maior nível de detalhe. Por exemplo, a recomendação de fertilizantes varia de acordo com a cultura, a região e a textura do solo em questão. Estas restrições são refletidas, na implementação do modelo, em variações de pesos, valores de parâmetros e até mesmo na estrutura do *workflow*.

Com base nesta constatação, são propostos dois tipos de descritores:

Tipo 1 Palavras-chave. São as que o usuário determina para descrever os objetivos do caso; e

Tipo 2 Restrições aos objetivos. São especificadas como predicados associados aos objetivos. Cada predicado é expresso como conjunções de triplas da forma <atributo, comparando, valor>. Os atributos são restritos aos tipos *string* e numérico, sendo que no caso em que o atributo é do tipo *string*, o comparando se restringe à igualdade (=). Um exemplo deste descritor é $< cultura, '=', soja > \wedge < textura\_solo, '¿', 30\% >$.

Cada tripla possui um peso de relevância com relação a um determinado objetivo. Uma mesma tripla pode ter pesos diferentes quando presente em restrições relacionadas a objetivos distintos. Estes pesos são fornecidos pelos usuários no momento da atribuição de uma restrição a um objetivo. Um caso pode ter vários objetivos, com restrições específicas associadas a cada um destes objetivos.

Os elementos do tipo *string* destes descritores (palavras-chave, nomes de atributos e valores textuais das restrições) formam um dicionário, que é expandido à medida que os especialistas inserem novos termos. Para situar o contexto destes termos no domínio, o dicionário é organizado como um conjunto de árvores n-árias, que seguem a idéia das hierarquias de abstração, apresentadas na seção 2.2.5.

Esta noção já vem sendo utilizada há anos por especialistas da área agrícola para padronização de vocabulário. Em 1997 foi publicado a segunda edição de um vocabulário controlado de termos agrícolas – Thesagro [CEN97] – contendo quase dez mil termos agrícolas, contextualizados através de relações hierárquicas, associativas e de equivalência. A figura 3.4 mostra como o termo *soja perene* é descrito no Thesagro.

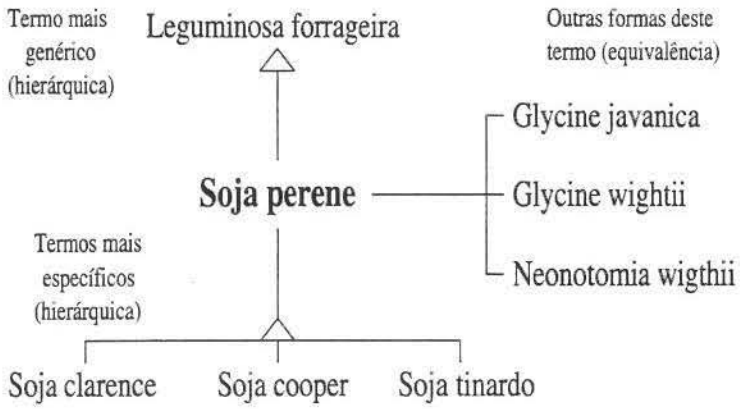

Figura 3.4: Hierarquia de conceitos do termo *soja perene* no Thesagro.

O dicionário básico utilizado neste trabalho é baseado no Thesagro. Os usuários, ao inserir um novo termo ao dicionário, devem situá-lo na hierarquia de termos e fazer os

ajustes necessários para acomodá-lo. Esta estrutura permite generalizar os descritores dos casos, além de servir como base para a análise de similaridade entre casos.

3.5 Análise de similaridade entre casos no WOODSS

O método de análise de similaridade entre casos proposto utiliza a idéia da função de *vizinhos mais próximos*, introduzida na seção 2.2.5. O algoritmo desenvolvido calcula a similaridade entre casos (sim[caso$_1$, caso$_2$]) como a média ponderada das similaridades dos seus objetivos (simObj[caso$_1$, caso$_2$]) e das suas restrições (simRestr[caso$_1$, caso$_2$]). A similaridade dos objetivos é priorizada sobre as restrições porque é mais relevante para o usuário obter um caso com finalidades semelhantes às desejadas, ainda que as restrições desejadas não sejam satisfeitas, do que um caso que satisfaça estas restrições em um contexto diferente.

Para especificar os algoritmos, será usada a notação que se segue. Conjuntos são expressados com iniciais maiúsculas ($Obj[e]$), elementos de conjuntos com uma letra minúscula (o), variáveis com inicial minúscula ($simObj$) e subrotinas em letras maiúsculas ($SIMOBJ()$).

O algoritmo de análise de similaridade entre casos proposto para o WOODSS é o algoritmo 3.1, que utiliza três subrotinas, descritas a seguir:

CAMINHO. Compara atributos do tipo *string* a partir dos caminhos entre termos nas hierarquias do dicionário. A subrotina CAMINHO considera apenas o comparando '='. O valor de similaridade entre dois termos (CAMINHO($string_1$, $string_2$)), é dado por:

$$CAMINHO(string_1, string_2) = 1 - \frac{numero\_arestas\_caminho\_entre\_string_1\_string_2}{numero\_maximo\_arestas}$$

O valor *numero_maximo_arestas* é definido como o máximo de arestas que o caminho pode ter, para limitar a procura. Se algum dos termos não se encontra na hierarquia ou se o caminho entre eles é maior que o máximo definido, CAMINHO($string_1$, $string_2$)=0. Caso estes termos sejam equivalentes, CAMINHO($string_1$, $string_2$)=1.

SIMOBJ. Compara descritores do tipo 1 (palavras-chave), utilizando a subrotina CAMINHO. Sejam o_1 e o_2 dois objetivos dados e $D[o_1]$ e $D[o_2]$ seus respectivos conjuntos de descritores, SIMOBJ(o_1, o_2) é dado pelo algoritmo 3.2.

DIFERENÇA. Calcula a diferença entre atributos do tipo numérico. Para o cálculo de DIFERENÇA, há três situações, de acordo como os comparandos fornecidos (comparando$_1$, comparando$_2$):

Algoritmo 3.1 Algoritmo de análise de similaridade proposto para o WOODSS.

Entrada: i (caso de entrada), O (conjunto de objetivos armazenados) e
C (conjunto de casos armazenados)
Saída: $\texttt{sim}[i, c]$ (valor de similaridade entre i-$c, \forall c \in C$)

1. Calcular a similaridade entre objetivos do caso de entrada e objetivos armazenados

```
para cada o ∈ O faça
  simObj[o_i, o] ← SIMOBJ(o_i, o);
onde o_i ∈ Obj[i] e Obj[i] é o conjunto de objetivos de i.
```

Seja $O' \subset O$ o conjunto de objetivos armazenados com $\texttt{simObj}[o_i, o] > 0$, $\forall o \in O'$, e $C[O']$ o conjunto de casos armazenados onde $\exists o_c \in Obj[c]$, $o_c \in O'$

2. Calcular as similaridade de objetivos ($\texttt{simObj}[i, c]$) e restrições ($\texttt{simRestr}[i, c]$) entre i-c, onde $c \in C[O']$

```
para cada c ∈ C[O'] faça {
  simObj[i, c] ← 0;
  simRestr[i, c] ← 0;
  para cada o_c ∈ Obj[c] ∩ O' faça {
    simObj[i, c] ← simObj[i, c]+ SIMOBJ(o_i, o_C);
    para cada r_i ∈ Restr[i, o_i] faça
      para cada r_c ∈ Restr[c, o_c] faça
        se r_i.atributo = r_c.atributo então
          se r_i.tipo = string então
            simRestr[i, c] ← simRestr[i, c]+CAMINHO(r_i.valor, r_c.valor);
          senão
            se r_i.tipo = numérico então
              simRestr[i, c] ← simRestr[i, c]
                +DIFERENÇA([r_i.comp, r_i.valor], [r_c.comp, r_c.valor]);
  }
}
```

3. Calcular a similaridade final ($\text{sim}[i, c]$) entre i-c

```
para cada c ∈ C, tal que simObj[i, c] > 0 faça
  sim[i, c] ← (P_1× simObj[i, c] + P_2× simRestr[i, c])/(P_1 + P_2); onde P_1 > P_2
```

Algoritmo 3.2 Algoritmo de análise de similaridade entre objetivos de casos.

Entrada: o_1 e o_2 (objetivos a serem comparados)
Saída: SIMOBJ (valor de similaridade entre os objetivos dados)

```
SIMOBJ ← 0;
para cada d_i ∈ D[o_1] faça {
  para cada d_j ∈ D[o_2] faça
    SIMOBJ ← SIMOBJ + CAMINHO(d_i, d_j);
}
SIMOBJ ← SIMOBJ / numero_descritores_D[o_1];
```

1. Ambos comparando$_1$ e comparando$_2$ são '='. O valor de similaridade é dado pela diferença entre os valores:

$$DIFERENÇA([=, valor_1], [=, valor_2]) = 1 - \frac{| valor_1 - valor_2 |}{maior\_diferenca}$$

2. Comparando$_1$ ou comparando$_2$ é '>' ou '<'. Neste caso, se o intervalo dado engloba o outro valor, DIFERENÇA=1. Caso contrário, o valor de similaridade é dado pela diferença entre o limite do intervalo e o outro valor, como no caso anterior.

3. Ambos comparando$_1$ e comparando$_2$ são '>' ou '<'. Se existe interseção entre os intervalos, DIFERENÇA=1. Caso contrário, o valor de similaridade é dado pela diferença entre os limites dos intervalos.

O valor *maior_diferença* é dado pela diferença entre o maior e o menor valor do domínio deste atributo. Estes valores devem ser fornecidos pelo usuário no momento da inserção da restrição no sistema. Caso não sejam conhecidos, o sistema identifica o maior e o menor valor previamente fornecidos para esta restrição e calcula a diferença.

3.6 Algoritmo de retenção de casos e estrutura de organização

As principais funcionalidades do algoritmo de retenção de casos proposto para o WOODSS são: avaliar se o novo caso deve ser armazenado, requisitar ao usuário os descritores para indexar este caso e, finalmente, armazená-lo na memória de casos.

Um novo caso só deve ser retido se for suficientemente "diferente", em sua semântica, de outros casos já armazenados. Se o novo caso possui objetivos diferentes dos casos já existentes, faz sentido retê-lo. Porém, qual a utilidade de armazenar vários casos referentes a um mesmo objetivo?

A resposta para a última questão baseia-se no fato de que, ainda que o objetivo seja o mesmo, as restrições adotadas podem diferir. Isto, por sua vez, implica variações nos parâmetros utilizados na implementação dos modelos. Este tipo de situação é muito comum em aplicações ambientais, em que restrições espaço-temporais podem exigir implementações bastante diferentes de um modelo global genérico. É preciso conhecimento especialista sobre o problema atacado para definir valores corretos para os parâmetros, adequando um caso a novas restrições. Desta forma, mostrar ao usuário um modelo genérico de um problema indica o rumo a tomar, mas pode não ser suficiente.

Por exemplo, considere novamente o problema de recomendação de quantidades de fertilizantes, baseada em tabelas fornecidas pelas instituições de pesquisa agrícola. Os modelos, como introduzido em um dos exemplos da seção 3.4, compreendem reclassificar os mapas de distribuição de teores de nutrientes de acordo com estas tabelas. Como as tabelas são específicas para cada cultura e variam de região para região, retornar um modelo genérico ao usuário requer que ele conheça os valores corretos de recomendação para adequá-lo à situação em questão.

Pela dificuldade da inferência automática de relevância de um novo caso, esta dissertação propõe que o WOODSS permita armazenar qualquer caso que o usuário julgue importante, desde que os casos não tenham conjuntos de descritores de indexação estritamente iguais. O algoritmo de retenção proposto é o algoritmo 3.3.

3.7 Algoritmo de recuperação de casos

O algoritmo de recuperação de casos proposto para o WOODSS retorna ao usuário o caso mais próximo do caso de entrada (a partir dos descritores fornecidos pelo usuário) e um conjunto de casos complementares. Os casos complementares têm por objetivo apresentar ao usuário algum caso que contenha uma restrição que falta no caso recuperado, indicando uma forma de tratá-la. Os casos complementares são os casos mais próximos do caso de entrada e que possuem as restrições faltantes no caso retornado.

O usuário pode fornecer os descritores de consulta de duas maneiras:

- Em um passo. O usuário fornece de uma só vez os descritores do tipo 1 e do tipo 2.
- Em dois passos. Primeiramente o usuário fornece os descritores do tipo 1. O sistema retorna a lista dos objetivos associados a algum destes descritores. O usuário então seleciona nesta lista os objetivos desejados e fornece os descritores do tipo 2,

Algoritmo 3.3 Algoritmo de retenção de casos proposto para o WOODSS.

Entrada: novo caso
Saída: confirmação de armazenamento do caso (booleano)

1. Documentar o novo caso
   1.1 Usuário fornece a descrição textual do caso
   1.2 Usuário determina descritores para indexar o caso
   - Palavras-chave dos objetivos do caso (descritores do tipo 1)
   - Restrições associadas a estes objetivos (descritores do tipo 2)

2. Decidir se armazena ou não o novo caso
   2.1 Sistema pesquisa memória de casos utilizando descritores obtidos no passo 1.2 e retorna ao usuário casos semelhantes ao novo caso
   2.2 Usuário decide se armazena o caso, analisando a similaridade entre o novo caso e os recuperados

3. Se decidir armazenar, reter o novo caso
   3.1 Usuário valida descritores para indexar o novo caso
      3.1.1 Ajusta os descritores fornecidos inicialmente em 1.2
      3.1.2 Acrescenta novos descritores, baseando-se nos descritores dos casos retornados, ressaltando as diferenças entre o novo caso e os demais
   3.2 Sistema retém o caso
      3.2.1 Armazena o workflow e a descrição do caso
      3.2.2 Armazena os descritores de indexação do caso

associando-os aos respectivos objetivos. Nesta forma de interação, o sistema auxilia o usuário a determinar precisamente o contexto de cada restrição (descritor do tipo 2), diminuindo possíveis ambiguidades na análise de similaridade.

O algoritmo de recuperação de casos proposto é o algoritmo 3.4. Este algoritmo não aproveita o fato de que aplicações que usam dados georeferenciados podem utilizar metadados associados aos mapas. Estes metadados são gerenciados pelo SIG utilizado e contêm tipicamente informações sobre a região geográfica e o sistema de coordenadas. Uma versão do algoritmo que considera este fato é o algoritmo 3.5.

3.8 Resumo

Este capítulo apresentou os algoritmos que devem ser incorporados ao WOODSS para permitir o uso de CBR, na retenção e recuperação de modelos. Os descritores escolhidos para um caso são seus objetivos, representados por listas de palavras-chave, e restrições formadas por conjunções de triplas do tipo <atributo, comparando, valor>. O cálculo de similaridade proposto é baseado na noção de vizinho mais próximo, comparando valores textuais através de caminhos em árvores de termos fornecidos pelo usuário e valores numéricos através de diferença algébrica.

O próximo capítulo mostra a implementação realizada, incluindo as mudanças feitas no banco de dados para armazenamento dos descritores e a reestruturação dos módulos de Atualizações e Consultas do WOODSS.

Algoritmo 3.4 Algoritmo de recuperação de casos proposto para o WOODSS.

Entrada: descritores de caso
Saída: conjunto de casos cujos descritores sejam similares aos descritores de entrada

1. Recuperar casos relevantes usando critérios de similaridade a partir de descritores dos tipos 1 e 2, refinando iterativamente a busca
   1.1 Usuário fornece os descritores
   - Em um passo
     a) Usuário fornece conjunto de palavras-chave e de restrições
   - Em dois passos
     a) Usuário fornece conjunto de palavras-chave e sistema retorna a lista dos objetivos associados
     b) Usuário fornece conjunto de restrições
        b.1) Usuário seleciona os objetivos desejados na lista retornada no item (a)
        b.2) Sistema retorna nomes dos atributos das restrições (triplas) associadas aos objetivos selecionados
        b.3) Usuário determina valores e comparandos para as respectivas triplas das restrições

   1.2 Sistema retorna conjunto de casos com maior valor de similaridade
   1.3 Se algum dos casos satisfaz o usuário, tome-o e vá para o passo 2
   Senão, retorne ao início do passo 1 para o usuário refinar os descritores

2. Recuperar casos complementares
   2.1 Sistema identifica as restrições fornecidas pelo usuário que o caso retornado não satisfaz
   2.2 Sistema retorna os casos com maior valor de similaridade que contenham estas restrições

Algoritmo 3.5 Algoritmo de recuperação de casos, sensível ao georeferenciamento.

Entrada: descritores de caso
Saída: conjunto de casos cujos descritores sejam similares aos descritores de entrada

1. Recuperar casos relevantes usando critérios de similaridade a partir de descritores e indicadores de localização geográfica, refinando iterativamente a busca
   1.1 Usuário fornece descritores e indicadores de localização
      1.1.1 Usuário fornece os descritores dos tipos 1 e 2
      1.2.1 Usuário fornece indicadores de localização (toponímia ou coordenadas)
   1.2 Sistema retorna conjunto de casos com maior valor de similaridade, tal que
      - O grau de similaridade entre descritores é maximizado
      - Existe interseção geográfica entre a localização fornecida pelo usuário e os arquivos de dados usados como entrada nos casos selecionados. Esta interseção é calculada usando funções do SIG utilizado

O restante do algoritmo é idêntico ao 3.4.

Capítulo 4

Aspectos de implementação

Este capítulo apresenta os principais aspectos de implementação para reestruturar o WOODSS usando CBR no gerenciamento de modelos. Os módulos do WOODSS afetados com esta reestruturação foram: o módulo de Consultas, o módulo de Atualizações e a Interface com o usuário.

A figura 4.1 é uma versão da figura 2.3, que apresenta a nova arquitetura do WOODSS, destacando, com sombreado mais escuro, os módulos afetados. O módulo Monitor passou a ser denominado *Interface SIG* e o módulo Gestor de *Workflows* passou a ser chamado *Interface BD*, mantendo as mesmas funcionalidades. Nesta figura, o encapsulamento esquemático dos módulos de Consulta, de Atualizações e a Interface com o usuário tem por objetivo ilustrar que estes módulos formam o núcleo do sistema e os módulos Interface SIG e Interface BD são os responsáveis por integrar estes módulos aos componentes externos ao sistema (SIG e Memória de Casos, respectivamente) embora todos componham a nova versão do WOODSS.

A especificação dos módulos de Consulta e Atualizações foi estendida, passando a incorporar funções para recuperação e retenção de casos. As modificações realizadas na Interface com o usuário visaram facilitar as tarefas de edição e documentação dos casos, além de novas janelas para comunicação com os novos módulos de Consulta e Atualizações. Esta nova versão, da mesma forma que a versão original, interage com o SIG Idrisi e com o sistema de gerenciamento de banco de dados *Visual FoxPro* e foi implementada utilizando a linguagem Java^{TM}.

Este capítulo está organizado da seguinte forma. A seção 4.1 mostra a reengenharia realizada no WOODSS para torná-lo mais aberto, facilitando expansões futuras. A seção 4.2 apresenta o esquema do banco de dados utilizado nesta implementação. As seções 4.3 e 4.4 descrevem como os módulos de Atualizações e de Consultas do WOODSS foram recodificados para acoplar os mecanismos de CBR.

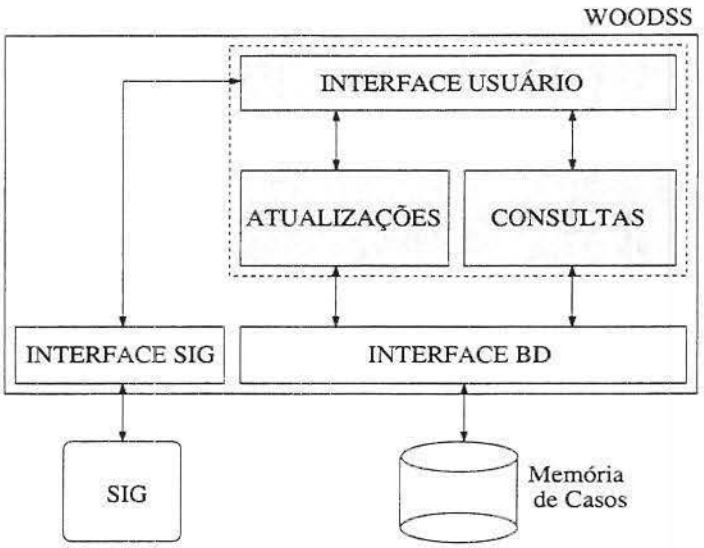

Figura 4.1: Nova arquitetura do WOODSS, incorporando CBR.

4.1 Reengenharia do WOODSS

A incorporação de CBR ao WOODSS exigiu sua total reengenharia e recodificação, para permitir expansões futuras. Esta nova implementação do WOODSS continua acoplada ao SIG Idrisi, apesar da modularização alcançada torná-lo praticamente independente do SIG com o qual interage. A figura 4.2 mostra o diagrama de classes gerado. Nesta figura, quando a cardinalidade dos relacionamentos é omitida, entenda-se '1', e o símbolo '*' indica a cardinalidade 'n'. As classes estão particionadas em quatro grandes pacotes: CORE, INTER, GIS e DATABASE. Cada um destes pacotes é responsável por um grupo de funcionalidades, descritos a seguir.

Pacote CORE. Este pacote pode ser entendido como a própria definição de um *workflow*. A classe `Workflow` encapsula todos os componentes e funcionalidades do *workflow*. Uma instância desta classe é formada por um conjunto de atividades e um conjunto de dependências. Note que, como *workflows* têm uma definição recursiva (uma atividade pode ser um *subworkflow*), a classe `Workflow` é derivada de `Activity`. Cada atividade ou dependência possui um registro de metadados particular (classes `ActivityMetadata` e `DependencyMetadata`), onde são armazenados desde parâmetros até comentários em alto nível sobre estes componentes. A classe `Metadata` contém metadados sobre o *workflow* global, tal como descrito na seção 3.3. A classe `User` representa os usuários do sistema, servindo tanto para autenticação e permissão de acesso de usuários quanto como parte dos metadados do *workflow*, indicando o especialista que o desenvolveu. Todas as funções de edição de *workflows* estão encapsuladas neste pacote, sendo invocadas pelos outros

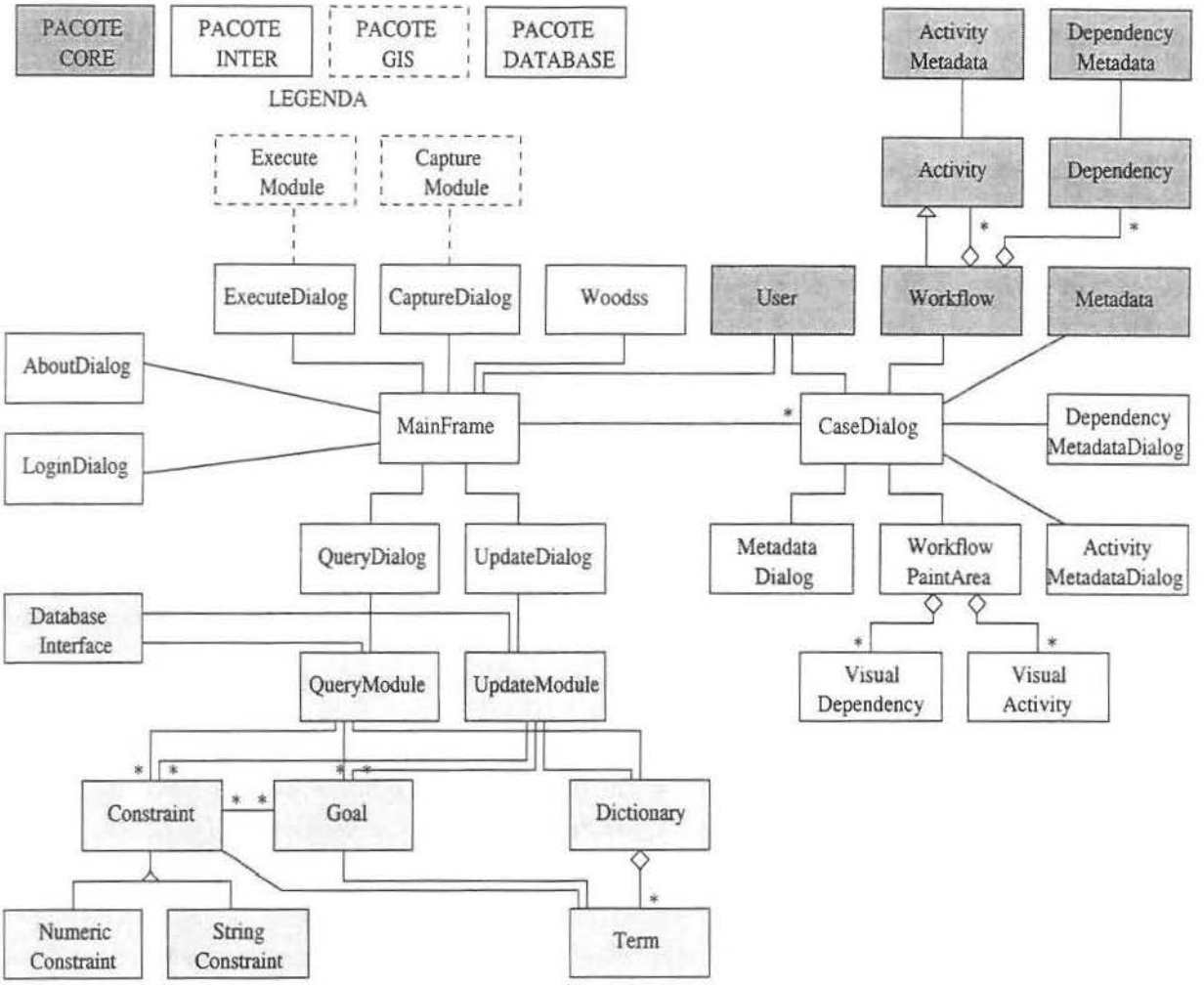

Figura 4.2: Diagrama de classes da nova versão do WOODSS.

pacotes que complementam a gerência de *workflows*.

Pacote INTER. Este pacote encerra as classes da interface com o usuário do sistema, responsáveis pela apresentação de *workflows* e por receber e responder a eventos do usuário. Esta implementação buscou descentralizar a gerência de janelas, de acordo com as funcionalidades do sistema. A classe `MainFrame` é o ponto de ligação entre os diferentes módulos do sistema. Todas as tarefas de apresentação e edição de *workflows* e metadados são gerenciadas pela classe `CaseDialog`, que possui um conjunto de subjanelas associadas. Da mesma forma as classes `QueryDialog`, `UpdateDialog`, `ExecuteDialog` e `CaptureDialog` gerenciam, respectivamente, as solicitações das funcionalidades de consultas, atualizações, execução e captura, interagindo com os pacotes `DATABASE` e `GIS`. A classe `Woodss` é a classe de inicialização do sistema, que realiza alguns testes iniciais e

mostra a janela principal do WOODSS. Finalmente, a classe `LoginDialog` captura dados de *login* dos usuários e a classe `AboutDialog` mostra informações sobre a versão do sistema.

Pacote DATABASE. Este é o pacote responsável pela comunicação do sistema com o SGBD que contém a memória de casos. Os módulos de Atualizações, de Consultas e InterfaceBD estão encapsulados neste pacote. As classes `UpdateModule` e `QueryModule` provêm as funções de gerenciamento da memória de casos, encerrando os mecanismos de recuperação e retenção de casos, propostos nesta dissertação. A classe `DatabaseInterface` é a ponte entre este pacote e o SGBD, comunicando-se via JDBC (*Java DataBase Connectivity*), uma API Java padrão que fornece primitivas para execução de comandos SQL (*Structured Query Language*) em um SGBD. As classes `Goal` e `Constraint` contêm os descritores de casos descritos no capítulo anterior e a classe `Dictionary` carrega o dicionário de termos utilizado, fornecendo funções de gerência de termos e comparação de valores.

Pacote GIS. O pacote GIS realiza a interação do WOODSS com o SIG acoplado, via tradução das operações do SIG em *workflows* (captura) e vice-versa (execução). Este pacote foi apenas parcialmente implementado. Para a tarefa de captura, a classe `CaptureModule` engloba as funções necessárias para ler um arquivo de *log*, e convertê-lo para o formato de um *workflow* científico, identificando suas atividades (com seus parâmetros), dependências e metadados. Para a tarefa de execução, a classe `ExecuteModule` gera um arquivo executável no SIG que reflete a estrutura do *workflow* que se deseja executar. A implementação completa deste pacote deve comunicar-se com o pacote DATABASE, armazenando no banco de dados as assinaturas de operações do SIG e outras informações. Desta forma, torna-se possível criar rotinas genéricas, flexíveis o suficiente para acomodarem as variações de operações e parâmetros em diferentes SIG. Isto permite uma maior independência do SIG utilizado, uma vez que cada SIG possui uma linguagem proprietária. Esta versão do WOODSS ainda interage apenas com o SIG Idrisi.

4.2 Esquema do banco de dados

Esta seção apresenta as relações que compõem o banco de dados da nova implementação do WOODSS. Primeiramente, apresenta-se o esquema do banco de dados da implementação original do WOODSS, e na seqüência, as relações acrescentadas na nova implementação.

4.2.1 Relações da implementação original do WOODSS

O esquema original do banco de dados do WOODSS continha as seguintes relações para armazenamento de *workflows*:

Workflows. Esta relação mantém um registro dos *workflows* armazenados. `id` é um identificador único para cada *workflow* e `title` é o título do *workflow*, fornecido pelo usuário.

```
Workflows(
     id: NUMERIC(14)
     title: CHAR(40)  )
```

Atividades. Contém os dados sobre as atividades do *workflow*. `id` é o identificador da atividade em um *workflow*, e `name` o título desta atividade. O atributo `origin` indica se a atividade foi obtida do Idrisi (I) ou fornecida manualmente pelo usuário (M), e `module` identifica a operação do Idrisi correspondente à atividade. Os atributos `input` e `output` correspondem, respectivamente, às dependências de entrada e de saída da atividade, representadas por listas de identificadores, separados por um caracter especial ('#'). Finalmente, `others` contém uma lista de valores, contendo os demais parâmetros da atividade, separados por um caracter especial.

```
Atividades (
     id: INT
     name: CHAR(40)
     origin: CHAR(1)
     module: CHAR(10)
     input: CHAR(10)
     output: CHAR(10)
     others: CHAR(60)  )
```

Arquivos. Armazena informações sobre os arquivos (mapas) usados no processo decisório documentado pelo *workflow*. `id` é o identificador, `name` e `extension` o nome e a extensão do arquivo, respectivamente, e `title` é o título do mapa, retirado do arquivo de documentação deste mapa (no Idrisi, todo arquivo de mapa possui um arquivo de descrição associado).

```
Arquivos (
     id: INT
```

```
    name: CHAR(40)
    extension: CHAR(3)
    title: CHAR(40)  )
```

Dependências. Contém as dependências entre as atividades do *workflow*. Os atributos `id1_Atividades` e `id2_Atividades` são chaves estrangeiras da relação `Atividades`, indicando as atividades envolvidas nesta dependência. `id_Arquivos` é uma chave estrangeira da relação `Arquivos`, que relaciona esta dependência (de dados) a um arquivo do Idrisi, e `type` identifica o tipo da dependência: (D) dados, (T) temporal ou (E) execução.

```
Dependências (
    id1_Atividades: INT
    id2_Atividades: INT
    type: CHAR(1)
    id_Arquivos: INT  )
```

Além das relações descritas acima, na versão original do WOODSS cada *workflow* possuía um arquivo textual associado, onde eram armazenados os seus metadados, em formato livre.

4.2.2 Relações acrescentadas na nova versão do WOODSS

O esquema do banco de dados da implementação original do WOODSS foi aumentando de um conjunto de relações, para comportar os mecanismos de recuperação e retenção de casos utilizando CBR. Estas relações armazenam basicamente: (a) os casos, (b) os seus descritores e (c) o dicionário de termos, utilizado para comparação de valores simbólicos dos descritores.

Relações que armazenam os casos

Case. Esta relação armazena os casos existentes no WOODSS. É formada por quatro identificadores: `id`, que é o identificador do caso; `Workflow_id`, que é o identificador do *workflow* associado a este caso (chave estrangeira da relação `Workflows`); `Metadata_id`, que associa o caso aos seus metadados (chave estrangeira da relação `Metadata`); e `User_id`, que identifica o usuário que inseriu o caso no sistema (chave estrangeira da relação `User`).

```
Case (
    id: INT
    Workflow_id: INT
    Metadata_id: INT
    User_id: INT  )
```

Metadata. Contém os metadados associados aos casos. Esta relação equivale ao arquivo textual de metadados, utilizado na versão original do WOODSS. Os campos são textuais, preenchidos em formato livre, como citado no capítulo anterior. `id` é o identificador do registro de metadados. `goal` e `model` armazenam, respectivamente, descrições dos objetivos do caso e dos modelos ambientais implementados. `area` e `inputdata`, contêm informações sobre a área de estudo e sobre os dados de entrada utilizados. `outcome` é uma descrição dos resultados das decisões tomadas sobre os mapas gerados e o atributo `comments` contém comentários complementares sobre o caso.

```
Metadata (
     id: INT
     goal: TEXT
     model: TEXT
     area: TEXT
     inputdata: TEXT
     outcome: TEXT
     comments: TEXT  )
```

User. Armazena os usuários cadastrados no sistema. Esta relação é utilizada tanto para o gerenciamento de permissões de acesso à memória de casos, quanto para identificar o especialista responsável pelo desenvolvimento de um caso. `id` e `username` são os identificadores e os nomes de usuário (únicos) dos usuários reconhecidos pelo sistema. O domínio do atributo `groupid` distingue três categorias de usuários: 2–usuários comuns, que podem apenas consultar casos, 1–usuários especialistas, que podem também inserir casos, e 0–administradores, que gerenciam a memória de casos e as contas de usuários do sistema. Os demais atributos são informações pessoais sobre o usuário (nome, organização e endereço eletrônico).

```
User (
     id: INT
     username: CHAR(15)
     groupid: INT
     name: CHAR(60)
     organization: CHAR(60)
     email: CHAR(60)  )
```

Relações do dicionário de termos

Como o dicionário de termos é utilizado nos descritores dos casos, suas relações serão apresentadas primeiro. O dicionário é organizado através de relacionamentos hierárquicos

e de equivalência, conforme comentado anteriormente. As relações que o armazenam são as seguintes:

Term. Descreve os termos do dicionário. `id` é o identificador do termo, `term` é o termo propriamente dito (pode ser formado por um conjunto de palavras, por exemplo, "soja perene") e `description` contém uma descrição do termo (quando necessário), para evitar ambiguidades.

```
Term (
     id: INT
     term: CHAR(60)
     description: TEXT  )
```

EquivalenceRelationship. Contém os termos equivalentes no dicionário. Para facilitar a análise de similaridade, nesta implementação diferenciam-se termos *base* e termos *equivalentes*. Termos base são utilizados na indexação dos casos e os termos equivalentes são utilizados apenas para mapear um termo equivalente, fornecido pelo usuário durante a documentação de um caso ou consulta à memória de casos, para um termo base. Convencionou-se, nesta implementação, que os termos base necessariamente devem fazer parte da hierarquia de conceitos (armazenada na relação `HierarchicalRelationship`) e os termos equivalentes não podem estar nesta hierarquia. O atributo `Term_id_base` é o identificador do termo base e `Term_id_equivalent` é o termo equivalente ao termo base, ambos chaves estrangeiras da relação `Term`.

```
EquivalenceRelationship (
     Term_id_base: INT
     Term_id_equivalent: INT  )
```

HierarchicalRelationship. Contextualiza os termos base do dicionário em uma hierarquia de conceitos. `ancestor` é o identificador de um termo em uma posição mais alta na hierarquia e `descendant` é o identificador de um descendente direto deste termo.

```
HierarchicalRelationship (
     Term_id_ancestor: INT
     Term_id_descendant: INT  )
```

Relações que armazenam os descritores dos casos

O conjunto de relações que armazenam os descritores dos casos (tipos 1 e 2) é o seguinte:

Goal. Contém uma descrição suscinta de cada objetivo individual que algum caso armazenado possui. Serve para identificar os tipos de problemas abordados nos casos. `id` é o identificador do objetivo e `description` é a sua descrição.

```
Goal (
     id: INT
     description: TEXT  )
```

CaseGoal. Relaciona casos e objetivos, onde um caso pode ter n objetivos e um objetivo pode estar relacionado a m casos. `Case_id` é o identificador do caso (chave estrangeira da relação `Case`) e `Goal_id` é o identificador do objetivo (chave estrangeira da relação `Goal`).

```
CaseGoal (
     Case_id: INT
     Goal_id: INT  )
```

GoalTerm. Armazena os descritores dos objetivos dos casos (descritores do tipo 1). `Goal_id` é o identificador do objetivo e `Term_id` é um termo base do dicionário de termos.

```
GoalTerm (
     Goal_id: INT
     Term_id: INT  )
```

Constraints. Contém uma descrição das restrições de acordo com o seu atributo. `id` contém os identificadores das restrições e o campo `attribute` armazena os identificadores dos seus atributos (termo base do dicionário).

```
Constraints (
     id: INT
     attribute: INT  )
```

GoalConstraints. Relaciona objetivos e restrições (relacionamento $n : m$). Contextualiza uma determinada restrição de acordo com um objetivo particular. Os campos desta relação indicam que a restrição `Constraints_id` ao objetivo `Goal_id` é do tipo `type` (s–*string* ou n–numérica), possui um peso de relevância `weight`, e tem o significado descrito em `description`. Note que o tipo, o peso e a descrição das restrições são colocados nesta relação porque restrições com o mesmo atributo podem ter significados diferentes de acordo com o contexto.

```
GoalConstraints (
     Goal_id: INT
     Constraints_id: INT
     type: CHAR(1)
     weight: INT
     description: TEXT  )
```

CaseGoalConstraints_String. Contém o valor da restrição do tipo *string*, identificada como `Constraints_id`, com relação ao objetivo `Goal_id` e ao caso `Case_id`. O valor (*string*) é um termo base do dicionário.

```
CaseGoalConstraints_String (
     Case_id: INT
     Goal_id: INT
     Constraints_id: INT
     value: INT  )
```

CaseGoalConstraints_Numeric Tem a mesma finalidade da relação anterior, mas para valores numéricos (inteiros ou reais). Esta relação possui ainda o atributo *comparison_operator*, cujo domínio, nesta implementação, é '<', '=' e '>'.

```
CaseGoalConstraints_Numeric (
     Case_id: INT
     Goal_id: INT
     Constraints_id: INT
     comparison_operator: CHAR(1)
     value: FLOAT  )
```

As próximas seções apresentam detalhes de implementação dos módulos de Atualizações e Consultas, que são o foco principal desta dissertação.

4.3 Módulo de Atualizações

O módulo de Atualizações foi completamente recodificado, para incorporar a retenção de casos. Foi definida a seguinte hierarquia de usuários para garantir o controle sobre os casos armazenados:

- Administrador: é o usuário que gerencia o sistema e manipula as contas de usuário. É o único que tem permissão para remover casos armazenados, sendo responsável por realizar uma manutenção periódica, eliminando casos pouco úteis e redundantes.

- Usuário especialista: é o usuário que pode acrescentar novos casos. Tem permissão para qualquer operação no sistema, exceto remover casos e manipular contas de usuário. Apenas o usuário especialista que inseriu um caso no sistema pode atualizá-lo posteriormente.

- Usuário comum: tem permissão somente para consultas. Um usuário comum pode recuperar casos, adaptá-los e executá-los, mas não pode retê-los no banco de casos.

Os componentes da interface que dizem respeito à retenção de modelos são habilitados e desabilitados de acordo com o nível do usuário. As informações sobre as contas de usuários são armazenadas na relação `User`, descrita na seção 4.2.

Como visto na seção 3.6, o processo de retenção de casos proposto para o WOODSS compreende três passos: (1) documentação do novo caso, (2) decisão de armazená-lo ou não, e (3) sua retenção. A seguir são apresentados alguns detalhes da implementação deste processo.

4.3.1 Documentação do novo caso

Quando o usuário solicita o armazenamento de um caso, a classe `MainFrame` passa para o módulo de Atualizações os componentes deste caso (uma instância da classe `Workflow`, com suas atividades e dependências e uma da classe `Metadata`).

Inicialmente, os metadados deste caso são apresentados ao usuário, para que ele atualize-os se desejar. Em seguida, o sistema requisita que o usuário atribua descritores para a indexação do caso. Esta etapa é dividida em: definir os objetivos do caso, e definir as restrições inerentes a estes objetivos.

Cada objetivo armazenado pelo sistema já possui um conjunto de palavras-chave associado para indexação (descritores do tipo 1). O módulo de Atualizações oferece ao usuário as seguintes operações:

- Remover objetivos atribuídos ao caso;

- Atribuir outros objetivos armazenados ao caso;

- Acrescentar novas palavras-chave a um objetivo; e

- Criar um novo objetivo e atribuí-lo ao caso. Para criar um novo objetivo, o usuário precisa fornecer uma breve descrição e associar-lhe um conjunto de descritores do tipo 1.

Definidos os objetivos do caso, o usuário deve determinar as restrições do caso inerentes a estes objetivos (triplas <atributo, comparando, valor>). O sistema carrega as

restrições previamente associadas a cada objetivo (quando existirem). O usuário pode, então, alterar os valores destas restrições ou associar novas restrições ao objetivo. Para associar uma restrição a um objetivo o usuário deve indicar o seu *atributo*, o seu *tipo* (*string* ou numérica), o seu *peso*, e uma *descrição* do seu significado com relação a este objetivo em específico. Note que apenas as restrições que tiverem valor definido serão armazenadas como descritores para o caso.

4.3.2 Decisão de armazenar ou não o novo caso

Para evitar redundâncias na memória de casos, o módulo de Atualizações solicita ao módulo de Consultas uma busca utilizando os descritores fornecidos no passo anterior. O conjunto recuperado de casos similares visa apoiar a decisão do usuário por armazenar ou não o novo caso, avaliando se o novo caso é "suficientemente diferente" dos outros casos armazenados. A decisão de armazenar o novo caso é bastante subjetiva, cabendo ao usuário tomar a decisão. Desta forma, o administrador do sistema deve realizar manutenções periódicas na memória de casos.

4.3.3 Retenção do novo caso.

Na etapa de retenção, o usuário valida os descritores fornecidos inicialmente, complementando com novos descritores se necessário. Uma vez validados os descritores, o sistema materializa as informações do caso nas relações do banco de dados. Se o novo caso é fruto da reutilização de um caso recuperado da memória de casos, há duas possibilidades:

- Se o usuário que requisitou a inserção do caso não é o mesmo que o inseriu inicialmente, o sistema atribui uma nova identificação ao caso e o retém com esta identificação, armazenando o *workflow* e os descritores nas relações correspondentes.
- Se o usuário é o mesmo que inseriu inicialmente o caso, ele tem duas opções: sobrescrever o caso antigo, utilizando a mesma identificação (atualização) ou inseri-lo normalmente como um novo caso, obtendo uma nova identificação.

Se o sistema encontrar algum termo novo, que ainda não consta no seu dicionário, requisita ao usuário que o acrescente ao dicionário. Estes termos são os descritores do tipo 1, os nomes de atributos e os valores textuais de descritores do tipo 2. Para isto, o usuário realiza sucessivas buscas no dicionário de termos por palavras-chave associadas ao novo termo, para identificar em que hierarquia ele deve ser inserido. Se não houver nenhuma hierarquia que acomode o novo termo, o usuário deve fornecer uma nova hierarquia que o contextualize. O usuário pode também refinar as hierarquias já existentes, acrescentando novos termos.

4.4 Módulo de Consultas

O módulo de Consultas original do WOODSS só permitia consultas a partir do *nome* dado aos *workflows* armazenados. A nova versão deste módulo calcula a similaridade entre casos considerando uma série de características dos casos, refletidas nos seus descritores. Como visto anteriormente (vide seção 3.7), o algoritmo de recuperação de casos proposto é composto de duas partes: (1) recuperar os casos mais similares com o caso de entrada e (2) recuperar casos complementares ao conjunto de casos inicialmente recuperado.

4.4.1 Recuperação de casos similares

Esta etapa do processo de recuperação de casos proposto, consiste, basicamente, na iteração dos seguintes passos, até que o usuário identifique casos que o satisfaçam:

1. O usuário fornece descritores (tipos 1 e 2), em um ou dois passos; e
2. O sistema retorna o conjunto de casos mais similares.

O valor final de similaridade entre o caso de entrada e os casos armazenados é obtido pela média ponderada das similaridades dos descritores dos tipos 1 e 2, priorizando o valor dos descritores do tipo 1. Definiu-se, nesta implementação, os pesos $P_1 = 2$ e $P_2 = 1$ para as similaridades dos descritores dos tipos 1 e 2, respectivamente. Desta forma, a similaridade final entre o caso de entrada (i) e um caso armazenado (c) é calculada como:

$$sim(i,c) = \frac{2 \times simObj(i,c) + simRestr(i,c)}{3}$$

A análise de similaridade foi implementada segundo os algoritmos do capítulo 3, ressaltando-se os seguintes aspectos de implementação:

- As hierarquias são construídas dinamicamente a partir do dicionário de termos, que é armazenado em um conjunto de tabelas no banco de dados;
- Por razões de desempenho, estipulou-se que percursos nestas hierarquias seriam limitados a dois níveis acima e dois abaixo de um dado termo;
- A análise de similaridade de descritores do tipo 1 é feita através de casamentos parciais de *strings*, utilizando as hierarquias de termos do dicionário do sistema; e

- A análise de similaridade de descritores do tipo 2 segue diretamente a idéia introduzida na seção 2.2.5: (1) Encontrar correspondências, (2) Calcular o grau de similaridade entre descritores (neste caso, o campo *valor* dos termos das restrições), e (3) ponderar a importância dos descritores.

  1. Encontrar correspondências. Para cada tripla das restrições de entrada, o sistema procura na tabela RestriçãoObjetivo apenas correspondências exatas entre os *atributos* e os *tipos* dos termos de restrições aos objetivos do caso de entrada e aos objetivos armazenados. Não são consideradas correspondências de abstrações ou especializações, utilizando as hierarquias do dicionário, pelo grande custo deste processamento e tampouco se considera a semântica das restrições. Por exemplo, as triplas <textura, '=', argilosa> e <textura, '>', 50%> são equivalentes, mas o sistema não possui meios para determinar tal equivalência.
  2. Calcular o grau de similaridade entre descritores. Este cálculo considera dois tipos de valores: *string* e numérico. Utiliza como método de comparação dos respectivos tipos de valores, o caminho entre os atributos do tipo *string* nas hierarquias do dicionário e o valor absoluto da diferença entre os atributos numéricos.
  3. Ponderar a importância dos descritores. Os pesos para cada termo dos descritores do tipo 2 são recuperados a partir da tabela `RestriçãoObjetivo`, particularmente para cada objetivo. Estes pesos são definidos pelos usuários no momento que uma nova restrição é inserida na memória de casos.

4.4.2 Recuperação de casos complementares

A recuperação de casos complementares não foi implementada por tratar-se de um procedimento idêntico ao de recuperação de casos, modificando os pesos das restrições não satisfeitas pelo caso retornado. Este processo pode ser executado a partir de uma sequência de interações do usuário, que realiza novas consultas, variando os descritores fornecidos para encontrar os casos desejados.

4.5 Resumo

Este capítulo apresentou as principais considerações sobre a implementação realizada para acoplar CBR ao WOODSS. O WOODSS foi completamente reestruturado para comportar expansões subseqüentes. Os módulos de Atualizações e Consultas foram estendidos para incorporar os algoritmos propostos no capítulo 3. O banco de dados original foi aumentado

de um conjunto de relações para armazenar os descritores para indexação dos casos. Esta implementação foi realizada utilizando a linguagem Java^{TM} e o SGBD *Visual Fox Pro*, e interage com o SIG Idrisi.

O próximo capítulo mostra, através de uma aplicação real, o uso do WOODSS recodificado inserindo mecanismos baseados em CBR, para a resolução de problemas decisórios na área de planejamento agrícola.

Capítulo 5

Exemplo de sessão

O objetivo deste capítulo é ilustrar a interação do usuário com a nova versão do WOODSS, apresentando um exemplo de utilização da nova versão do WOODSS, em um problema real de planejamento agro-ambiental.

O problema escolhido foi o de recomendação de fertilizantes e corretivos em taxa variável, introduzido na seção 3.2.1. A seção 5.1 define o problema e descreve o processo de solução, utilizando o SIG Idrisi. A seção 5.2 mostra como o WOODSS pode ser utilizado para auxiliar o processo decisório e ilustra a interação do usuário nos processos de retenção (5.2.1) e recuperação de casos (5.2.2).

5.1 Recomendação de fertilizantes e corretivos utilizando um SIG

Para ilustrar o uso de um SIG e da nova versão do WOODSS, suponha que um especialista deseja construir os mapas de quantidades de calcário e fertilizantes, para a cultura da soja em uma propriedade no estado de São Paulo. Este problema já foi introduzido na seção 3.2.1.

O exemplo apresentado nesta seção considera o uso do SIG Idrisi. Em outros SIG existirão algumas diferenças, pois cada SIG implementa uma linguagem proprietária. Os módulos do Idrisi utilizados são: *overlay*, *scalar* e *reclass*. Os módulos *overlay* e *scalar* fornecem as operações matemáticas básicas sobre mapas. A diferença entre estes módulos é que *overlay* toma dois mapas de entrada e tem como resultado um terceiro, e *scalar* tem por entrada um mapa e um valor escalar, resultando em outro mapa. O módulo *reclass* reclassifica um mapa de acordo com novos critérios fornecidos pelo usuário.

Para simplificar o entendimento, neste exemplo não são considerados vários fatores, como aplicações anteriores de insumos, estágio na rotação de culturas, diferentes fórmulas

de fertilizantes, entre outros. O processo de solução é dividido em duas partes: (1) geração do mapa da necessidade de calagem; (2) construção dos mapas das quantidades de fertilizantes.

1. Mapa da necessidade de calagem

O cálculo da necessidade de calagem em São Paulo é usualmente realizado utilizando o método de *saturação de bases trocáveis do solo* [RCQF96], dado por:

$$NC(t/ha) = \frac{CTC \times (V2\% - V\%)}{10 \times PRNT}$$

Onde:

- CTC: capacidade de troca de cátions do solo a pH=7.0, calculada como:
  $CTC = S + (H^{+} + Al^{+++})\ mmol_c/dm^3$,
  onde S é a soma das bases do solo, dado por: $S = Ca^{++} + Mg^{++} + K^{+} mmol_c/dm^3$;
- $V\%$: valor da saturação de bases trocáveis do solo, em porcentagem, antes da correção ($V\% = \frac{100 \times S}{CTC}$);
- $V2\%$: porcentagem desejada de saturação de bases; e
- $PRNT$: Poder Relativo de Neutralização Total, que é uma característica do calcário a ser utilizado.

Nesta equação, existem dois dados de entrada escalares e cinco sob a forma de mapas. Os valores escalares são o PRNT, dado pelo fornecedor do calcário, e $V2\%$, que varia de acordo com a cultura e com a região. Considere, neste exemplo o valor do $PRNT = 70\%$ e da saturação desejada $V2\% = 60\%$, que é o valor recomendado pelo IAC para a soja em São Paulo. Os mapas de entrada fornecidos são os de teores de cálcio (Ca), magnésio (Mg), potássio (K), hidrogênio trocável (H) e alumínio trocável (Al). Uma maneira de implementar esta equação no SIG Idrisi utiliza o algoritmo a seguir.

1. Calcular $S = Ca^{++} + Mg^{++} + K^{+}$, sobrepondo os mapas de teores de cálcio (Ca) e magnésio (Mg), e em seguida o mapa resultante e o mapa de teor de potássio (K), utilizando a opção de soma do módulo *overlay* do Idrisi;
2. Calcular $CTC = S + (H^{+} + Al^{+++})$, sobrepondo sucessivamente os mapas de teores de hidrogênio (H) e alumínio (Al), e o mapa resultante com S, obtido no passo anterior, novamente usando a opção soma da função *overlay*;

3. Calcular $V\% = \frac{100 \times S}{CTC}$, usando a opção de multiplicação por valor constante do módulo *scalar* $(100 \times S)$ e sobrepondo o mapa resultante com o mapa CTC, através da opção de divisão da função *overlay*;

4. Fazer $CTC(60\% - V\%)$, utilizando a opção subtração por valor constante do módulo *scalar*, e em seguida sobrepondo o mapa resultante com o mapa CTC, com a opção de multiplicação o módulo *overlay*;

5. Calcular $NC = \frac{CTC(V2\% - V\%)}{700}$, utilizando a opção divisão por valor constante da função *scalar*.

2. Mapas das quantidades de fertilizantes

Neste exemplo, a geração dos mapas das quantidades de fertilizantes utiliza os dados da tabela 3.1. As quantidades recomendadas na tabela devem ser ajustadas aos fertilizantes a serem aplicados, pois diferentes fertilizantes têm porcentagens distintas do nutriente desejado. Por exemplo, os fertilizantes fosfatados superfosfato simples e superfosfato triplo têm concentrações de P_2O_5 18% e 41%, respectivamente. Portanto, a quantidade necessária de superfosfato triplo é menor do que a de superfosfato simples, para a mesma recomendação.

São entradas para o processo os mapas dos teores de fósforo (P) e de potássio (K), as concentrações dos fertilizantes escolhidos e a produtividade esperada da cultura. Considere a produtividade esperada em torno de 2.7 t/ha e os fertilizantes fosfatado e potássico, respectivamente, o superfosfato simples (41% de P_2O_5) e o cloreto de potássio (58% de K_2O). Os passos para construção dos mapas de quantidades de fertilizantes são os seguintes:

1. Reclassificar o mapa de teores de fósforo atribuindo as quantidades da tabela, nos intervalos 0-6, 7-15, 16-40 e >40, utilizando a função *reclass*;

2. Reclassificar o mapa de teores de potássio, nos intervalos 0-0.7, 0.8-1.5, 1.6-3.0 e >3.0, com a função *reclass*;

3. Ajustar a quantidade de superfosfato simples, usando a opção de divisão do módulo *scalar*; e

4. Ajustar a quantidade de cloreto de potássio, também através da opção de divisão do módulo *scalar*.

As saídas do processo global são os mapas de quantidades de calcário, superfosfato simples e cloreto de potássio. Em um ciclo completo de agricultura de precisão, estes

mapas seriam fornecidos para implementos agrícolas dotados de sensores de localização que fariam a aplicação localizada dos insumos.

A próxima seção mostra como a nova versão do WOODSS poderia ser utilizada para auxiliar o usuário na solução do exemplo em questão.

5.2 Interação do usuário com o WOODSS

Como citado anteriormente, o WOODSS permite ao usuário três tipos de interação: (1) diretamente com o SIG, usando o WOODSS para documentar suas operações; (2) diretamente com o WOODSS, construindo um *workflow* e executando-o no SIG; e (3) alternando entre o SIG e o WOODSS. Continuando o exemplo apresentado, a seção 5.2.1 considera a primeira interação, ilustrando a documentação e o armazenamento deste processo decisório. A seção 5.2.2 descreve as opções de recuperação de casos providas pelo WOODSS.

5.2.1 Documentação e armazenamento de um caso

Ao iniciar o WOODSS, surge uma janela de autenticação, onde o usuário se identifica fornecendo sua senha. O usuário tem acesso, então à tela principal do sistema (figura 5.1). Através desta tela ele pode capturar interações no SIG, carregar um *workflow* do banco de dados ou criar um novo *workflow* vazio. Estas operações podem ser acionadas através dos menus ou dos botões da barra de ferramentas.

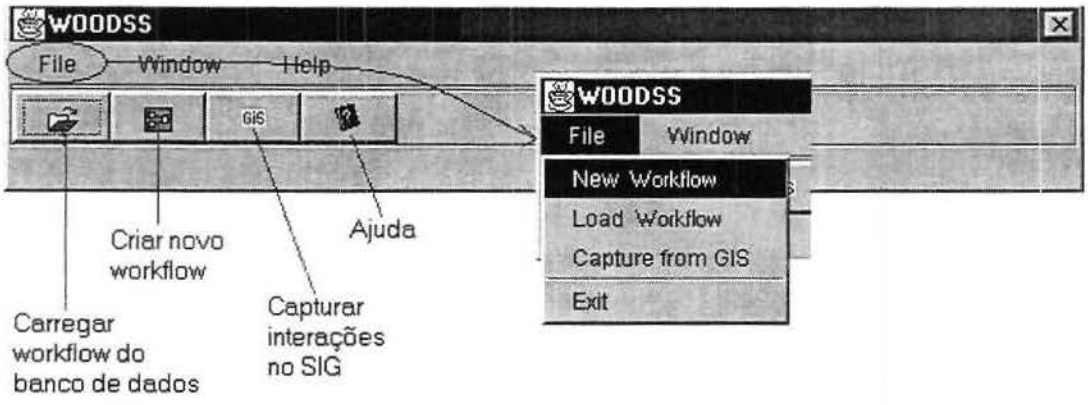

Figura 5.1: Tela principal da nova versão do WOODSS.

Clicando em "*Capture from GIS*" no menu "*File*" o WOODSS gera um novo *workflow* capturando as interações do usuário a partir de um arquivo de *log* gerado pelo Idrisi. Esta versão do WOODSS permite a manipulação de vários *workflows* simultaneamente, em

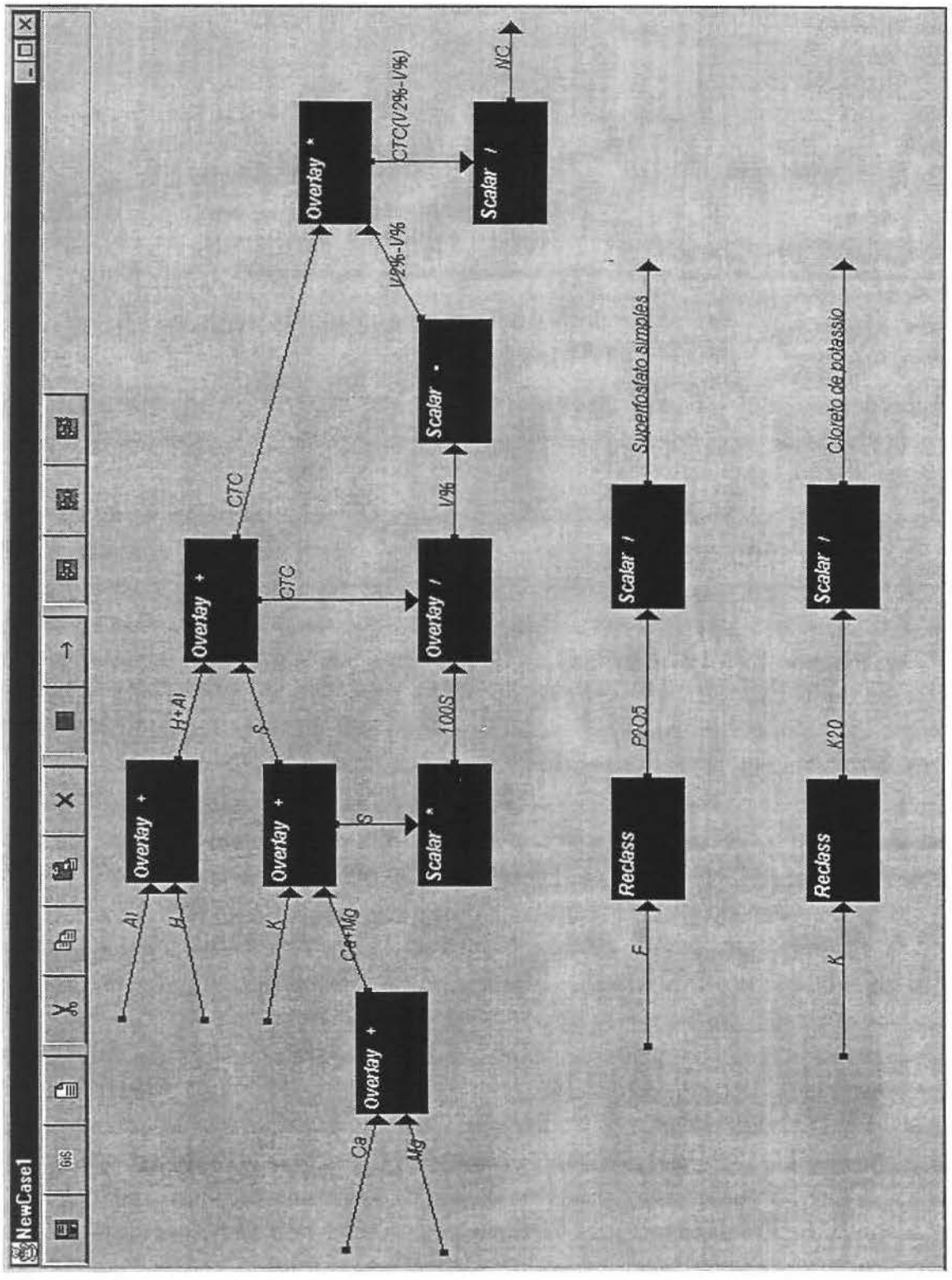

Figura 5.2: Janela de manipulação de *workflows*, apresentado o *workflow* do exemplo.

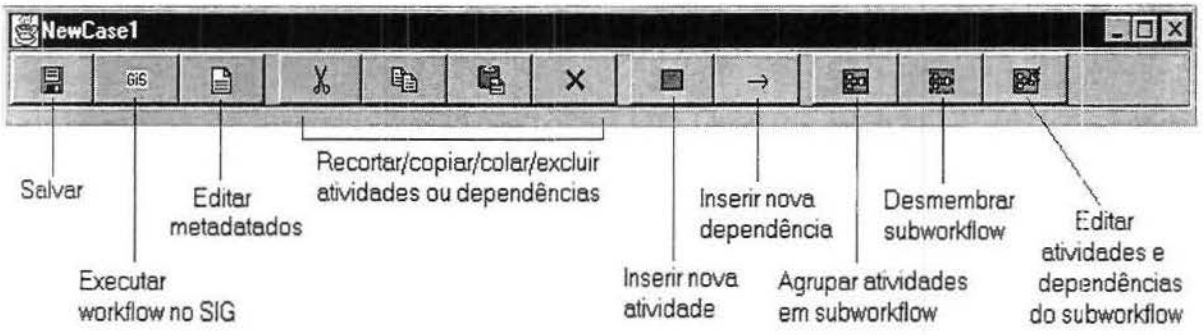

Figura 5.3: Funções dos botões da janela de manipulação de *workflows*.

janelas independentes subordinadas à janela principal. A figura 5.2 mostra uma janela de manipulação de *workflows* da nova versão do WOODSS, que apresenta o *workflow* resultante do exemplo.

Uma janela de manipulação de *workflow* dá acesso às operações de edição do respectivo *workflow*. O usuário pode editar os seus metadados, inserir, remover e alterar atividades e dependências e manipular *subworkflows*, agrupando e desagrupando conjuntos de atividades. Também através desta janela, o usuário pode solicitar o armazenamento do novo caso e a execução dos *workflows* no SIG. Dado que o Idrisi não permite que outros programas executem operações de forma direta, na verdade o WOODSS gera um arquivo de macro, que é disponibilizado ao usuário para que este o execute no SIG. A figura 5.3 resume as funções dos botões da janela de manipulação de *workflows*.

Para armazenar o novo caso, o usuário realiza o procedimento a seguir. Clicando no botão *"Salvar"*, o sistema apresenta o conjunto de metadados do caso para que seja atualizado. A figura 5.4 mostra os metadados fornecidos pelo usuário para o exemplo. Note que os metadados podem ser editados em qualquer momento durante a construção do *workflow*, clicando-se no botão *"Editar metadados"* na tela de manipulação de *workflows*.

Em seguida, aparece uma tela para a determinação dos descritores de indexação do novo caso, que também pode ser visualizada clicando-se no botão rotulado *"Descriptors..."*, na janela de metadados. Primeiramente, o usuário deve definir os objetivos do novo caso. Clicando no botão rotulado *"Add"* (figura 5.5), ele tem acesso a uma tela de busca dos objetivos armazenados. O usuário fornece palavras-chave associadas ao novo caso e o sistema apresenta os objetivos que satisfazem sua consulta. Os valores percentuais situados antes das descrições dos objetivos são seus valores de similaridade, considerando os descritores da consulta. O botão rotulado *"Details..."* apresenta os descritores (palavras-chave e restrições) dos objetivos selecionados, permitindo sua edição.

Conforme citado no capítulo 3, as palavras-chave, os atributos das restrições e os valores do tipo devem constar no dicionário de termos do WOODSS. A figura 5.6 mostra

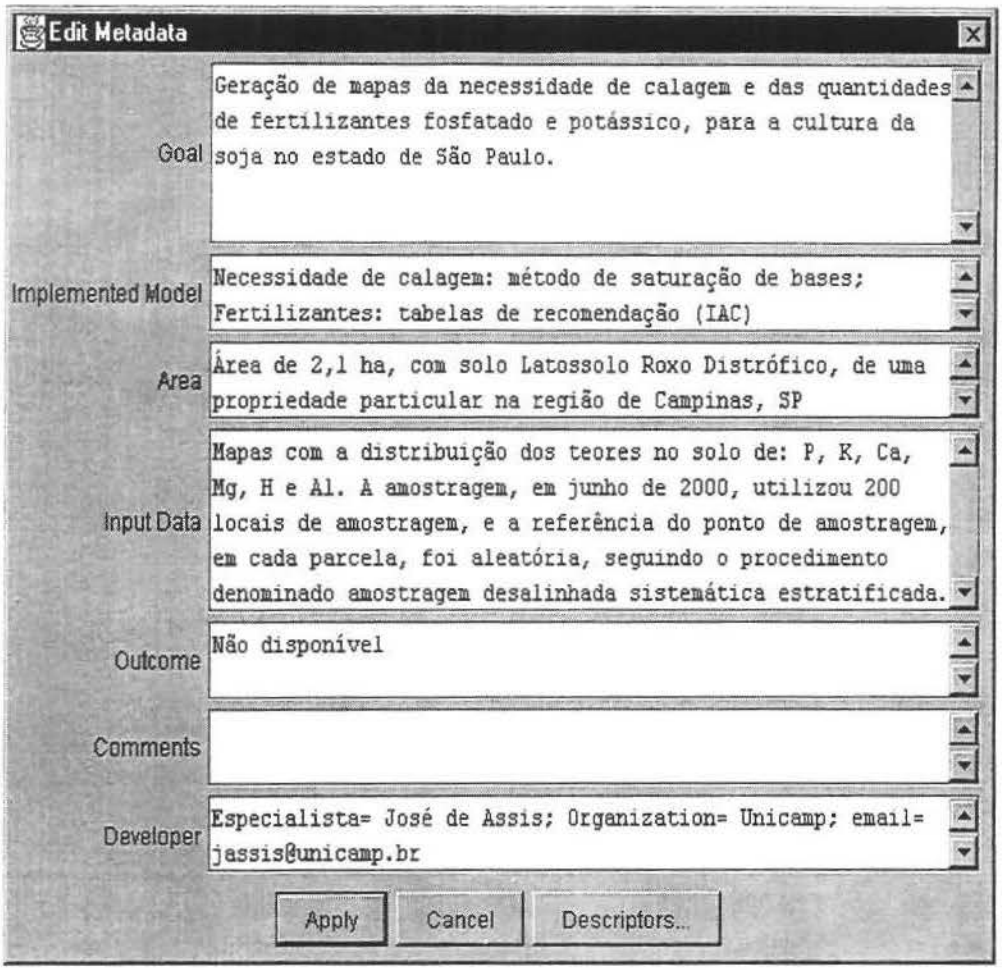

Figura 5.4: Metadados inseridos pelo usuário para o exemplo.

a janela de manipulação do dicionário, representado na árvore à esquerda da janela. Os relacionamentos hierárquicos são atualizados diretamente nesta árvore, enquanto os relacionamentos de equivalência e as descrições dos termos são editados em caixas de texto. O usuário pode obter termos do dicionário do sistema a partir dos botões rotulados apenas com reticências nas demais janelas de determinação de descritores.

Se os objetivos existentes são insuficientes para documentar o novo caso, o usuário pode criar um novo objetivo clicando no botão *"Add"*. A figura 5.7 mostra uma cópia de tela na qual o usuário cria ou edita um objetivo. Nesta janela o usuário fornece uma descrição do objetivo, atribui palavras-chave (descritores do tipo 1) e restrições (descritores do tipo 2) ao objetivo, definindo o atributo, o tipo, o peso e a descrição de cada restrição. Voltando à janela da figura 5.5 o usuário seleciona os objetivos desejados e adiciona-os ao novo caso ou refina a busca de objetivos.

Definidos os objetivos do novo caso, o usuário preenche os valores e comparandos das

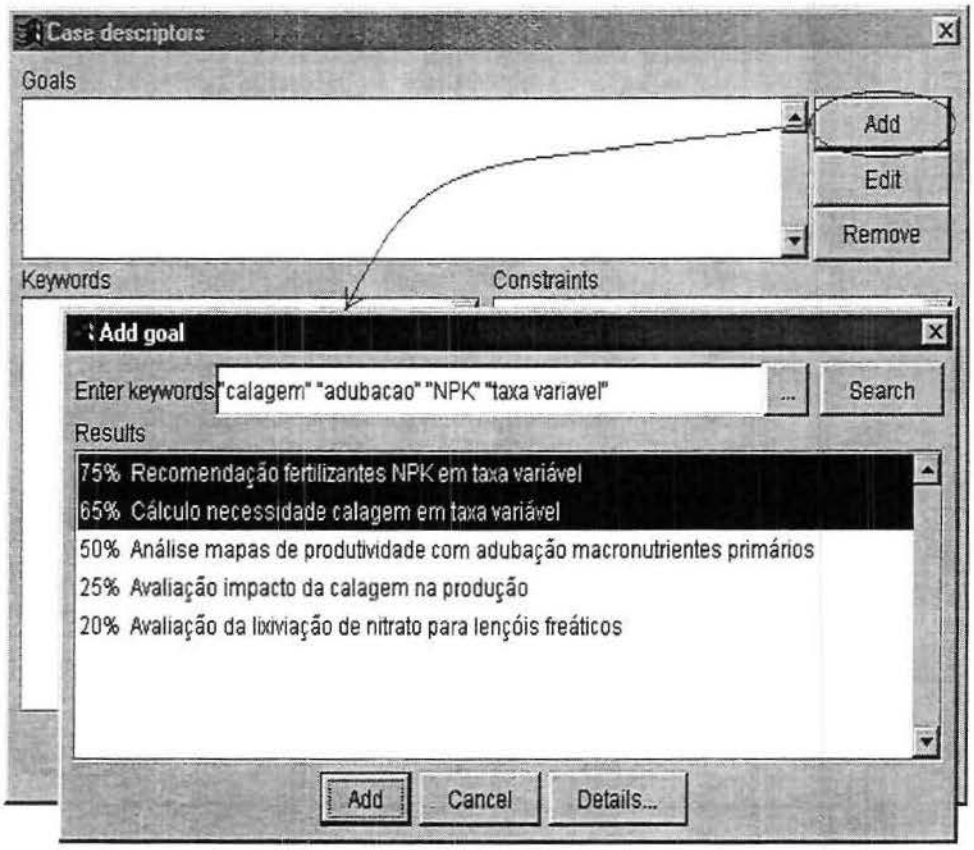

Figura 5.5: Janelas para determinação dos objetivos do caso.

restrições relevantes para o novo caso na janela da figura 5.8. Apenas as restrições com valores atribuídos são armazenadas como descritores do novo caso.

O sistema então realiza uma busca, tomando os descritores fornecidos e apresenta os casos mais semelhantes encontrados (vide figura 5.9). O usuário pode analisar os casos retornados clicando no botão rotulado "*Details...*" e decidir por cancelar a retenção do novo caso (botão "*Cancel*"), voltar e ajustar os descritores fornecidos (botão "*Back*") ou efetivar a retenção (botão "*Save*").

5.2.2 Recuperação de casos

Suponha agora a interação do usuário diretamente com o WOODSS, recuperando casos da memória de casos e utilizando-os em novos processos decisórios. Clicando em *"Load workflow"* no menu *"File"* (figura 5.1) o usuário tem acesso às janelas de busca do sistema. A nova versão do WOODSS provê duas formas de busca: em um ou dois passos.

Na busca em um passo (figura 5.10), o usuário entra com todos os descritores desejados (tipos 1 e 2) de uma só vez. Na formulação da consulta, os termos devem ser colocados

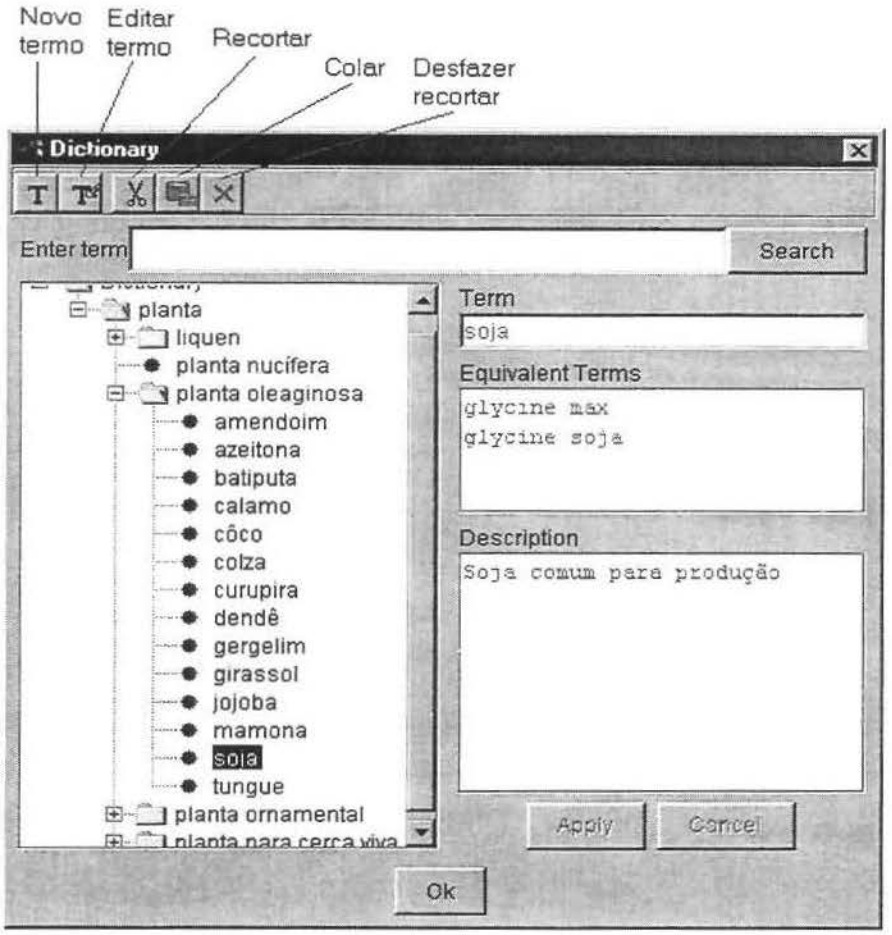

Figura 5.6: Janela para manipulação de termos do dicionário.

entre aspas e as restrições na ordem <atributo, comparando, valor>. Os termos podem ser obtidos a partir do dicionário do sistema. O usuário pode então selecionar um ou mais casos e carregá-los em janelas independentes de manipulação de *workflows* (botão *"Open"*). Clicando no botão *"Details..."* o sistema abre uma janela de metadados para cada caso selecionado e o botão *"Two step search"* alterna para a busca em dois passos, que é bastante semelhante, variando apenas o fornecimento dos descritores.

A principal vantagem fornecida pelo novo mecanismo de recuperação de casos é que o sistema realiza uma busca elaborada, procurando identificar similaridades semânticas entre o problema de entrada e os casos armazenados. Esta funcionalidade facilita a escolha de modelos adequados aos problemas enfrentados pelos tomadores de decisão no domínio, resultando em menor esforço, reutilizando soluções, e em conseqüente aumento de produtividade e eficácia.

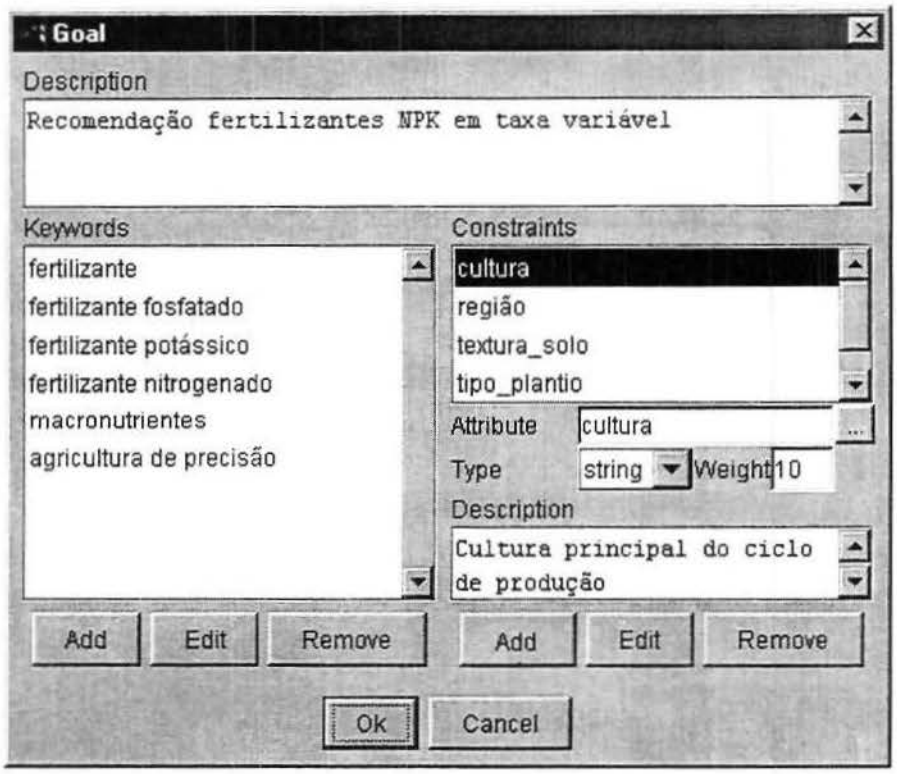

Figura 5.7: Janela de edição de objetivos.

5.3 Resumo

O capítulo descreveu como o WOODSS pode ser usado pelos especialistas da área de planejamento agro-ambiental como ferramenta que auxilia seus processos decisórios. O exemplo apresentado consiste na utilização do WOODSS na aplicação real de recomendação de fertilizantes e corretivos em taxa variável, mostrando a interação do usuário no armazenamento e recuperação de casos.

O próximo capítulo apresenta conclusões e contribuições desta dissertação e propõe algumas extensões aos mecanismos propostos, sob os pontos de vista conceitual e de implementação.

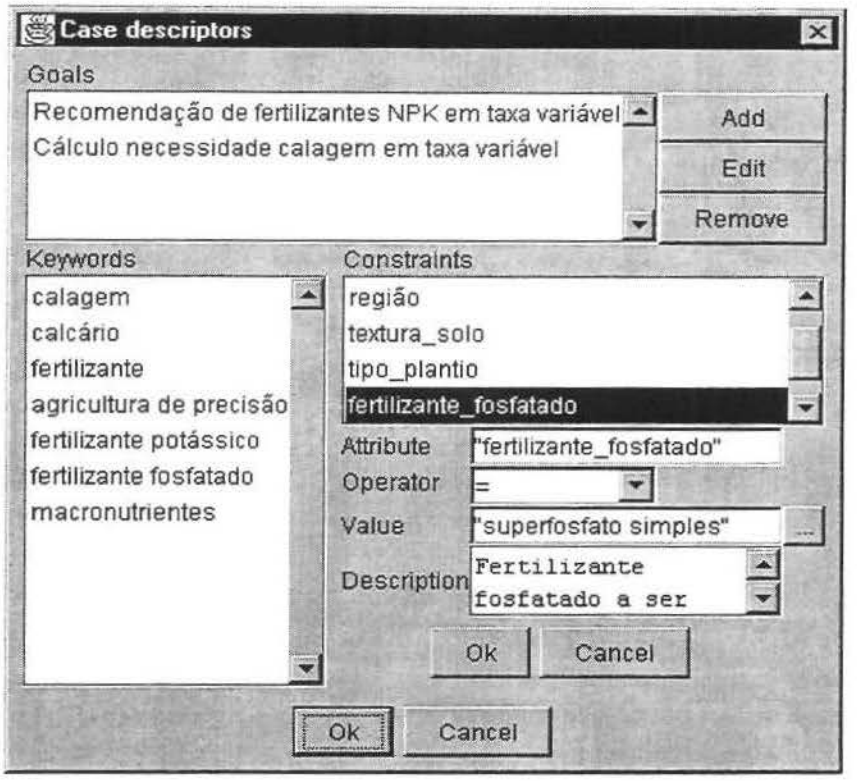

Figura 5.8: Janela com os descritores atribuidos para o exemplo.

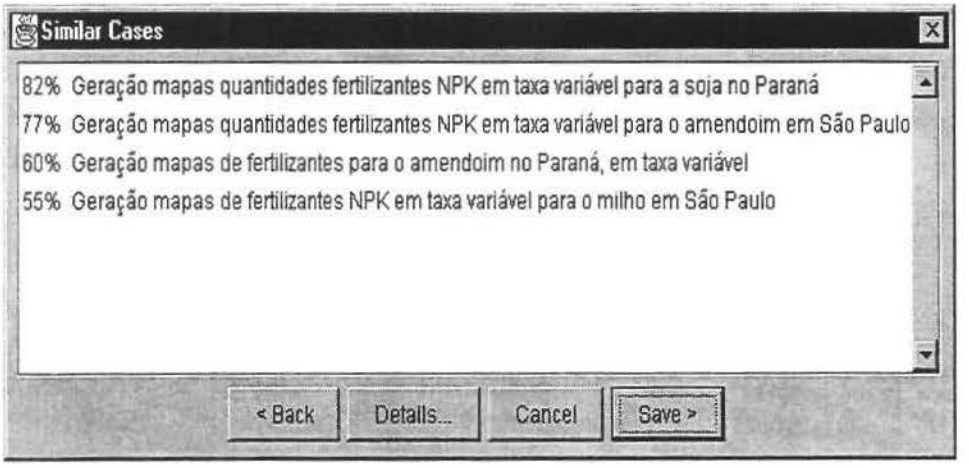

Figura 5.9: Casos armazenados similares ao novo caso.

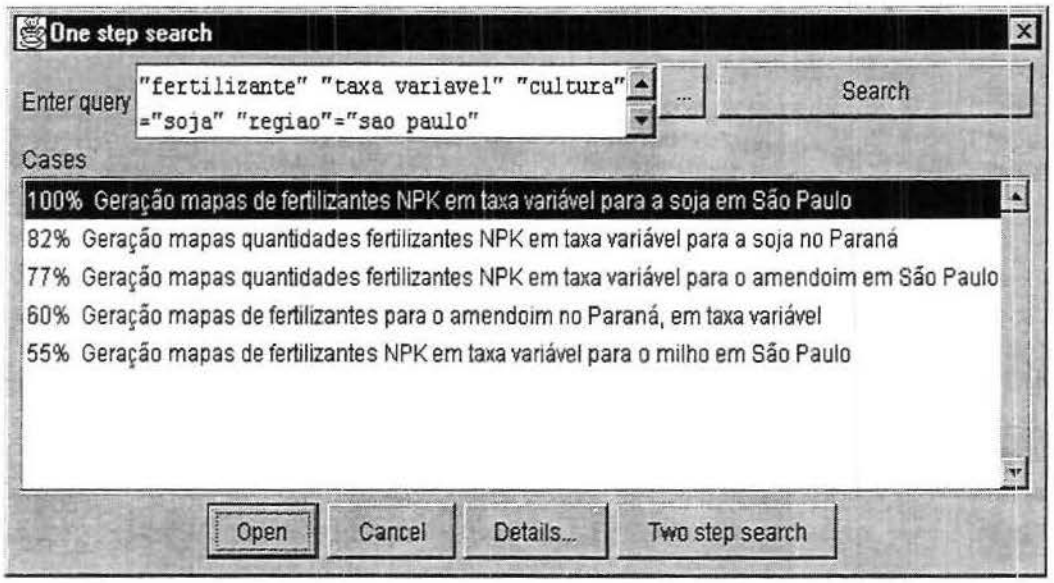

Figura 5.10: Cópia da tela de busca em um passo.

Capítulo 6

Conclusões e extensões

6.1 Contribuições

Esta dissertação concentrou-se em oferecer esquemas para auxiliar tomadores de decisões no domínio ambiental a identificar modelos adequados aos seus problemas, permitindo a geração de soluções a partir de precedentes. Sistemas espaciais de apoio à decisão em geral são deficientes neste sentido, em vista da quantidade de características inter-relacionadas que precisam ser consideradas. A proposta deste trabalho foi aplicar a noção de CBR como estratégia de ataque a este problema, considerando instâncias de implementação de modelos ambientais em um SIG como casos.

Inicialmente, foi realizada uma revisão de CBR e um estudo de alguns problemas do domínio, analisando seu processo de solução utilizando um SIG. Este estudo permitiu o levantamento de requisitos para documentação e indexação de casos neste contexto, visando antecipar reutilizações futuras e possibilitar a identificação de semelhanças entre situações.

Seguiu-se a especificação de algoritmos para a recuperação e retenção de casos, usando técnicas de CBR. Como base para estes algoritmos foi desenvolvido um esquema de análise de similaridade entre casos, fundamentado no algoritmo de vizinhos mais próximos. O algoritmo de análise de similaridade diferencia as características dos casos em dois níveis: os seus objetivos e as restrições a estes objetivos. Estas características são materializadas em descritores de indexação dos tipos palavras-chave e predicados, formados por valores numéricos e textuais. A comparação de valores textuais baseia-se em percursos em hierarquias de conceitos e a de valores numéricos em sua diferença algébrica. O valor final de similaridade entre dois casos é dado pela média ponderada da similaridade entre os descritores de seus objetivos e restrições.

Esta dissertação tomou como ponto de partida o sistema WOODSS [Sef98, SMRY99], um SDSS ambiental desenvolvido no LIS-IC/UNICAMP. Este sistema foi recodificado,

acoplando os mecanismos de recuperação e retenção de casos propostos e tornando-o mais aberto para acomodar expansões futuras. O mecanismo de recuperação baseado em CBR desenvolvido possibilita uma recuperação mais precisa, facilitando a identificação dos casos mais relevantes para um novo processo decisório.

As principais dificuldades deste trabalho residem na abrangência do domínio enfocado e na conseqüente flexibilidade exigida dos mecanismos propostos. As técnicas de CBR avaliadas mostraram-se limitadas frente a este panorama, devido à excessiva complexidade das análises consideradas ideais. Isto levou ao desenvolvimento de algoritmos genéricos de recuperação e retenção de casos, podendo ser aplicados a outros domínios sem maiores modificações.

É importante ressaltar que a eficácia da recuperação de casos depende diretamente da organização do dicionário de termos. As abstrações e equivalências devem ser cuidadosamente definidas, para que as distâncias entre os nós das hierarquias reflitam adequadamente suas distâncias conceituais.

Todavia, foram identificadas características que fazem do CBR uma tecnologia promissora para apoio à decisão ambiental, em domínios mais restritos. Estas características, discutidas na seção 3.1, incluem a modelagem a partir de instâncias, a forma de aquisição de conhecimento e a reutilização de soluções.

As principais contribuições desta dissertação foram:

- Levantamento das pré-condições necessárias para a aplicação de CBR para apoio à decisão no contexto ambiental, acoplando este conceito a SIG e bancos de dados;
- Especificação de algoritmos de recuperação e retenção de casos utilizando uma nova proposta de análise de similaridade; e
- Extensão do sistema WOODSS, tornando-o mais efetivo para a manipulação de modelos, através do acoplamento de mecanismos baseados nestes algoritmos, permitindo a resolução de problemas a partir de precedentes.

6.2 Extensões

Diversas extensões podem ser propostas a esta dissertação. Dentre as futuras direções possíveis desta pesquisa, destacam-se:

Considerar a estrutura dos workflows em um nível semântico na análise de similaridade. Este trabalho descartou a possibilidade de utilizar a "topologia" dos *workflows* na avaliação de similaridade entre casos, conforme justificado na seção 3.4. Entretanto, uma proposta factível seria analisar a estrutura dos *workflows* em um nível

de abstração mais próximo do senso do especialista, alcançado através de agrupamentos de atividades complementares em *subworkflows*. Neste nível de abstração, já existem informações implícitas nas estruturas dos *workflows* que podem sugerir com maior grau de precisão o significado do caso. Desta forma, uma extensão a esta dissertação seria considerar na análise de similaridade as informações descritivas associadas aos casos e as estruturas de seus *workflows*.

Desenvolver esquemas para inferência automática de descritores. A determinação de descritores é a tarefa mais importante do processo de retenção de casos. O mecanismo de retenção apresentado nesta dissertação delega esta tarefa ao usuário, o que tende a ser demasiadamente cansativo e muitas vezes impreciso. Embora o domínio seja abrangente, uma extensão importante seria o desenvolvimento de esquemas que automatizem esta tarefa, pelo menos em parte.

Utilizar algoritmos para auxiliar na determinação dos pesos das características. Os mecanismos desenvolvidos também deixam a definição de pesos de características a cargo do usuário. Assim, esta tarefa torna-se subjetiva e heterogênea, resultando em avaliações errôneas de similaridade. Existem algumas propostas de algoritmos para determinação de pesos que poderiam ser adaptados para o WOODSS, em especial o proposto por Wettschereck e Aha [WA95], que considera pouco ou nenhum conhecimento específico do domínio.

Considerar restrições mais complexas. Os algoritmos propostos nesta dissertação consideram apenas restrições sob a forma de predicados formados por conjunções de termos. Uma outra extensão aos esquemas aqui apresentados seria permitir predicados mais complexos, uma vez que apenas conjunções de termos são muitas vezes insuficientes para representar uma restrição.

Especificar algoritmos de busca sensíveis ao georeferenciamento. Como citado na seção 3.7, o algoritmo de recuperação desenvolvido não trata atributos georeferenciados de modo especial. Um trabalho futuro consiste em aprimorar o algoritmo 3.5, que apresenta um esboço de um algoritmo sensível ao georeferenciamento, e implementá-lo aproveitando funções do SIG.

Utilizar estruturas de indexação escaláveis para a memória de casos. Um dos maiores problemas da estratégia de vizinhos mais próximos é que, salvo em casos especiais, todo o conjunto de casos precisa ser analisado. Assim, o desempenho da busca degrada-se com o crescimento do conjunto de casos armazenados. Além disso, em geral as estruturas

de memória de casos são montadas em memória, e, portanto, limitadas ao seu tamanho. Uma opção promissora é o uso de árvores M [CPZ97]. A árvore M é uma estrutura de dados multidimensional balanceada, projetada para suportar acessos a disco, que indexa casos considerando distâncias relativas entre objetos (espaço métrico – daí arvore M). Esta característica dá à árvore M uma flexibilidade para indexar múltiplos tipos de objetos, podendo ser embutida em muitas aplicações CBR.

Restringir o domínio e construir um sistema especialista sobre o WOODSS. Uma extensão mais arrojada seria tomar o WOODSS como base para o desenvolvimento de sistemas especialistas para aplicações que usam dados espaciais. Restringindo-se o domínio de atuação é possível especializar os mecanismos de recuperação e retenção do WOODSS, ajustando descritores e pesos, e gerar um conjunto de regras de adaptação de *workflows*. O sistema especialista poderia, então, usar primitivas do WOODSS para recuperar casos, editá-los de acordo com as regras de adaptação definidas e executá-los no SIG, obtendo os mapas resultantes e apresentando-os ao usuário final.

Aprimorar a interface com o usuário. Outro trabalho futuro, de menor importância do ponto de vista de pesquisa, mas essencial para a usabilidade do WOODSS, refere-se ao seu projeto de interface. Este deve ser aprimorado para facilitar a interação do usuário, em termos de organização de janelas, rotulação de botões e seu posicionamento na tela. Além disso, seria interessante disponibilizar opções de ajuda sensíveis ao contexto para auxiliar na utilização dos mecanismos que o WOODSS oferece.

Bibliografia

[AP94] A. Aamodt e E. Plaza. Case-based reasoning: foundational issues, methodological variations and system approaches. *AI Communications*, 7(1):39–59, 1994.

[Bar89] R. Bareiss. *Exemplar-based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification and Learning.* Academic Press, Boston, 1989.

[Bar96] P. Barthelmess. Sistemas de workflow: Análise da Área e proposta de modelo. Dissertação de Mestrado, IMECC-UNICAMP, 1996.

[BH94] K. L. Branting e J. D. Hastings. An empirical evaluation of model-based case matching and adaptation. Em *Case-Based Reasoning Workshop*, páginas 72–78, Seattle, Washington, 1994. American Association for Artificial Intelligence.

[CCH+96] G. Câmara, M. A. Casanova, A. S. Hemerly, G. C. Magalhães, e C. B. Medeiros. *Anatomia de Sistemas de Infomação Geográfica.* Instituto de Computação - UNICAMP, 1996.

[CEN97] Ministerio da Agricultura CENAGRI. Thesagro – thesaurus agricola nacional, 1997.

[Che97] W. Cheetham. Case-based reasoning in color matching. Em D. Leake e E. Plaza, editores, *Proceedings of ICCBR'97.* LNAI Springer Verlag, 1997.

[CNP00] Embrapa CNPSoja, editor. *Recomendações Técnicas para a Cultura da Soja no Paraná 1999–2000.* Embrapa Centro Nacional de Pesquisa de Soja (CNPSoja), Londrina, PR, 2000.

[CP91] M. J. Carrascal e L. F. Pau. A survey of expert systems in agriculture and food processing. *AI and Vision*, 1991.

[CPA97] Embrapa CPAO, editor. *Milho: informações técnicas*. Embrapa Centro de Pesquisa Agropecuária do Oeste (CPAO), 1997.

[CPZ97] P. Ciaccia, M. Patella, e P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. Em *Proceedings of the 23rd International Conference on Very Large Databases*, páginas 426–435, 1997.

[CWP95] M. D. Crossland, B. E. Wynne, e W. C. Perkins. Spatial decision support systems: an overview of technology and a test of efficacy. *Decision Support Systems*, 14(3):219–235, Julho 1995.

[Den91] P. J. Densham. Spatial decision support systems. Em D. J. Maguire, M. F. Goodchild, e D. W. Rhind, editores, *Geographical Information Systems: Principles and Applications*, capítulo 26, páginas 403–412. Longman Scientific and Technical, New York, NY, 1991.

[Den96] Pi-Sheng Deng. Using case-based reasoning approach to the support of ill-structured decisions. *European Journal of Operational Research*, 93:511–521, 1996.

[dFdSR94] Comissão de Fertilidade do Solo RS/SC. *Recomendações de adubação e de calagem para os estados do Rio Grande do Sul e de Santa Catarina*. Núcleo Regional Sul, 3 edição, 1994.

[DG90] P. J. Densham e M. F. Goodchild. Spatial decision support systems. Relatório Técnico, NCGIA – National Center for Geographic Information and Analysis, 1990. Technical Report 90-5.

[dPdT99] Comissão Centro Brasileira de Pesquisa de Trigo. *Recomendações para os anos de 1999 e 2000*. EPAMIG, Uberaba, MG, 1999.

[Fed93] K. Fedra. GIS and environmental modeling. Em M. F. Goodchild, B. O. Parks, e L. T. Steyaert, editores, *Environmental Modeling with GIS*, capítulo 5, páginas 35–50. Oxford Univesity Press, New York, NY, 1993.

[Has96] J. D. Hastings. *A Mixed Paradigm Approach to Problem Solving in Incomplete Causal-Theory Domain*. Tese de Doutorado, University of Wyoming, Laramie, Wyoming, 1996.

[HB99] A. Holt e G. L. Benwell. Applying case-based reasoning techniques in gis. *The International Journal of Geographical Information Science*, 13(1):9–25, 1999.

[JFU95] G. Jacucci, M. Foy, e C. Uhrik. An artificial intelligence methodology for the adaptation of agricultural models. *Engineering Applications of Artificial Intelligence*, 1995. Pergamon Press Ltd., Oxford.

[Kee97] P. Keenan. Using GIS as a DSS generator. Working Paper MIS 95-9, Department of Management Information Systems, Faculty of Commerce, University College Dublin, Dublin, Ireland, Abril 1997.

[KJ91] J. L. Kolodner e M. Y. Jona. Case-based reasoning: An overview. Relatório Técnico 15, Northwestern University, Junho 1991.

[Kol84] J. L. Kolodner. *Retrieval and Organization Strategies in Conceptual Memory: A Computer Model.* Erlbaum, Northvale, NJ, 1984.

[Kol93] J. L. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[Kot89] P. Koton. *Using Experience in Learning and Problem Solving.* Tese de Doutorado, Department of Computer Science, MIT, 1989.

[LFK97] Y. F. D. Law, S. B. Foong, e S. E. J. Kwan. An integrated case-based reasoning approach for intelligent help desk fault management. *Expert Systems with Applications*, 13(4):265–274, 1997.

[LGM99] A. S. Lopes, L. R. Guilherme, e R. Marques. Guia de fertilidade do solo – versão multimídia. CD-ROM, 1999.

[Mag91] D. J. Maguire. An overview and definition of GIS. Em D. J. Maguire, M. F. Goodchild, e D. W. Rhind, editores, *Geographical Information Systems: Principles and Applications*, capítulo 1, páginas 9–20. Longman Scientific and Technical, New York, NY, 1991.

[MD88] W. R. Moninger e R. M. Dyer. Survey of past and current ai work in the environmental sciences. *AI Applications in Natural Research Management*, 2(1):48–52, 1988.

[Men95] J. Mendel. Fuzzy logic systems for engineering: A tutorial. *Proc. IEEE*, 83(3), 1995.

[MSC99] K. B. Matthews, A. R. Sibbald, e S. Craw. Implementation of a spatial decision support system for rural land use planning: Integrating geographic information system and environmental models with search and optimization algorithms. *Computers and Electronics in Agriculture*, 23:9–26, 1999.

[Pir97] F. Pires. *Um Ambiente Computacional para Modelagem de Aplicações Ambientais.* Tese de Doutorado, IC-UNICAMP, Dezembro 1997. Orientação C. B. Medeiros.

[QDM00] D. M. Queiroz, G. P. Dias, e E. C. Mantovani. Agricultura de precisão na produção de grãos. Em Borém et. al., editor, *Agricultura de Precisão*, páginas 1–42. Universidade Federal de Viçosa, Viçosa, MG, Brasil, 2000.

[RCQF96] B. Raij, H. Cantarella, J. A. Quaggio, e A. M. Furlani, editores. *Recomendações de adubação e calagem para o estado de São Paulo.* Instituto Agronômico de Campinas e Fundação IAC, 2 edição, 1996. Boletim técnico.

[RS89] C. K. Riesbeck e R. C. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.

[RY97] A. E. Rizzoli e W. J. Young. Delivering environmental decision support systems: Software tools and techniques. *Environmental Modelling and Software*, 12(2–3):237–249, 1997.

[SB98] M. J. Shaffer e M. K. Brodahl. Rule-based management for simulation in agricultural decision support systems. *Computers and Electronics in Agriculture*, 21:135–152, 1998.

[Sch82] R. C. Schank. *Dynamic Memory: A Theory of Learning in Computers and People.* Cambridge University Press, 1982.

[SCH00] A. M. Saraiva, C. E. Cugnasca, e A. R. Hirakawa. Aplicação em taxa variável de fertilizantes e sementes. Em Borém et. al., editor, *Agricultura de Precisão*, páginas 109–145. Universidade Federal de Viçosa, Viçosa, MG, Brasil, 2000.

[Sef98] L. Seffino. Woodss - sistema espacial de apoio ao processo decisório baseado em workflows. Dissertação de Mestrado, IC-UNICAMP, Julho 1998. Orientação C. B. Medeiros.

[SH99] K. Shin e I. Han. Case-based reasoning supported by genetic algorithms for corporate bond rating. *Expert Systems with Applications*, 16:85–95, 1999.

[SMRY99] L. Seffino, C. B. Medeiros, J. V. Rocha, e Bei Yi. WOODSS - a spatial decision support system based on workflows. *Decision Support Systems*, 27(1–2):105–123, Novembro 1999.

[Sno86] R. T. Snodgrass. Temporal databases. *IEEE Computer*, páginas 35–42, Setembro 1986.

[Ste93] L. T. Steyaert. A perspective on the state of environmental simulation modeling. Em M. F. Goodchild, B. O. Parks, e L. T. Steyaert, editores, *Environmental Modeling with GIS*, capítulo 3, páginas 16–30. Oxford Univesity Press, New York, NY, 1993.

[SV95] J. Surma e K. Vanhoof. Integrating rules and cases for the classification task. Em *Proceedings of the 1st International Conference on Case-Based Reasoning - ICCBR'95*, páginas 325–334, Sesimbra, Portugal, 1995. Springer.

[SW89] R. H. Sprague e H. J. Watson. *Decision Support Systems: Putting Theory into Practice*, capítulo 1, páginas 9–35. Prentice Hall, 2 edição, 1989.

[VB99] F. Verdenius e J. Broeze. Generalized and instance-specific modelling for biological systems. *Environmental Modelling and Software*, 14:339–348, 1999.

[WA95] D. Wettschereck e D. W. Aha. Weighting features. Em *Proceedings of 1st International Conference on Case-Based Reasoning - ICCBR'95*, Sesimbra, Portugal, 1995. Springer.

[Wat97] I. D. Watson. *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann, 1997.

[Wat99] I. D. Watson. Case-based reasoning is a methodology not a technology. *Knowledge-Based Systems*, 12:303–308, 1999.

[WI97] Y. Wang e N. Ishii. A method of similarity metrics for structured representations. *Expert Systems with Applications*, 12(1):89–100, 1997.

[WVMP98] M. Weske, G. Vossen, C. B. Medeiros, e F. Pires. Workflow management in geoprocessing applications. Em *Proceedings of 6th ACM International Symposium on Geographical Information Systems - ACMGIS'98*, páginas 88–93, 1998.

[WWVM96] J. Wainer, M. Weske, G. Vossen, e C. B. Medeiros. Scientific workflow systems. Em *Proceedings of the NSF Workshop on Workflow Process Automation: State-of-the-art and Future Directions*, 1996.