UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE
ÁREA: CIÊNCIA DA COMPUTAÇÃO

PROPOST: UMA FERRAMENTA BASEADA EM CONHECIMENTO PARA GESTÃO DE PORTIFÓLIO DE PROJETOS

EDUARDO NEWTON OLIVEIRA VIEIRA

São Luís, MA
2007

PROPOST: UMA FERRAMENTA BASEADA EM CONHECIMENTO PARA GESTÃO DE PORTIFÓLIO DE PROJETOS

Eduardo Newton Oliveira Vieira

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Eletricidade na área de Ciência da Computação**.**

Orientadora: Profª. Dra. Rosario Girardi

São Luís, MA

2007

Vieira, Eduardo Newton Oliveira.

PROPOST: Uma ferramenta baseada em conhecimento para Gestão de Portifólio de Projetos / Eduardo Newton Oliveira Vieira. – 2007.
206f.

Texto impresso (fotocópia)
Orientador: Rosário Girardi
Dissertação (Mestrado) – Programa de Pós-graduação em Engenharia de Eletricidade, Área de concentração: Ciência da Computação, Universidade Federal do Maranhão, 2007.

1. Sistemas de Informação Gerencial. 2. Gestão de Portifólio. 3. Propost. I. Girardi, Rosário. II. Título

CDU 004.78:65

EDUARDO NEWTON OLIVEIRA VIEIRA

PROPOST: UMA FERRAMENTA BASEADA EM CONHECIMENTO PARA GESTÃO DE PORTIFÓLIO DE PROJETOS

Dissertação de Mestrado apresentada à Coordenação do Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como requisito para a obtenção do título de Mestre em Engenharia de Eletricidade, na área de Ciência da Computação.

Aprovada em / /

BANCA EXAMINADORA

_____

Profª Drª Maria Del Rosario Girardi Gutierrez
Orientadora

_____

Prof. Dr. Zair Abdelouahab
Examinador Interno

_____

Prof. Dr. Evandro de Barros Costa
Examinador Externo

A meus pais, minha esposa e filhas.

AGRADECIMENTOS

Agradeço inicialmente a Deus, criador de todo o universo, por permitir a realização deste trabalho;

aos meus pais, Newton e Neusa, pela educação proporcionada, e cujos exemplos na vida prática são as maiores lições que carrego na vida;

à minha esposa Jô pela força e compreensão nas horas difíceis;

às minhas filhas Camila e Giovana, fontes da minha inspiração, por terem vindo ao mundo encher a minha vida de alegria;

aos demais membros da minha família, pela fraternidade e incentivo;

à professora Rosário Girardi, pela orientação e conhecimentos ministrados, os quais foram fundamentais para a realização deste trabalho;

aos colegas do mestrado e grupo de pesquisa GESEC, pelo companheirismo ao longo dessa jornada - em especial ao Lucas Drumond, pela valiosa ajuda na implementação da ferramenta;

aos amigos e colegas da Companhia Vale do Rio Doce, em especial à minha gerência, pela compreensão e colaboração ao longo dessa desafiante e difícil jornada de trabalho e estudo;

aos demais amigos que fiz ao longo da vida, alguns dos quais encontram-se distantes, outros mais próximos – mas todos presentes na minha memória, pelos bons momentos e experiências compartilhados, muitos destes de suma importância na formação da minha personalidade;

e finalmente agradeço a todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho.

“A vida é combate, que os fracos abate, que os fortes, os bravos só pode exaltar”.

Gonçalves Dias

RESUMO

Este trabalho apresenta a PROPOST *(Project Portfolio Support Tool)*, uma ferramenta baseada em conhecimento, para suporte à Gestão de Portifólio de Projetos – um modelo de gestão em ascensão na atualidade. Esta ferramenta possui seu foco no processo de definição de projetos, e foi modelada usando a metodologia MAAEM e a ferramenta dirigida por ontologias ONTORMAS, bem como pelo reuso das ontologias ONTOINFO e ONWOWUM, as quais descrevem famílias de produtos de software para o desenvolvimento de aplicações nas áreas de Recuperação e Filtragem de Informação, respectivamente.

A PROPOST objetiva promover a otimização de recursos através da reutilização de sistemas de informação existentes, bem como evitar duplicidade na definição de projetos para a composição do portifólio de software das empresas. Sendo assim, a concepção desta ferramenta objetivou contribuir para a solução de um problema da atualidade, relacionado à redundância na composição do portifólio de projetos, bem como suporte a outras atividades relacionadas à gestão do portifólio (seleção, priorização e avaliação).

O desenvolvimento da PROPOST também serve de referência sobre as contribuições das ontologias no processo de desenvolvimento de software. Adicionalmente, esse trabalho também constituiu um estudo de caso para avaliação da metodologia MAAEM e da ontologia ONTORMAS usadas no processo de modelagem, tendo proporcionado várias contribuições para a melhoria das mesmas.

Palavras-chave: Gestão de Portifólio; Sistemas de Recomendação; *Mineração de uso*; Filtragem Colaborativa; Recuperação da Informação; Gerência de Projetos.

ABSTRACT

This work introduces PROPOST *(Project Portfolio Support Tool)*, a knowledge-based software tool for supporting Project Portfolio Management – an increasing management model nowadays. This tool focuses on a project definition process, and was modeled using the MAAEM methodology and the ONTORMAS ontology-driven tool, as well as by reusing the ONTOINFO and ONTOWUM ontologies, which describe software product families for the development of Information Retrieval and Filtering applications, respectively.

PROPOST looks for providing resource optimization by supporting reuse of existing information systems as well as avoiding duplicity on project definition for the composition on the organization's software portfolio. The tool was created not only as a contribution for solving a current problem related to redundancy on portfolio definition, as well as support for several activities related to portfolio management (select, prioritization and evaluation).

The development of PROPOST provides references on how ontology-based development can help in the software development process. It also contributes as a case study for evaluating the MAAEM methodology and the ONTORMAS ontology used in modeling process, having provided several hints for their improvement.

Keywords: Portfolio Management; Recommender Systems; Usage Mining; Collaborative Filtering; Information Recovery; Project Management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AMS	Agent Management System
DDEMAS	Domain Design for Multi-Agent Systems
FI	Filtragem da Informação
FC	Filtragem Colaborativa
FBC	Filtragem Baseada em Conteúdo
FH	Filtragem Híbrida
FIPA	Foundation for Intelligent Physical Agents
FIPA-ACL	Linguagem de comunicação de agentes da FIPA
FM	Feature Matrices
GESEC	Grupo de pesquisa em Engenharia de software e Engenharia do Conhecimento
GPP	Gerência de Portifólio de Projetos
GMP	Gerência de Múltiplos Projetos
GRAMO	Generic Requirements Analysis Method based on Ontologies
HTML	Hypertext Markup Language
JADE	Java Agent Development Framework
JENA	Uma API Java para manipulação de RDF.
KDD	Knowledge Discovery in Databases
K-MEANS	Algoritmo utilizado, dentre outros, para mineração de dados e descoberta de padrões de uso.
MADEM	Multi-agent Domain Engineering Methodology
MAAEM	Multi-agent Application Engineering Methodology
MUW	Mineração de uso da Web
ONTOINFO	Um Modelo de Domínio e *Framework* Multiagente para famílias de aplicações de Recuperação e Filtragem de Informações.

ONTORMAS	Ontology for Reusing Multi-agent Software – Uma ontologia genérica que representa o conhecimento da MAAEM e serve como ferramenta para a análise, projeto e a implementação de aplicações multiagentes, através da reutilização de artefatos de software produzidos na Engenharia de Domínio Multiagente
ONTOWUM	Um Modelo de Domínio e *Framework* Multiagente para a Personalização na Web baseado na Modelagem de usuários e na Mineração de uso
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
PROPOST	Project Portfolio Support Tool
RI	Recuperação da Informação
RDF	Resource Description Framework
RMA	Remote Management Agent
RUP	Rational Unified Process
SRI	Sistema de Recuperação de Informações
SR	Sistema de Recomendação
SMA	Sistema Multiagente
W3C	World Wide Web Consortium
WBS	Work Breakdown Structure
WUM	Web Utilization Miner

SUMÁRIO

1 INTRODUÇÃO

1.1 Problemática e Motivação

Os sistemas de informação estão cada vez mais voltados ao negócio empresarial e ao suporte às decisões estratégicas. Devido a isso, são crescentes as demandas por novos projetos de sistemas nas organizações. Porém, os recursos (pessoal, material e financeiro) estão sempre condicionados a limites estabelecidos no orçamento. Por esse motivo, a otimização do uso dos recursos disponíveis é fundamental para o ganho de produtividade e, nesse contexto, é necessário o conhecimento do portifólio[1] de sistemas de informação existentes, com vista ao seu eventual reaproveitamento, sempre que fosse possível.

Contudo, na prática, a utilização do conhecimento produzido em muitas organizações ainda ocorre, em geral, de maneira aleatória e desestruturada. E no caso do desenvolvimento de projetos de software, um dos problemas resultantes refere-se à redundância na definição de novos projetos devido à falta de conhecimento em relação ao portifólio existente. Tal problema ocorre de maneira mais intensa, normalmente nas organizações: que possuem muitas atividades diversificadas; que atuam em muitas localidades diferentes; que possuem um porte mais acentuado; que herdam muitos sistemas legados nas aquisições e fusões, etc.

No contexto acima mencionado, não são raros os casos, por exemplo, em que uma área gerencial em uma organização pode desenvolver um sistema de informação similar a um já existente, simplesmente por desconhecer o seu portifólio atual de sistemas. Porém, se houvesse uma ferramenta que proporcionasse uma visão global direcionada para o problema acima citado, antes de definir pelo desenvolvimento de novos sistemas, as áreas gerenciais poderiam verificar se já não existe um que atendesse às suas necessidades.

[1] Segundo o PMI (PMBOK, 2004), "um portifólio é uma coleção de projetos ou programas agrupados para facilitar o seu gerenciamento efetivo". Analogamente, o portifólio de sistemas de informação em uma organização refere-se ao conjunto de sistemas existentes na mesma.

A Gestão de Portifólio de Projetos objetiva a seleção, priorização e avaliação dos projetos de uma organização, de forma que somente os mais relevantes e alinhados ao negócio empresarial possam ser desenvolvidos. Contudo, as atuais ferramentas de suporte à Gestão de Portifólio ainda não utilizam tecnologias mais avançadas, tais como a *mineração de uso*, a *recuperação e filtragem de informações*, com vistas à prevenção do problema anteriormente citado.

As necessidades ora mencionadas constituíram a principal motivação para a concepção da ferramenta apresentada neste trabalho. A mesma visa agregar valor à resolução do problema da redundância na definição de projetos nas organizações, apresentando, dentre outras, funcionalidades que possibilitam o reuso de sistemas existentes, bem como visam prevenir a referida duplicidade. E, conseqüentemente, esta ferramenta estaria também contribuindo para a otimização da utilização e alocação dos recursos (pessoal, material e financeiro) existentes na organização.

1.2 Objetivos do Trabalho

1.2.1 Objetivo Geral

Este trabalho possui como objetivo principal a concepção de uma ferramenta de suporte à Gestão de Portifólio de Projetos denominada PROPOST - *Project Portfolio Support Tool*, a qual proporcione ganho de escala no processo de definição de novos projetos, através da reutilização de conhecimento existente nas empresas.

1.2.2 Objetivos Específicos

No sentido de alcançar o objetivo geral pretendido, buscar-se-á atingir os seguintes objetivos específicos:

a) Conceber um processo de definição de projetos, com foco na recomendação de sistemas existentes, o qual será contemplado através das funcionalidades contidas na ferramenta proposta neste trabalho.

b) Avaliar a metodologia MAAEM (LINDOSO, 2006) em todas as suas fases e propor melhorias à mesma, através do desenvolvimento da ferramenta proposta.

c) Avaliar as abstrações reutilizáveis ONTOINFO (GIRARDI, et. al, 2005) (JANSEN PEREIRA, 2006) e ONTOWUM (MARINHO, 2005) (GIRARDI, LINDOSO, 2005c) (GIRARDI, BALBY, 2006) através da sua reutilização no desenvolvimento da ferramenta proposta.

d) Avaliar a ferramenta desenvolvida através da elaboração de um estudo de caso.

1.2.3 Benefícios Esperados

A ferramenta PROPOST tem como principal benefício esperado a economia gerada através da otimização dos recursos existentes. Isso ocorre porque a recomendação proporciona que, antes de decidir sobre o desenvolvimento de um novo sistema, o gerente tenha uma sugestão de sistemas existentes que potencialmente poderiam estar atendendo à sua necessidade – seja total ou mesmo parcialmente. Com isso, a redundância no desenvolvimento de projetos similares, que é muito observada principalmente nas empresas de grande porte e atuam em várias localidades diferentes, poderia estar sendo reduzida ou mesmo eliminada, dependendo das necessidades existentes.

Outro benefício a ser alcançado com esta ferramenta refere-se à otimização no processo de definição de projetos de sistemas de informação nas organizações, bem como também no próprio desenvolvimento destes softwares. Em relação à definição de projetos, o benefício esperado dar-se-á através das funcionalidades propostas na referida ferramenta. Em relação ao processo de desenvolvimento dos sistemas de informação, o benefício esperado dar-se-á através da reutilização do conhecimento armazenado em ontologias.

O desenvolvimento dessa ferramenta também objetiva a avaliação da metodologia MAAEM usada na sua modelagem, cujo benefício esperado consistirá em melhorias a serem incorporadas na mesma.

Adicionalmente, esperamos também estar contribuindo para uma maior divulgação e disseminação, no âmbito das empresas, de técnicas de vanguarda relacionadas ao desenvolvimento de software, tais como: ontologias, recuperação e filtragem, mineração de uso, gestão do conhecimento, abordagem multiagente, etc.

1.3 Organização da Dissertação

Este trabalho está estruturado da seguinte forma. O capítulo 2 aborda conceitos básicos relacionados à Gestão de Portifólio de Projetos, um modelo gerencial baseado no qual a ferramenta PROPOST objetiva dar suporte; também apresenta uma análise das principais ferramentas existentes na atualidade para suporte à GPP. O capítulo 3 faz uma abordagem sobre as principais tecnologias relacionadas ao desenvolvimento da ferramenta ora proposta; também são apresentados conceitos relacionados à Engenharia de Software Multiagente baseada na reutilização, e um resumo das metodologias utilizadas. No capítulo 4 é apresenta toda a modelagem da ferramenta. O capítulo 5 contém a prototipação da interface com o usuário. O capítulo 6 relaciona um estudo de caso para avaliação da performance da ferramenta. E finalmente o capítulo 7 contém a conclusão deste trabalho, bem como os resultados, contribuições e perspectivas futuras.

2 GESTÃO DE PORTIFÓLIO DE PROJETOS

2.1 Considerações Gerais

A inovação – seja esta relativa a novos produtos e serviços, novos processos, novas estratégias, etc - é um fator fundamental para a competitividade, e constitui uma característica comum entre as empresas bem sucedidas (PORTER, 2003). Devido aos sistemas de informação estarem cada vez mais voltados ao negócio empresarial e ao suporte a decisões estratégicas nas empresas, há uma crescente demanda pelo desenvolvimento de novos sistemas (KENDALL, 2004).

Porém, mesmo nas organizações que possuem grande lucratividade, os seus recursos (pessoal, material e financeiro) vão estar sempre condicionados a limites estabelecidos no planejamento do seu orçamento. Sendo assim, para a geração de novos projetos de sistemas de informação, é preciso que as empresas possuam antes de tudo um conhecimento prévio do conteúdo já existente em seu portifólio, com vistas a seu eventual reaproveitamento, caso haja necessidade (LEVINE, 2005). Isso é necessário não apenas para evitar redundâncias no desenvolvimento, como também proporcionar uma melhor utilização dos recursos existentes na produção dos novos projetos.

A Gestão de Portifólio é um modelo de gestão em ascensão na atualidade, e proporciona uma visão efetiva relacionada ao portifólio existente. A mesma envolve um processo dinâmico de *Avaliação, Seleção e Priorização* de projetos, de forma que somente os mais relevantes e estratégicos à organização sejam desenvolvidos, e nesse contexto, a otimização de recursos é fundamental (COOPER, 2001). Mas desenvolver apenas novos produtos bem sucedidos em termos de prazo, custo e qualidade não é o bastante. É preciso que estes produtos estejam alinhados à estratégia de atuação das empresas no mercado onde as mesmas atuam, a qual é definida pela Gestão Estratégica do negócio (PATTERSON, 1999) (PORTER, 2003). Isso indica ser fundamental o alinhamento entre a Gestão de Portifólio e a Gestão Estratégica do negócio das empresas. E é esse alinhamento que proporcionará a criação de produtos para agregar valor, de acordo com as expectativas da corporação.

2.2 Relação entre Gestão de Portifólio e Gestão Estratégica

Os projetos são reconhecidamente os meios mais largamente utilizados para viabilizar as estratégias das empresas, com objetivo de garantir a sua competitividade (YELIN, 1999). Devido a isso, para que uma organização possa obter vantagens competitivas, os seus projetos devem estar alinhados às suas estratégias (PORTER, 2003).

Tendo em vista que os recursos (financeiros, humanos, tecnológicos, etc.) nas empresas são limitados, é necessário otimizar sua utilização, de forma a serem alocados nos projetos que mais agregam valor ao negócio.

A Gestão Estratégica (KAPLAN, NORTON, 1997) da organização (Figura 1) requer inicialmente a definição da missão, visão, crenças e valores da empresa, a partir das quais é elaborado um Planejamento Estratégico sobre a sua atuação, que resulta em Metas e Estratégias a serem alcançadas. Na visão do PMI (*Project Management Institute* – EUA), um portifólio é um conjunto de projetos ou programas e outros trabalhos agrupados para facilitar o gerenciamento eficaz desse trabalho, a fim de atender aos objetivos de negócios estratégicos (PMBOK, 2004).

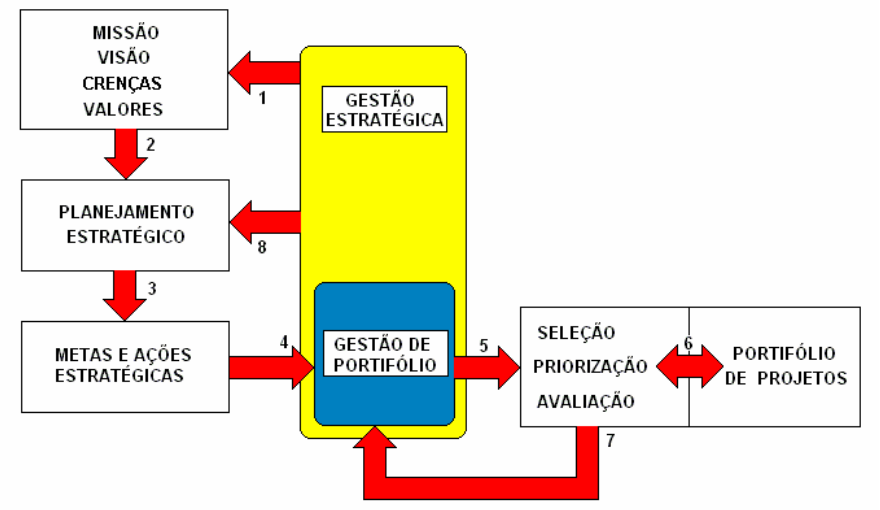


Figura 1 Relação entre Gestão de Portifólio e Gestão Estratégica

A Gestão de Portifólio de Projetos (COOPER, 2001) envolve um processo dinâmico de Avaliação, Seleção e Priorização de projetos, de forma que somente os mais relevantes e estratégicos à organização sejam desenvolvidos. O referido processo deve ocorrer em um ciclo contínuo, devido às constantes mudanças (sociais, políticas, econômicas, tecnológicas, etc.) que normalmente ocorrem no dia a dia e, nesse contexto, o tempo de resposta das empresas pode ser decisivo.

Como resultado da Gestão de Portifólio de Projetos, novos projetos podem ser criados, e pode haver aceleração, desaceleração ou mesmo descontinuidade de projetos existentes, face às novas oportunidades ou ameaças que possam surgir.

Os projetos definidos na Gestão de Portifólio (COOPER, 2001) em geral dão suporte a decisões em vários níveis gerenciais da organização, podendo inclusive influenciar na reformulação dos princípios relacionados à sua estratégia de atuação. Por esse motivo, a Gestão de Portifólio é considerada parte integrante da Gestão Estratégica de uma organização.

A gestão de Portifólio envolve inicialmente um ciclo de planejamento estratégico anual (PATTERSON, 1999), no qual as principais atividades a serem realizadas são a análise do direcionamento atual do negócio, a revisão dos mapas de projetos (*roadmaps*) das tecnologias e dos produtos disponíveis e necessários, a geração dos planos operacionais e do orçamento necessário para efetivá-los. Sendo assim, é função da gestão de portifólio garantir que os esforços de desenvolvimento sejam equilibrados para garantir a obtenção de três objetivos principais, a seguir:

- Identificar produtos específicos que atingirão as metas de lucro e receita;

- Mover a empresa rapidamente em direção aos seus objetivos estratégicos;

- Enfatizar a aplicação das competências essenciais e tecnologias disponíveis na empresa.

Adicionalmente, a gestão de portifólio envolve também o acompanhamento dos produtos durante todo o seu ciclo de vida, de modo que as informações da experiência do mercado com os produtos já desenvolvidos, possam contribuir para a definição das futuras estratégias de desenvolvimento de novos produtos e, conseqüentemente, no seu portifólio. Através da mesma, as empresas alocam recursos (LEVINE, 2005), recursos de forma apropriada com vistas a alcançar os objetivos corporativos de desenvolvimento de novos produtos.

Na prática, estes objetivos são desdobramentos diretos das estratégias tecnológicas e de mercado previamente estabelecidas pelos *stakeholders* (pessoas envolvidas nos projetos) da organização.

O principal objetivo da gestão de portifólio é balancear estrategicamente o conjunto de projetos de modo que se alinhem com as estratégias de tecnologia, mercado e negócios (MCGRATH et al, 1998). Ou seja, o estabelecimento de uma situação de equilíbrio ponderado de fatores, tais como prazos, custos e qualidade, em relação à disponibilidade de recursos e os objetivos estratégicos das organizações.

2.3 Seleção, Avaliação e Priorização de Projetos

Os estudos para a definição de modelos para seleção e priorização de projetos são antigos (Cooper, 1988). Ao longo dos últimos anos vários modelos têm sido propostos, dentre os quais:

- BARD et al (1988) utilizaram modelos e índices financeiros tais como o Valor Presente Líquido (VPL), o Retorno sobre o Investimento (ROI) e indicadores de “pay-back”, onde se buscava determinar quanto tempo a empresa demora a obter o valor financeiro investido.
- SOUDER e MANDAKOVIC (1986) propuseram a utilização de modelos financeiros e probabilísticos, os quais utilizavam as técnicas de Simulação de Monte Carlo e de15 Árvores de Decisão.

- HALL e NAUDIA (1990) desenvolveram modelos de pontuação baseados em perguntas qualitativas, relacionadas, basicamente aos riscos e benéficos relacionados aos projetos, as quais, tendo sido respondidas, permitiam calcular um “score” para priorizar os projetos.
- LILIEN e KOTLER (1983), defenderam a utilização de abordagens comportamentais - como por exemplo as baseadas no conhecido método *Delphi* - que provocam interações de opiniões dos gestores, até o ponto de haver consenso ou uma decisão de comum acordo sobre os produtos a serem desenvolvidos.
- Recentemente existem ainda os trabalhos que buscam relacionar o grau de alinhamento dos projetos às estratégias de negócio da organização, onde o mais famoso é o COBIT que, embora não seja propriamente um modelo para priorização, contém diretrizes para a definição de indicadores os quais podem fornecer subsídios para análise de performance dos projetos.

Contudo, na prática, não existe um “modelo perfeito” a ser adotado. Ou seja, o modelo ideal sempre será aquele que esteja mais adequado às necessidades da empresa, a qual pode variar, de acordo com a dinâmica das mudanças. E quanto mais complexo um modelo se tornar, mais difícil será o seu entendimento e, consequentemente, maior será o risco de resultados indevidos. Para resolver esse impasse, o ideal será a utilização de modelos mais simples, e de fácil configuração de parâmetros, cujos fatores de ajustes possam ser determinados interativamente. Vale frisar que mesmo estes, constituirão apenas uma diretriz, pois na maioria dos casos a decisão final pode envolver fatores externos de difícil contemplação em um modelo.

Um exemplo típico a ser usado para a priorização de projetos deve levar em conta itens considerados de relevância na organização onde os mesmos estão inseridos, podendo estes serem associados a pesos, definidos de acordo com a relevância de cada item para o cálculo final da nota que cada projeto deverá obter. Sendo assim, para o referido modelo, poderiam ser definidos critérios tais como: relevância do projeto para o usuário; relevância para a organização como um todo; relevância tecnológica; dificuldade de

implementação; prazo; custo; risco tecnológico; retorno de investimento; etc. Para cada um destes itens poderiam ser atribuídas pontuações de 0 a 5, por exemplo e, dependendo dos pesos estabelecidos para os mesmos, seria calculada uma pontuação final para cada projeto, a qual definiria a sua ordem de prioridade no portifólio.

O modelo anteriormente sugerido consta como uma das funcionalidades a serem contempladas pela ferramenta PROPOST, tendo sido elaborado especificamente para a mesma.

2.4 Relação entre Gestão de Portifólio, Gestão de Projetos e Gestão de Produtos

Um projeto é motivadamente constituído para conceber um produto final, seja este um bem material, um serviço a ser prestado ou qualquer outro resultado. Desde o seu início até a sua concepção final, um projeto passa por diversas fases, as quais definem o seu ciclo de vida. A Figura 2 ilustra as fases típicas do ciclo de vida de um projeto em geral. Nesta figura constam as seguintes fases: inicial, cujos produtos resultantes são o termo de abertura e a declaração de escopo; intermediária, cujos produtos resultantes são o plano do projeto, versões intermediárias do produto, relatórios de progresso e termos de aceitação e aprovação; e por último, a entrega do produto na sua versão final.

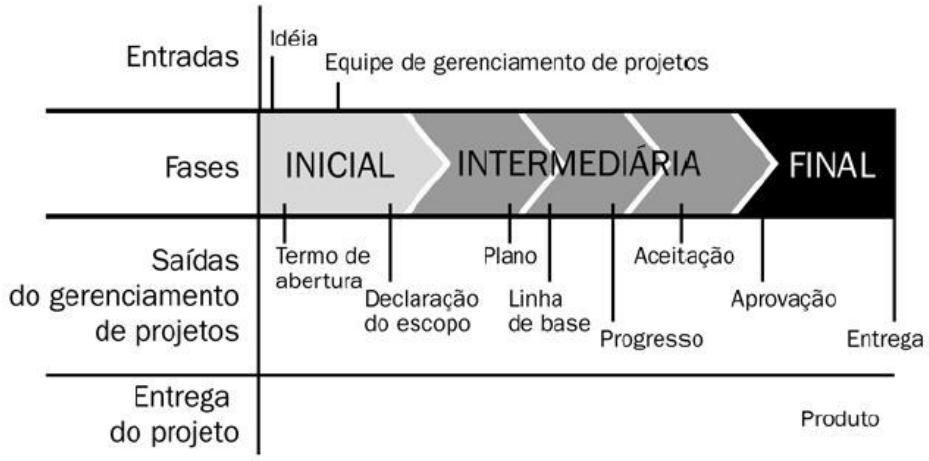


Figura 2 Fases típicas no ciclo de vida de um projeto (PMBOK, 2004)

No caso do desenvolvimento de software, por exemplo, essas fases são normalmente definidas por metodologias de Engenharia de Software, as quais determinam uma seqüência de passos a serem seguidos e artefatos a serem produzidos ao longo dos mesmos, como, por exemplo, o ciclo de vida mostrado na Figura 3, que é referente ao processo RUP (*Rational Unified Process*). Nesse caso, as fases do projeto são: iniciação, elaboração, construção e transição. Cada uma dessas fases pode ser dividida em uma ou mais interações, dependendo do tamanho e complexidade do projeto. As atividades realizadas nessas fases são referentes a: modelagem de negócios, requisitos, análise e design, implementação, teste e implantação, cada uma das mesmas podendo se estender ao longo das demais fases, conforme mostrado no cronograma.

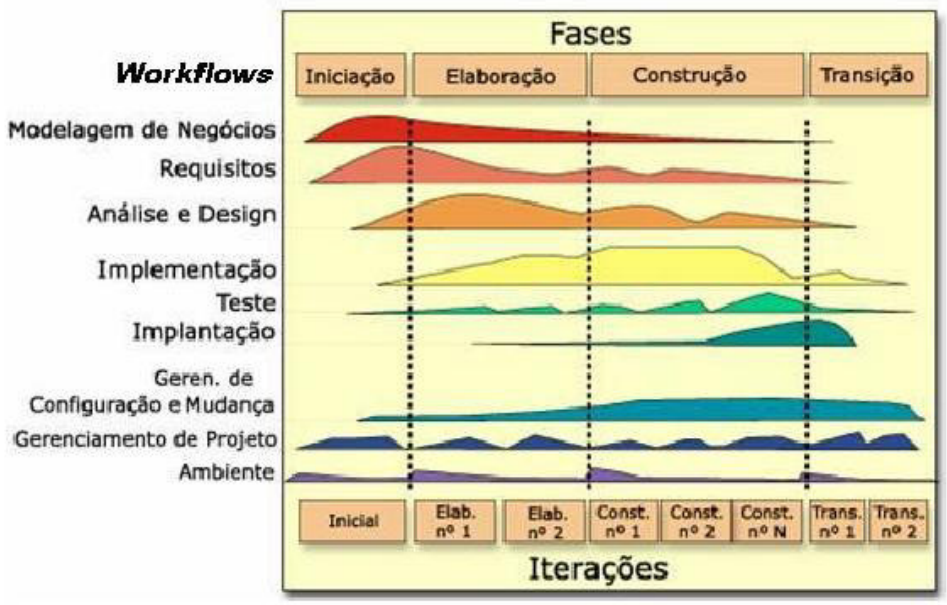


Figura 3 Fases do ciclo de vida de um projeto baseado no RUP (PRESSMAN, 2001)

Dessa forma, ao final do desenvolvimento, o produto resultante será o sistema de informação, o qual deve ser concebido dentro das especificações dos usuários. E a partir daí inicia-se o ciclo de vida do produto, o qual estende-se durante todo o tempo de vida útil do produto.

Durante a sua vida útil, um produto pode passar por diversas necessidades de modificações, sejam elas adaptações, atualizações, correções, melhorias, etc. O gerenciamento do produto contempla todas as ações gerenciais necessárias para que o produto permaneça atendendo às necessidades do cliente.

Adicionalmente, é essa área que fornece subsídios para a área de novos projetos, sejam estes relacionados a melhorias no produto já existente ou mesmo para a sua substituição, quando não for mais possível ou viável a sua modificação para atender aos objetivos para os quais o mesmo foi planejado. Logo, gerência de projetos e gerência de produtos são áreas complementares e, dependendo do caso, podem ter uma retroalimentação mútua. Isso pode ser visto claramente através da comparação do ciclo de vida de projetos e produtos, conforme mostrado na Figura 4. Em algumas organizações, o ciclo de vida do projeto é considerado parte do ciclo de vida do produto.

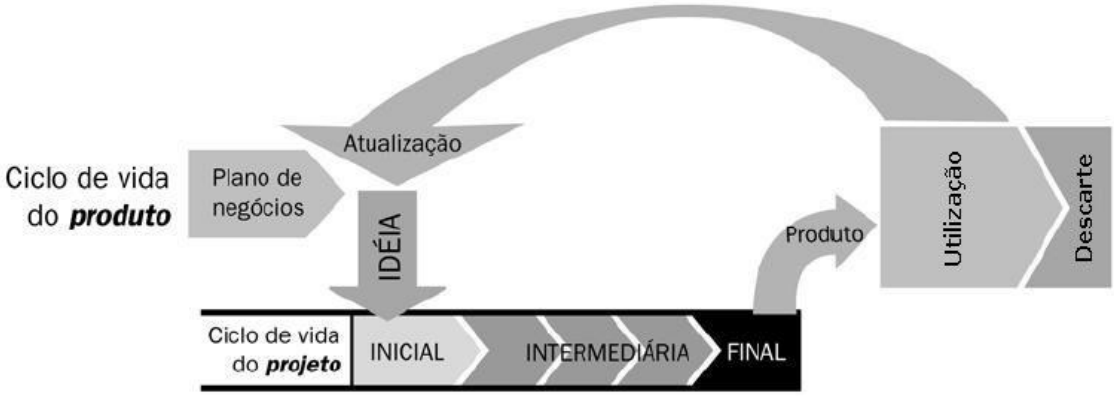


Figura 4 Relação entre ciclos de vida do projeto e do produto (PMBOK, 2004)

Segundo o PMI - *Project Management Institute*, o gerenciamento de projetos (PMBOK, 2004) consiste na aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto, a fim de atender aos seus requisitos. O PMBOK - *Project Management Body of Knowledge* (PMBOK, 2004) apresenta as práticas de gerenciamento de projetos divididas pelas seguintes áreas de conhecimento: escopo, prazo, custo, recursos humanos, comunicação, qualidade, contratação, riscos e integração.

Nesta visão, os processos ocorrem dentro de cinco grupos básicos (iniciação, planejamento, execução, controle e finalização) e podem se sobrepor ou interagir entre si conforme a fase do projeto. Dessa forma, a gerência de projetos encarrega-se adicionalmente de fornecer subsídios para que a gerência do produto possa iniciar a gestão do ciclo de vida do mesmo.

Um Portifólio pode fazer referência tanto a produtos como aos projetos que vão conceber os produtos. E, conforme citado anteriormente, é de fundamental importância para uma empresa ter total controle do seu portifólio de produtos existentes, para que possa gerenciar o seu portifólio de projetos.

Esse controle pode proporcionar, dentre outros, o aproveitamento de todo o conhecimento resultante no desenvolvimento dos produtos, o qual certamente proporcionará ganho de produtividade no desenvolvimento dos seus projetos. Logo, pode-se concluir que a Gestão de Portifólio de Projetos está intimamente ligada à Gestão de Portifólio de Produtos, uma vez que esta pode lhe fornecer subsídios necessários à otimização no desenvolvimento dos seus projetos.

Os termos Gerência de Portifólio de Projetos - GPP e Gerência de Múltiplos Projetos – GMP, são complementares (DYE,PENNYPACKER, 2003), porém, no contexto da gestão de projetos em geral as mesmas possuem uma conotação diferente. A primeira diferença está no nível corporativo em que ambas estão focadas; a GPP está focada em um nível estratégico, no qual os responsáveis pelas decisões são os executivos da alta direção da empresa; por outro lado, GMP está focada em um nível tático, onde o nível de tomada de decisão está no âmbito dos gerentes de projeto. Em relação ao objetivo principal, a GPP preocupa-se mais com a seleção e priorização dos projetos, considerando o planejamento para um horizonte de médio e longo- prazo, enquanto que a GMP está mais centrada com a alocação dos recursos, visando o atendimento em curto prazo e o refinamento do planejamento em um nível mais alto na GPP .

O atual cenário dos negócios no mundo globalizado é complexo e requer rápidas decisões, melhor alocação dos recursos escassos e uma clara definição de foco. Uma organização consiste de um *mix* constantemente alterado de grandes e pequenos projetos (KENDALL, 2004). Nesse contexto, a Gestão de Portifólio de Projetos possui os desafios de alocação de recursos, priorização e monitoramento de projetos, com vista às aderências ao tempo, escopo e custos dos requisitos em projetos individuais, sempre tendo como foco principal o interesse corporativo.

2.5 Ferramentas líderes no gerenciamento de portifólio

Atualmente existem muitas ferramentas que se propõem ao gerenciamento de portifólio de projetos. A seguir é apresentado um estudo do estado da arte realizado nas principais ferramentas líderes de mercado, o qual contempla uma breve descrição sobre as principais funcionalidades das mesmas. As ferramentas estudadas foram: Primavera Enterprise, Planview Enterprise, CA Portifolio Manager e Microsoft Project Enterprise.

2.5.1 Primavera Enterprise

O Primavera Enterprise (http://www.primavera.com) é o produto de gerenciamento de projetos mais utilizado em todo o mundo, muito embora a sua utilização no Brasil ainda seja reduzida, provavelmente devido ao seu custo. Sua suíte é composta de três módulos principais: Project Planner, Portifólio Analyst e Primavision.

- **Project Planner:** é uma ferramenta usada no planejamento de múltiplos projetos com acesso simultâneo por gerentes, líderes e demais participantes, na qual estão contemplados: análise de risco, alertas, gerência de documentação e controle de recursos.
- **Primavision:** é usado para análise de performance segundo métricas como “valor ganho” e “roi” (retorno do investimento), sendo totalmente *Web*, podendo sua utilização ser estendida para a criação de ordens de serviço (“work orders”), alocação e visualização de recursos na *Web*.

- **Portfolio Analyst:** é uma ferramenta de visualização dos dados consolidados, dirigida à rápida tomada de decisão, e possui seu foco na análise de Projetos, voltada para diretores e gerentes. Sintetiza os dados dos projetos com vistas sumarizadas, planilhas e gráficos, que facilitam a comparação e análise de vários projetos, a qualquer nível de detalhe alinhado pela WBS (Estrutura Analítica do Projeto).

Através do Portfolio Analyst, pode-se averiguar o status do projeto e realizar análise de tendências. O usuário pode acessar uma série de opções de índices de performance dos projetos, tecendo comparações entre projetos, análises do orçado e a tendência do realizado, realizado até a última verificação de cálculo do projeto, comparações entre previsto e realizado de horas trabalhadas, custos, despesas e diversos relatórios ilustrativos por vários níveis de projetos.

2.5.2 Microsoft Project Server

O Microsoft Project (http://www.microsoft.com/brasil/office/project), assim como o Primavera, é uma das ferramentas mais utilizadas para gerenciamento de projetos nas empresas na atualidade. Devido a possuir um custo mais acessível que o Primavera, e também por fazer parte dos produtos da Microsoft, tendo grande facilidade de integração com os demais produtos desse fornecedor, o mesmo é o software de gerenciamento de projetos mais utilzado no Brasil. Diferente das demais ferramentas apresentadas anteriormente, o MSProject é um software *desktop* que oferece recursos tanto para o processo de seleção e priorização de projetos quanto ao processo de controle dos mesmos.

A versão Server do Microsoft Project permite um repositório central dos projetos, acessível a todos os membros dos projetos e com possibilidade de relatórios web. As ferramentas de gerenciamento de portifólio baseadas na *Web* ajudam os executivos a alinhar as pessoas e os projetos com as metas nos negócios, a identificar rapidamente os problemas e a tomar medidas corretivas.

Com a utilização do MSProject, os membros da equipe atualizam facilmente as informações do projeto, colaboram e permanecem informados através de correio eletrônico e ferramentas baseadas na *Web*. E os gerentes de projeto continuam usando os recursos conhecidos e abrangentes do Project Professional em gerenciamento de projetos. O MSProject possui uma arquitetura expansível que permite às empresas integrar esta solução aos sistemas existentes da linha de negócios, no sentido de obter uma solução mais completa.

2.5.3 CA - Clarity Portfolio Manager

A CA (http://www.niku.com/) é uma ferramenta bastante completa, cujo fornecedor é a Computer Associates. Esta ferramenta até o início de 2005 chamava-se Niku, e atualmente é conhecida simplesmente como CA ou Clarity. É pouca conhecida no Brasil, apesar de muito usada em países onde a gestão de portifólio está mais evoluída. Dentre as principais características apresentadas como ponto de destaque dessa ferramenta, constam as seguintes:

- Ambiente fortemente integrado que facilita o acesso e navegação a qualquer portifólio ou projeto existentes.
- Flexibilidade e precisão nas medidas de investimentos e análises de investimento.
- Identificação de melhores alternativas com múltiplos cenários de simulação.
- Situação atual dos investimentos on-line, para facilitar tomadas rápidas de decisão.
- Otimização e possibilidade de criação de cenários para simulação de portifólio com suporte de vários recursos gráficos.

A Clarity possui uma *suit* de produtos que, na prática são módulos funcionais relacionados ao suporte às várias atividades da gestão de projetos e portifólios. São eles:

- **Portfólio Manager**: módulo responsável pelo gerenciamento integrado dos projetos componentes dos portifólios, oferecendo funcionalidades de otimização, simulação, controle e gerenciamento, com foco em uma visão corporativa em relação às dependências, impactos de custo e prazo, priorizações, etc.
- **Resource Manager**: é um módulo especificamente focado no gerenciamento dos recursos existentes na organização, bem como na otimização das suas alocações aos projetos, em observância às necessidades identificadas para o portifólio.
- **Project Manager:** neste módulo as principais funcionalidades relacionadas à gestão de projetos estão concentradas (criação de atividades, dependências, apropriações de prazo, custo, acompanhamentos via gráfico de Gantt, etc.)
- **Demand Manager:** este módulo é o responsável pelo gerenciamento das demandas relacionadas aos projetos, dando amplos recursos de visualização do status da sua execução e utilização dos recursos alocados para as mesmas.
- **Process Manager:** módulo responsável pelo gerenciamento dos processos, o qual, dentre outros, oferece recursos de acompanhamento de fluxo (*workflow*) e visualização gráfica para controle e rastreabilidades de atividades, documentos, recursos, apropriações, etc. Possui ainda funcionalidades para automatização de processos e remoção de gargalos nos mesmos.
- **Clarity Studio:** este módulo oferece funcionalidades para construção de interfaces personalizadas para a apresentação de resultados.

2.5.4 Planview Enterprise

O Planview (http://www.planview.com) é considerado, juntamente com o Primavera, os melhores e mais consistentes produtos para análise de portifólio na atualidade. Contudo, igualmente não possui boa penetração no mercado brasileiro, sendo praticamente desconhecida, muito embora seja bastante conhecida em outros países. É considerado o produto de gestão de projetos e portifólio de projetos que possui os recursos mais amigáveis em termos de aprendizado e navegação.

O Planview Enterprise possui, dente outras, funcionalidades para a realização dos processos através de gerações automatizadas ou através de guias passo a passo, com o emprego das melhores práticas, no sentido de oferecer recursos de maturidade e maximizar o retorno dos investimentos, tendo como consequência melhorias na performance dos projetos.

Este produto possui três componentes principais que atuam nas áreas estratégicas de projetos e serviços na organização, fornecendo informações on-line para proporcionar melhor visibilidades aos executivos e gerentes, no sentido de tomada de decisões. São eles:

- **PlanView Enterprise Portfolio Management**: é o módulo responsável pela criação de uma visibilidade ampla para toda a organização relacionada à análise e gerenciamento de riscos de negócio, priorização de estratégias, planejamento de cenários, priorização de investimentos baseado em critérios chaves, alinhamento dos portifólio ao negócio da empresa, análise de performance, etc.
- **PlanView Project Portfolio Management:** este módulo possui funcionalidades relacionadas a estimativas efetivas de custo, prazo e tendências, bem como planejamento de todo o trabalho a ser realizado nos projetos do portifólio. Dentre outros, o mesmo possui funcionalidades relacionadas a colaboração nas atividades, identificação e mitigação de riscos nos projetos, gerenciamento de ciclos de vida desde o início até a conclusão, quadros sinalizadores, geração de cenários, análise e visualização gráfica de desempenho e execução, histogramas de alocações de recursos, etc.

- **PlanView Service Portfolio Management:** este módulo é o responsável por coletar e gerenciar o custo total de mão de obra, software, hardware, infra-estrutura e *outsourcing* envolvidos nas atividades do projeto. Proporciona análise financeira, topologia de serviços, inventário de recursos, reconstrução de falhas, controle do ciclo de vida dos serviços, etc.

O Planview Enterprise possui também um repositório central para armazenamento das informações que possibilita facilidade na integração das informações, balanceamento na distribuição de atividades relacionadas à otimização de recursos, geração de cenários de simulação, etc.

Outra característica deste produto está relacionada à facilidade na realização dos processos através de gerações automatizadas ou através de guias passo a passo, com o emprego das melhores práticas, no sentido de oferecer recursos de maturidade e maximizar o retorno dos investimentos, tendo como consequência melhorias na performance dos projetos. Oferece também recursos que podem ser utilizados pelo escritório central de gerenciamento de projetos para acompanhamento, controle e análise do portifólio, bem como elaboração de vários tipos de relatórios a partir de templates pré-existentes.

2.5.5 Análise da performance das ferramentas de gestão de portifolio

Para melhor nos embasarmos em relação à eficiência das funcionalidades contidas nas ferramentas estudadas, consultamos várias pesquisas realizadas por instituições de renome em relação à performance das mesmas. A seguir apresentamos a pesquisa realizada pela *Forrester Research, Inc* (*Forrester*, 2006), que é uma conceituada instituição na área de estudos e pesquisas na área tecnológica. A referida pesquisa foi também a mais recente publicada por uma instituição de renome, na época em que elaboramos este trabalho. A mesma analisou os vários produtos considerados eficientes e de grande penetrabilidade no mercado da gestão de portifólio de projetos na atualidade, dentre os quais estão inclusas as ferramentas por nós pesquisadas: Primaerva Enterprise, Planview Enterprise, CA Portifolio Manager e Microsoft Project Enterprise.

A pesquisa consistiu em avaliação dos produtos por vários especialistas na área, levando-se em conta critérios variados (técnicos, estratégias de marketing, penetrabilidade de mercado, etc), normalmente demandados para ferramentas de tal porte.

Para efeito deste trabalho, serão apresentados apenas os resultados relacionados aos critérios técnicos, os quais basearam-se na existência, completeza e qualidade das funcionalidades oferecidas, na sua facilidade de uso e entendimento, bem como na real eficiência do seu funcionamento em relação à objetividade e apresentação dos resultados para o usuário. Dessa forma, foram levados em conta os itens para a referida análise, cujo resultado é mostrado na Tabela 1:

- **Gerenciamento de Demandas:** relacionado à facilidade e eficiência na criação de novas atividades e tarefas em geral.
- **Gerenciamento do Portifólio:** relacionado à facilidade e eficiência na criação de novos portifólios, bem como às funcionalidades existentes para análise, simulação e otimização.
- **Gerenciamento de Projetos**: relacionado à facilidade e eficiência na criação de novos projetos, bem como suas dependências e funcionalidades para acompanhamento de performance e evolução dos projetos.
- **Gerenciamento de Recursos:** relacionado à facilidade e eficiência na criação de previsões, estimativas e rastreabilidades nas alocações de recursos aos projetos.
- **Gerenciamento Financeiro:** relacionado à facilidade e eficiência na criação e gerenciamento de orçamentos e análises financeiras.
- **Metodologia:** relacionado à facilidade e eficiência na criação e configuração de fluxos de processos e templates relacionados à metodologias a serem usadas nos projetos.
- **Controle de Fluxo (*workflow*):** relacionado ao controle de fluxo e visualização gráfica, oferecidos para os processos pertinentes ao gerenciamento do portifólio.

- **Relatórios**: relacionado à facilidade e eficiência na criação de relatórios e impressão.
- **Gerenciamento Integrado de Informação:** relacionado à facilidade e eficiência com que a ferramenta proporciona a rápida visualização de eventos, a identificação de impactos e tomadas de decisões ou ações corretivas.
- **Tecnologias da aplicação:** relacionada a quais tecnologias e plataformas de hardware são necessárias para a instalação e funcionamento da ferramenta.

As pontuações resultantes na Tabela 1 foram obtidas a partir de notas atribuídas por profissionais reconhecidos como especialistas na área de Gestão de Portifólio de Projetos, e com conhecimento das relativas ferramentas. Para cada critério anteriormente citado foram atribuídas pontuações, as quais foram ponderadas pelo correspondente peso percentual, resultando na composição final abaixo apresentada.

	Forrester's Weighting	CA	Microsoft	PlanView	Primavera
CURRENT OFFERING	50%	4.12	2.17	4.24	4.19
Demand management	5%	4.00	1.90	5.00	4.80
Portfolio management	20%	4.20	1.35	4.65	4.85
Project management	10%	4.35	3.50	4.20	3.50
Resource management	10%	4.70	3.80	4.70	4.80
Financial management	10%	4.05	1.70	3.95	4.70
Methodology	15%	4.70	2.70	5.00	4.70
Workflow	10%	4.25	1.40	4.70	4.10
Reporting	5%	4.75	3.65	4.05	4.35
Integrated IT management	10%	2.30	0.60	2.20	2.00
Application technology	5%	3.50	2.40	2.60	3.00

Tabela 1 Análise de Requisitos Técnicos em produtos de Gestão de Portifólio (*Forrerest Research*, 2006)

2.6 Principais funcionalidades da Gestão de Portifólio

Com base no estudo realizado sobre as ferramentas líderes do mercado da gestão de portifólio, pode-se constatar que, de uma maneira geral, existem muitas funcionalidades comuns entre as mesmas. Dentre estas funcionalidades, constam as seguintes:

- **Funcionalidades relacionadas à Modelagem**

Estas funcionalidades estão relacionadas à forma como os projetos são estruturados dentro do portifólio, suas atividades, alocações de recursos, visualizações gráficas, metodologias, etc.

- **Funcionalidades relacionadas a Simulações**

As simulações permitem o cálculo de vários cenários baseados em parâmetros configurados manualmente pelos usuários. Por exemplo, o usuário pode querer saber quantos projetos ele pode executar com um determinado custo e, a partir destes, poderá interativamente aumentar o número de investimentos em software, hardware, pessoal, etc. e realizar outras simulações.

- **Funcionalidades relacionadas à Otimização**

Permitem a geração automática das melhores alternativas de projetos, de acordo com os recursos existentes, e com a otimização da utilização deste, proporcionando resultados dentro de um menor prazo e custo possíveis. Na prática, observa-se que, de uma maneira geral, essa funcionalidade ainda não é utilizada em toda a sua potencialidade, tendo em vista o grande número de parâmetros necessários para os devidos ajustes, que acabam por introduzir complexidade na sua execução. Dependendo das particularidades do portifólio em questão, uma pequena falha no ajuste de um parâmetro pode levar a um resultado fora da realidade.

- **Análise de Performance do Portifólio**

Essa funcionalidade fornece o status dos projetos que compõem o portifólio atual, no sentido de informar quais estão sendo desenvolvidos dentro das expectativas (principalmente de custo e prazo) e quais possuem desvios. Nessa funcionalidade também são oferecidos recursos gráficos, tais como: gráfico de pizza, histogramas, 3d, Gantt, etc.

2.7 Iniciativas acadêmicas no âmbito nacional

Tendo em vista a Gestão de Portifólio de Projetos ainda ser recente mesmo no panorama internacional e, conseqüentemente, pouco difundida no Brasil, as iniciativas brasileiras no âmbito acadêmico ainda são poucas em relação ao desenvolvimento de ferramentas para suporte a essa área. Nesse contexto, na época da elaboração deste trabalho, as ferramentas em desenvolvimento mais diretamente relacionadas ao assunto foram as seguintes:

2.7.1 ODE – Um ambiente para Gestão de Projetos de Software

O ODE - *Ontology-based software Development Environment* (PEZZIN e FALBO, 2004) (MORO, NARDI e FALBO, 2005), é um ambiente baseado no conhecimento que vem sendo desenvolvido no Laboratório de Engenharia de Software – LabES – do Departamento de Informática da Universidade Federal do Espírito Santo (UFES). O mesmo está fundamentado em várias ontologias, as quais permitiram a integração de diversas ferramentas para a composição de uma infra-estrutura que permite controlar projetos de software e seus respectivos processos. Como exemplo destas ferramentas, consta o ControlPro (MORO, NARDI e FALBO, 2005), cujo principal objetivo visa o apoio ao acompanhamento de projetos de software de ODE permitindo alterações dinâmicas em seu processo, incluindo alterações de atividades e de seus ativos (artefatos, procedimentos e recursos).

O ambiente ODE oferece aos usuários serviços gerais e específicos. Os serviços gerais podem ser utilizados a qualquer momento pelos usuários, podendo ser chamados tanto no próprio menu geral, como a partir de qualquer uma das suas ferramentas. São elas: Coleta de Conhecimento e Aprovação de Lições Aprendidas; Recuperação e Acesso a Itens de Conhecimento Através da Busca; Caracterização de uso de itens de Conhecimento; e Manutenção dos Itens de Conhecimento. Os serviços específicos são aqueles providos a partir das ferramentas específicas que podem ser integradas ao ambiente. Dentre estas funcionalidades, destacam-se: Controle de Projetos, Cadastro de Conhecimento, Cadastro e Alocação de Recursos, Definição de Processos, Acompanhamento de Projetos, Agenda e Registro de Esforços, Gerência do Conhecimento.

2.7.2 GMP: Uma Ferramenta para a Gestão de Múltiplos Projetos

O GMP (FREITAS e MOURA, 2004), é um sistema de gerenciamento de projetos, em desenvolvimento no Centro de Informática – Universidade Federal de Pernambuco (UFPE). O mesmo está voltado para ambientes corporativos nos quais há a execução de vários projetos simultaneamente, compartilhando recursos escassos como tempo, pessoas e investimentos. Através de uma interface gráfica Web simples de usar, os gerentes de projeto podem acompanhar em tempo real o andamento de todos os projetos sob sua responsabilidade de maneira mais precisa e eficiente.

Essa ferramenta leva em conta aspectos relacionados à tomada de decisão, tais como o controle efetivo do orçamento e da alocação de pessoas; e métricas para estimativa do percentual de conclusão dos projetos, dentre outros etc. Além da facilidade de implantação, utilização e acesso, característicos dos sistemas Web, o GMP tem como objetivo melhorar a alocação de recursos entre as atividades dos diversos projetos e controlá-los simultaneamente de modo a obter a máxima eficiência na condução dos mesmos.

O GMP é especificamente voltado para projetos de desenvolvimento de software, para os quais leva em consideração detalhes relevantes referentes aos requisitos funcionais e não funcionais e casos de uso. A evolução dos casos de uso, por exemplo, é estimada segundo fórmulas específicas que indicam o percentual de conclusão de cada caso e, conseqüentemente, de cada requisito funcional do sistema a ser desenvolvido. Dentre as principais funcionalidades oferecidas, constam: controle de custos e prazos, progresso funcional, controle de usuários, recursos e permissões de acesso, fórum de discussões, base de lições aprendidas, notificações via e-mail, etc.

2.8 Considerações Finais do Capítulo

A Gestão de Portifólio é um modelo gerencial em ascensão na atualidade, no meio empresarial. Contudo, tendo em vista ser este um modelo ainda recente, muitos de seus conceitos ainda são desconhecidos e/ou confundidos por muitos profissionais e, dessa forma, muitas vezes são utilizados de forma inadequada. Sendo assim, este capítulo teve inicialmente o objetivo de fornecer os principais conceitos relacionados ao domínio da Gestão de Portifólio de Projetos. Ao longo do mesmo, quando necessário, foram abordados determinados conceitos de forma comparativa em relação a outros conceitos similares normalmente utilizados no âmbito da gestão de projetos, no sentido de esclarecer o seu real significado. Nesse contexto, vale ratificar que a Gestão de Portifólio de Projetos preocupa-se em fornecer uma visão global do portifólio existente, na qual todas as mudanças que porventura possam ocorrer em um determinado projeto, por exemplo, são tratadas de uma forma corporativa, sempre visando os impactos resultantes em todos os demais projetos do portifólio.

Outro objetivo deste capítulo foi um estudo no estado da arte das ferramentas que suportam a Gestão de Portifólio, no sentido de identificar necessidades relacionadas a funcionalidades, nas quais a ferramenta proposta neste trabalho poderia estar dando alguma contribuição. Ou seja, este trabalho não tem como objetivo definir uma ferramenta para “competir” com as “grandes líderes de mercado”, mas sim para agregar valor a necessidades relacionadas ao tema em estudo, cujas funcionalidades que lhes dão suporte ainda podem ser melhoradas através de tecnologias de vanguarda, conforme demonstraremos mais adiante.

Dentre as ferramentas líderes de mercado, por nós estudadas e, de acordo com os resultados obtidos pela pesquisa apresentada, pode-se observar que os produtos Primavera Enterprise e Planview Enterprise são os que oferecem funcionalidades consideradas mais efetivas em relação à gestão de portifólio de projetos, seguidos da CA. Contudo, tais ferramentas ainda são pouco conhecidas no Brasil, apesar de bastante utilizadas em outros países onde a gestão de portifólio encontra-se em um estágio mais avançado. Por outro lado, nota-se um resultado interessante relacionado ao Microsoft Project Enterprise, que apesar de baixa classificação nas pesquisas é um produto líder no Brasil. Dentre os motivos aos quais atribuímos essa liderança constam: seu baixo custo em relação às demais ferramentas; a popularidade que os produtos da Microsoft possuem; e o fato de que no Brasil a gestão de Portifólio de Projetos ainda é uma área nova e pouco explorada. Este último fator também contribui para que no âmbito acadêmico nacional, ainda haja poucas iniciativas do desenvolvimento de ferramentas na referida área.

Nenhuma das ferramentas pesquisadas possui as funcionalidades de recomendações baseadas em técnicas de *Filtragem da Informação* e *Mineração de uso*, e nem acesso à informação baseado em técnicas de *Recuperação da Informação,* com o objetivo de prevenir o problema da duplicidade de definição de projetos. Devido a isso, consideramos a ferramenta proposta nesse trabalho uma inovação em relação às funcionalidades concernentes a essas particularidades, oferecidas pela mesma.

3 TECNOLOGIAS USADAS NO DESENVOLVIMENTO DA PROPOST

3.1 Recuperação e Filtragem da Informação

Dentre as principais funcionalidades existentes na ferramenta PROPOST, destacam-se a pesquisa de soluções de software, que utiliza técnicas relacionadas à Recuperação da Informação, e a recomendação de soluções de software, que utiliza técnicas relacionadas à Filtragem Colaborativa e Mineração de uso. A seguir são apresentados os principais conceitos relacionados a estas tecnologias.

3.1.1 Recuperação de Informação – RI

A Recuperação de Informação (BAEZA, RIBEIRO NETO,2000) lida com representação, armazenamento, organização e acesso a itens de informação (documentos), sendo este um processo apropriado para atender às necessidades de informação em curto prazo, em fontes de informação relativamente estáveis e não-estruturadas, como a *Web*. A representação e a organização da informação devem proporcionar ao usuário de um Sistema de Recuperação de Informação (SRI) um acesso fácil à informação de seu interesse.

Na última década, a área de RI tem atingido um alcance muito maior do que seus objetivos iniciais, que eram indexar texto e buscar documentos em uma coleção. O fato que impulsionou as pesquisas na área foi o crescimento da *Web*, a partir dos anos 90. A crescente complexidade dos objetos armazenados e o grande volume de dados exigem processos de recuperação cada vez mais sofisticados (BAEZA, RIBEIRO NETO,2000).

O aumento contínuo de documentos disponíveis na *Web* tem tornado. A tarefa de encontrar a informação desejada cada vez mais difícil. Muitas ferramentas de busca e técnicas de indexação têm sido desenvolvidas para resolver esse problema (LAWRENCE, 2000).

3.1.1.1 O processo de Recuperação de Informações

O processo de acessar documentos em um SRI (Sistema de Recuperação de Informações) começa quando o usuário se depara com uma necessidade de informação. Em contato com a interface, o usuário traduz sua necessidade de informação em uma consulta. A tarefa do sistema é, através do processamento da consulta, encontrar os documentos que possam lhe interessar. Ou seja, o SRI (BAEZA, RIBEIRO NETO,2000) deve de alguma forma “interpretar” o conteúdo das informações encontradas nos documentos de uma coleção e ordená-los de acordo com um grau de relevância para o usuário.

O processo de recuperação de informação em um SRI em geral pode ser dividido em duas etapas principais: a primeira trata da constituição da coleção de documentos e da geração de um índice para a mesma; e a segunda trata da utilização do sistema por seus usuários.

Uma coleção de documentos (BAUER, 2001) trata de assuntos de interesse da comunidade de usuários. Dada uma coleção, é necessário que a mesma seja organizada de forma a facilitar o acesso aos documentos. Para tanto, um instrumento central em uma biblioteca, tradicional ou digital, é o índice. Devido a características e preferências pessoais que surgem no processo de análise do conteúdo dos documentos, diferentes termos de indexação podem ser selecionados para representar os mesmos documentos.

Para evitar problemas advindos da variabilidade no uso da linguagem, ferramentas como os vocabulários controlados, dentre eles os *thesaurus* (SILVEIRA et al, 2003), foram criadas ao longo dos anos, os quais descrevem os conceitos dentro de um determinado assunto e os relacionamentos entre os mesmos.

Durante a fase de indexação, essas ferramentas funcionam como uma espécie de vocabulário autorizado para orientar o indexador na seleção dos termos de indexação. Na fase de pesquisa, essas ferramentas podem ser utilizadas para orientar o usuário na formulação e contextualização da consulta.

3.1.1.2 Modelo Genérico da Recuperação de Informação

Na Figura 5 é apresentado um modelo genérico de um Sistema de Recuperação de Informação (BELKIN, CROFT, 1992). No lado direito do modelo o foco está no usuário, o qual possui uma necessidade de informação pontual ao acessar o sistema. O sistema, por sua vez, lhe proporciona uma funcionalidade para representação da sua necessidade em uma *query* (ou consulta), a qual consiste em uma linguagem que possa ser interpretada pelo sistema.

No lado esquerdo da Figura 5, o atores iniciais são os produtores de texto (elementos de informação), que dão origem às coleções de texto. Essas coleções são representadas internamente em uma forma mais adequada para ser processada pelo computador, ou seja, a indexação dos mesmos. Nesse processo, são considerados o agrupamento dos textos em coleções (por exemplo, bancos de dados), a representação interna dos textos, e a organização dessas representações em bancos de dados e *surrogates* de textos.

A última fase do processo consiste na comparação entre a *query* (*representação da consulta*) e os *surrogates* de texto (representação dos *elementos de informação*), para que sejam processadas a seleção e recuperação de um conjunto de textos que possam, possivelmente, serem considerados relevantes. Os textos recuperados são avaliados pelo usuário e, caso não satisfaçam inteiramente a necessidade de informação, podem levar a uma modificação na *representação da consulta* pelo usuário, ou mesmo em um refinamento da sua necessidade de informação.

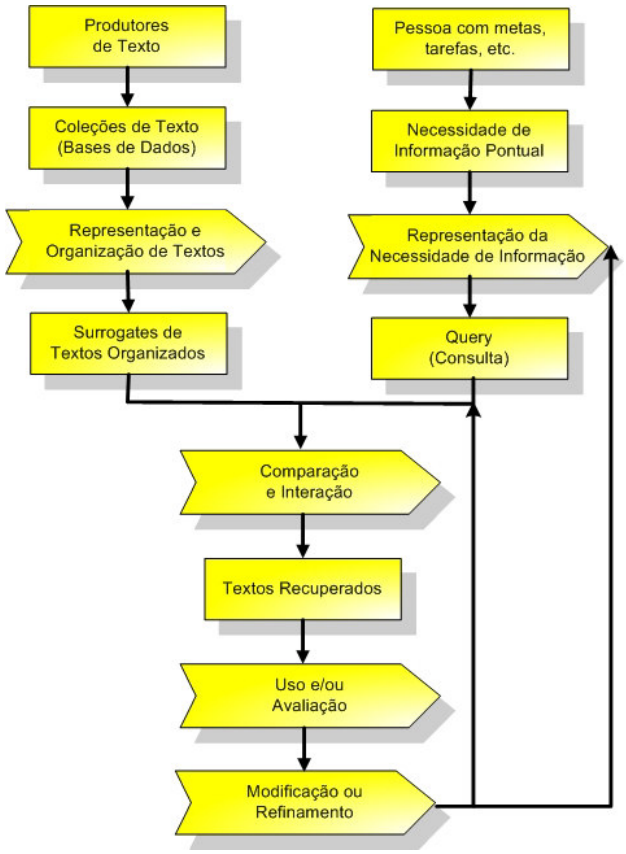


Figura 5 Modelo Genérico da Recuperação de Informação (BELKIN, CROFT, 1992)

3.1.2 Filtragem de Informação – FI

A Filtragem de Informação (BURKE, 2000) é a área de pesquisa que oferece ferramentas para separar informações relevantes e irrelevantes, e a sua entrega para pessoas que precisam dela. A FI (BELKIN, CROFT, 1992) objetiva atender às necessidades de longo prazo do usuário, as quais são expressas como perfis, que são utilizados no processo de busca. Os perfis podem ser adquiridos tanto de maneira explicita, através de informações diretamente fornecidas pelos usuários, bem como implicitamente, através da captura automática baseada nas observações do comportamento do usuário. Na definição da relevância das informações podem ser utilizadas tanto informações do usuário alvo, como também de outros usuários com hábitos similares ao mesmo, ou mesmo uma abordagem híbrida através da combinação de várias informações disponíveis.

As principais técnicas atualmente utilizadas no processo de filtragem da informação são: Filtragem Colaborativa (FC), Filtragem Baseada no Conteúdo (FBC) e Filtragem Híbrida (FH) (ADOMAVICUS, TUZHILIN, 2005).

- Filtragem Baseada em Conteúdo - FBC: baseia-se em informações obtidas através da análise do conteúdo dos itens de informação pelos quais o usuário demonstrou preferência no passado (ADOMAVICUS, TUZHILIN, 2005). Ou seja, para fazer a recomendação, os sistemas que utilizam essa abordagem inicialmente tentam "entender" padrões de preferências existentes nos itens previamente avaliados pelo próprio usuário e, a partir daí, fazem a extração de vários termos ou palavras-chave, que servem para a configuração de um perfil contendo as suas preferências. Para cada item de informação são configurados perfis de representação, contento os termos considerados mais importantes destes itens. A definição dos itens de maior relevância para os usuários é realizada através da análise da similaridade entre as representações internas dos itens de informação e a representação do perfil dos usuários, a partir da qual é montado um ranking.
- Filtragem Colaborativa - FC: baseia em avaliações realizadas pelo usuário alvo, mas sim nas avaliações deste item realizadas por outros usuários que possuem um padrão semelhante de avaliação ao usuário alvo (HERLOCKER et al, 2004). Ou seja, usuários que no passado avaliaram os mesmos itens que o usuário alvo de uma maneira similar, teoricamente teriam gostos semelhantes aos gostos que este. O funcionamento básico desta técnica, divide-se em três passos principais: definição da similaridade, seleção dos vizinhos mais próximos e predição (que é a recomendação propriamente dita).
- Filtragem Híbrida - FH: consiste em combinar diversos recursos utilizados na Filtragem Baseada no Conteúdo e na Filtragem Colaborativa (HERLOCKER et al, 2004), com o principal objetivo de suprir certas limitações existentes nessas duas abordagens. Existem inúmeras técnicas para realizar essa combinação, as quais basicamente fundamentam-se em três princípios: combinação de resultados em separado, introdução de métodos de FC em FBC, introdução de métodos de FBC em FC, e construção de modelos genéricos envolvendo as duas técnicas. É utilizada

para suprir certas limitações das abordagens baseadas no conteúdo e colaborativas.

3.1.2.1 Limitações das Técnicas de Filtragem

A FBC permite uma exploração mais personalizada de detalhes relativos ao usuário, tendo em vista que refere-se a informações exclusivas ao seu comportamento. Porém, os dois principais problemas associados com a mesma, e que devem ser inicialmente resolvidos estão relacionados à dificuldade na definição da representação dos documentos, e à definição de um perfil efetivamente representativo em relação às preferências do usuário (ADOMAVICUS, TUZHILIN, 2005). Outras limitações comumente observadas na utilização desta técnica são referentes a: limitação das análises a conteúdos apenas na forma de texto; não distinção de qualidade entre bons e maus textos; possibilidade de recomendar apenas itens semelhantes aos que o usuário já avaliou (*superespecialização*); usuários novos (sem avaliações) não recebem recomendações; etc.

A FC permite uma exploração mais ampla de itens ainda não avaliados pelo usuário, solucionando algumas limitações da FBC (ADOMAVICUS, TUZHILIN, 2005). Contudo, a mesma também possui algumas limitações, dentre as quais destacam-se: itens novos (sem avaliações) não podem ser recomendados; usuários novos não possuem vizinhos – conseqüentemente não recebem recomendações; só é possível recomendar itens semelhantes aos que o usuário já avaliou (*superespecialização*); o aumento do número de itens sem avaliações diminui a quantidade de vizinhos (*esparsividade*); bases de dados muito grandes demandam longo tempo de processamento (*escalabilidade*); etc.

3.1.2.2 Sistemas de Recomendação

Sistemas de Recomendação são tipos particulares de aplicações de filtragem de informação (ADOMAVICUS, TUZHILIN, 2005). Os mesmos vêm sendo desenvolvidos para mediar, apoiar e automatizar o processo cotidiano de compartilhar recomendações (TERVEEN, HILL,2001), e têm se mostrado uma ferramenta poderosa a ser utilizada em comércio eletrônico, bibliotecas digitais e gerência de conhecimento. Nos últimos anos, estes sistemas passaram a serem utilizados comercialmente em lojas na *Web*.

O uso crescente desses sistemas tem possibilitado o aumento das vendas, por ser uma abordagem que ajuda o consumidor a encontrar o produto desejado (SCHAFER et al., 2001). De acordo com Cazella e Reategui (2004) "em um sistema típico as pessoas fornecem recomendações como entradas e o sistema agrega e direciona para os indivíduos considerados potenciais interessados neste tipo de recomendação". Para cumprir o seu objetivo, um sistema de recomendações deve seguir três etapas necessárias para a geração da recomendação, as quais consistem em: identificação do usuário, definição de um perfil para representar os interesses do usuário e geração das recomendações.

Após a identificação do usuário, o sistema de recomendação deve ser capaz de coletar dados destinados à representação das suas preferências, e isso é realizado através da definição de um *perfil do usuário*. Este perfil pode ser definido de duas maneiras: implícita ou explícita. O perfil definido explicitamente é obtido através de informações explicitamente fornecidas pelo usuário em relação às suas preferências. No perfil definido implicitamente, as informações relacionadas às preferências do usuário são baseadas no comportamento em relação às informações acessadas pelo mesmo (por exemplo, as páginas da *Web* que este acessa).

Dentre as técnicas utilizadas para a definição do perfil implícito do usuário, encontra-se a *Mineração de uso* na *Web*, a qual consiste, em geral, na investigação ou monitoramento das páginas visitadas pelos usuários, na busca de ocorrências e tendências que possam levar a inferências relacionadas aos interesses de informação dos mesmos. Essas inferências, por sua vez,

conduzem à identificação de padrões (CHEN, *et. al.,* 1998) de relevância para o domínio da aplicação, possibilitando, desta forma, a definição dos modelos que representam o comportamento e interesses dos usuários (PIERRAKOS, *et. al.,* 2003). Nesse contexto, os dados de uso podem ser encontrados tanto nos *logs* dos servidores Web, como nas próprias estruturas dos *sites Web*, ou mesmo obtidos através de *cookies,* etc.

Na proporção que ocorre mais interação do usuário com o sistema, este é capaz de realizar refinamentos e atualizações do *perfil do usuário* em relação às suas necessidades de informação, as quais podem se modificar com o tempo. A Figura 6 mostra um modelo genérico de sistema de recomendação (BURKE, 2002).

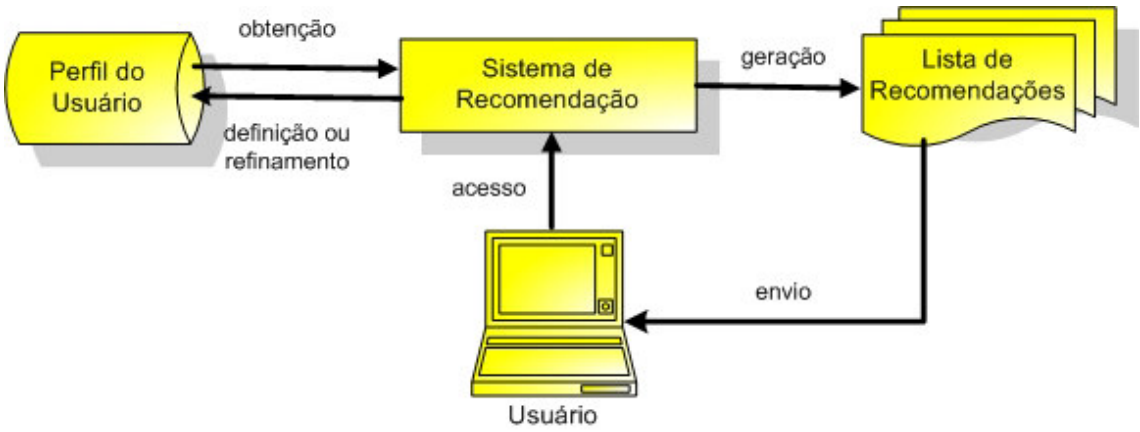


Figura 6 Sistema de Recomendação Genérico (BURKE, 2002)

3.2 Mineração de Uso na Web

A descoberta de conhecimentos em bases de dados contempla um processo não trivial de extração de conhecimentos implícitos, não conhecidos e potencialmente úteis dos dados (FRAWLEY et al. 1991). Nesse contexto, o termo conhecimento refere-se a padrões, relações e associações interessantes entre os dados. De uma forma geral, a mineração na *Web* pode ser conceituada como a descoberta e análise inteligente de informações úteis da *Web* (COOLEY et.al., 1997). Com informações úteis queremos dizer informações que sejam interessantes aos usuários.

A mineração na *Web* é tradicionalmente dividida em três categorias: mineração de conteúdo, mineração de estrutura e *mineração de uso*. Os usuários podem estar interessados, por exemplo, nas informações contidas dentro dos documentos da *Web – mineração de conteúdo*; nas informações contidas entre os documentos da *Web – mineração de estrutura*; ou nas informações contidas nos dados gerados pela utilização ou interação com a Web – *mineração de uso*.

A *Mineração de Uso* da Web (MUW) é a área que se dedica à descoberta dos padrões de uso de páginas *Web* pelos usuários, cujo entendimento possui várias utilidades, tais como fornecer subsídios para sistemas de suporte à tomada de decisão e recomendação. Dentre estes sistemas pode-se citar: recomendação de produtos, definição de campanhas promocionais, planejamento de estratégias de marketing, reestruturação e adaptação automática dos *sites*, entre outros (BUCHNER, MULVENNA, 1999).

O processo geral da MUW (Figura 7) é composto das fases de: aquisição de dados, pré-processamento, descoberta de padrões e pós processamento. Na aquisição de dados, obtêm-se os dados de uso, que podem ser provenientes de várias fontes; identificando-se o seu conteúdo e estrutura. Esses dados podem ser coletados tanto de servidores *Web*, de máquinas clientes ou de fontes intermediárias, como por exemplo os servidores *proxy*. Após a aquisição dos dados, é realizado o pré-processamento dos mesmos, deixando-os “limpos” de ruídos e inconsistências, contemplando basicamente as tarefas de filtragem dos dados, identificação de usuários e identificação de sessões dos usuários. Em seguida, ocorre a fase de descoberta de padrões de uso, que envolve a aplicação de técnicas estatísticas e de aprendizagem de máquina aos dados, tais como: agrupamento, classificação, descoberta de regras de associação e descoberta de padrões seqüenciais. No pós-processamento, o conhecimento é avaliado e apresentado em formato inteligível, como por exemplo, através de relatórios ou ferramentas de visualização.

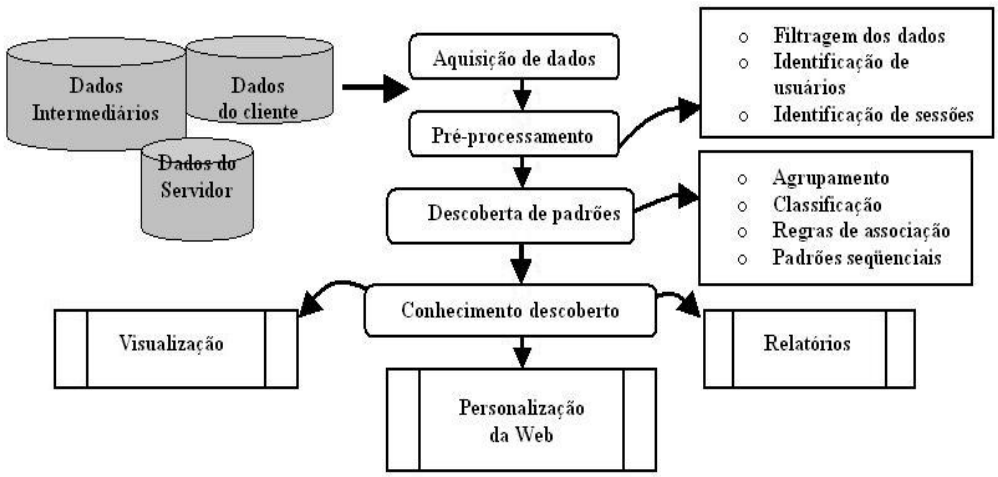


Figura 7 Fluxo genérico da MUV (PIERRAKOS, 2003)

3.3 Engenharia de Software Multiagente baseada na Reutilização

A Engenharia de Domínio Multiagente e a Engenharia de Aplicações Multiagente são processos complementares e interdependentes que possibilitam o reuso no desenvolvimento de software multiagente (LINDOSO, 2006). No contexto da Engenharia de Software Multiagente (GIRARDI, 2004), a Engenharia de Domínio Multiagente objetiva desenvolver artefatos de software reutilizáveis orientados a agentes a partir do conhecimento acerca de um determinado domínio, de requisitos comuns e variáveis de uma família de aplicações nesse domínio, e experiências passadas de desenvolvimento.

Os artefatos resultantes da Engenharia de Domínio podem ser, reutilizados na Engenharia de Aplicações e, no contexto do desenvolvimento multiagente, servem para a construção de aplicações a partir da definição de requisitos específicos, que guiam a seleção, adaptação e composição daqueles artefatos, havendo a influência de padrões de software nas diversas fases desses processos.

A relação entre a Engenharia de Domínio Multiagente e a Engenharia de Aplicações Multiagente é mostrada na Figura 8, onde pode-se observar que ambos os processos são baseados em metodologias de desenvolvimento próprias, respectivamente a MADEM e a MAAEM.

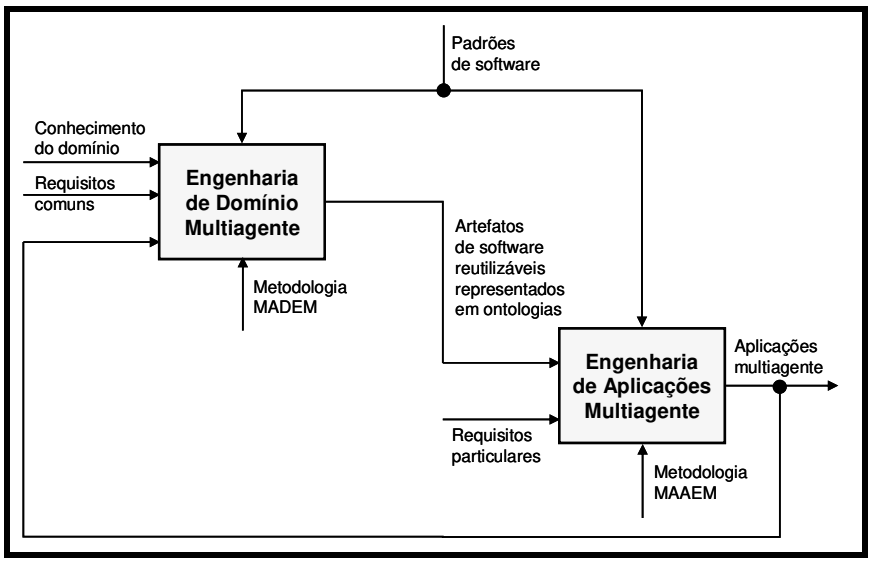


Figura 8 Engenharia de Software Multiagente baseada na Reutilização

3.3.1 A Engenharia de Domínio Multiagente

A Engenharia de Domínio Multiagente é um processo que objetiva desenvolver artefatos de software reutilizáveis orientados a agentes a partir do conhecimento acerca de um determinado domínio (GIRARDI, 2004). Na Figura 9 são mostradas as três fases da Engenharia de Domínio Multiagente, que são a Análise, o Projeto e a Implementação de Domínio, bem como os fluxos de entrada e saída estabelecidos entre elas. Nessa figura pode-se observar que as fases da Engenharia de Domínio possuem como fluxo de entrada padrões extraídos de experiências previamente realizadas.

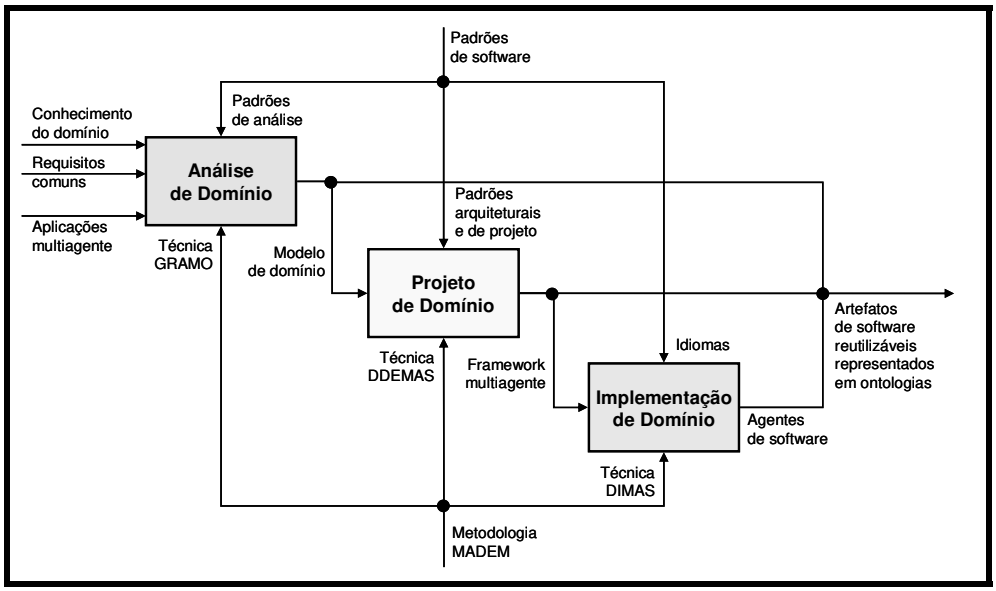


Figura 9 O processo da Engenharia de Domínio Multiagente

Na fase de análise de domínio são realizadas atividades no sentido de identificar oportunidades de reuso e de determinar requisitos comuns e variáveis para uma família de aplicações orientadas a agentes em um domínio. Essa fase possui como produto final um modelo de domínio.

Na fase de projeto de domínio são realizadas atividades que visam a uma solução documentada do problema especificado no modelo de domínio. Essa fase possui como produto final um ou mais *frameworks* e, eventualmente, uma coleção de padrões de projeto.

Na fase de implementação do domínio as atividades desenvolvidas levam à estruturação dos componentes reutilizáveis. Essa fase possui como produto final um conjunto de agentes de software.

3.3.1.1 A Metodologia MADEM

A metodologia MADEM - *Multi-agent Domain Engineering Methodology* (GIRARDI, LINDOSO, 2005a) (GIRARDI, BALBY, 2006), é voltada para a área de Engenharia de Domínio Multiagente, e teve como origem a integração de vários trabalhos de pesquisa do grupo GESEC.

Dentre os trabalhos que originaram a MADEM, pode-se citar: a GRAMO *Generic Requirement Analysis Method based on Ontologies* (FARIA, 2004), que consiste em um método baseado em ontologias para a análise de requisitos genéricos, o qual guia a captura e especificação de requisitos de uma família de sistemas em um domínio de aplicação; e a DDEMAS - *Domain Design for Multi-agent Systems* (FERREIRA, GIRARDI, 2002) (GIRARDI, LINDOSO, 2005a) (GIRARDI, BALBY, 2006), uma técnica para o projeto de domínio de sistemas multiagente, para especificação do projeto de uma família de sistemas em um domínio de aplicação.

3.3.2 A Engenharia de Aplicações Multiagente

A Engenharia de Aplicações Multiagente é um processo complementar à Engenharia de Domínio Multiagente, tendo em vista que foca o desenvolvimento de sistemas individuais a partir do reuso de componentes previamente modelados e produzidos na mesma (LINDOSO, 2006).

Na Figura 10 pode-se observar as fases que compõem a Engenharia de Aplicações Multiagente, sendo elas: Análise, Projeto e Implementação da Engenharia de Aplicações Multiagente, bem como a suas respectivas interligações. A metodologia que orienta a execução dessas fases é a MAAEM (LINDOSO, 2006), a qual será vista com mais detalhes na próxima seção. Como se pode observar, os padrões gerados na Engenharia de Aplicações são reutilizados e revisados na Engenharia de Domínio, podendo servir de retroalimentação para outras fases desta, como também servir de retroalimentação para a própria Engenharia de Aplicações.

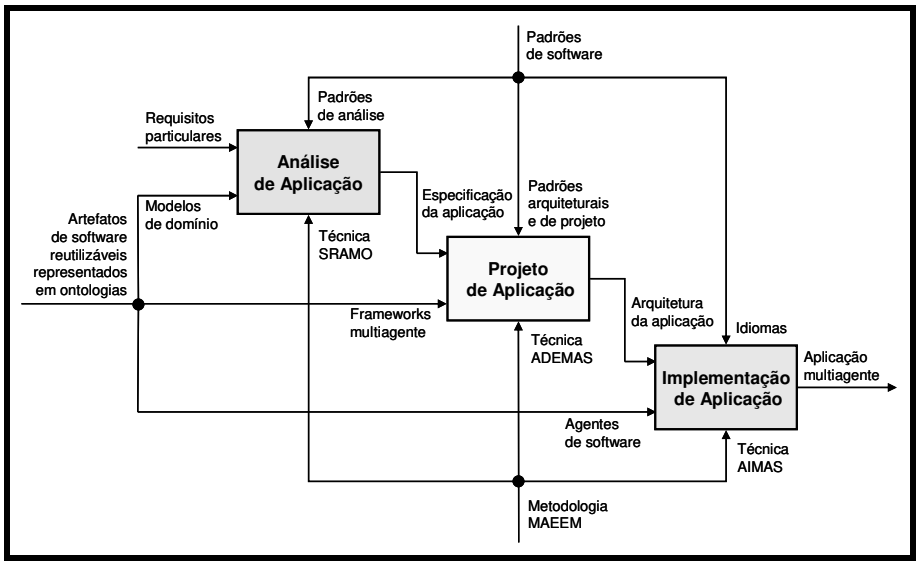


Figura 10 Processo da Engenharia de Aplicações Multiagente

Na fase de análise da aplicação são definidos os requisitos para a aplicação, tendo como base modelos de domínio e as necessidades dos usuários. Dessa forma, busca-se incorporar as funcionalidades requeridas pelos usuários em projetos e componentes pré-existentes, através dos modelos de domínio. As necessidades não cobertas pelos modelos existentes são então consideradas como novos requisitos. Posteriormente, são analisadas as interações entre essas características, com o objetivo de avaliar sua viabilidade e identificar requisitos adicionais e contextos não descritos nos modelos de domínio. Essa fase possui como produto final a especificação da aplicação.

Na fase de projeto de aplicação, é definida uma solução ao problema especificado na fase de análise, tendo como partida a comparação da lista de características e os novos requisitos com a especificação de projeto. Essa lista é especializada para a aplicação em questão a fim de que sejam identificadas as variações em relação ao projeto genérico, as quais são decorrentes dos novos requisitos. Dessa forma, os novos requisitos não satisfeitos pelos componentes já existentes servem de ponderação para as vantagens e desvantagens de se construir novos ou modificar os prontos. Concluindo essa fase, são realizadas as especificações dos novos componentes, a configuração do produto ou uma revisão do projeto. Essa fase possui como produto final a arquitetura da aplicação.

Por último, estando-se de posse dos produtos da análise e do projeto da aplicação, pode-se evoluir para a fase de implementação da aplicação, a qual objetiva a identificação e descoberta dos componentes que serão utilizados na integração final do sistema. Essa fase possui como produto final a aplicação multiagente propriamente dita.

3.3.2.1 A Metodologia MAAEM

A metodologia MAAEM - *Multi-agent Application Engineering Methodology* (LINDOSO, 2006) descreve as fases e atividades necessárias para a análise, o projeto e a implementação de aplicações multiagente através da reutilização de artefatos de software anteriormente produzidos na Engenharia de Domínio Multiagente. A mesma está calcada nos princípios do desenvolvimento baseado em componentes, e dessa forma contempla os processos de seleção, adaptação e composição desses artefatos para a construção de uma aplicação. Nesse contexto, a MAAEM pressupõe a existência de um conjunto de abstrações de software reutilizáveis baseadas em agentes, tais como: modelos de domínio, *frameworks* multiagente, sistemas de padrões e agentes de software. Esses artefatos são produtos da Engenharia de Domínio Multiagente, que no contexto de pesquisa do grupo GESEC (2006) é guiada pela metodologia MADEM – *Multi-agent Domain Engineering Methodology* (GIRARDI, LINDOSO, 2005a) (GIRARDI, BALBY, 2006).

Com o objetivo de promover a integração entre a metodologia MADEM e a metodologia MAAEM, essas metodologias contam com os benefícios oferecidos pela Engenharia do Conhecimento à Engenharia de Software (FALBO et al., 2002), e possuem o seu conhecimento representado através de uma ontologia, a ONTORMAS – *Ontology for Reusing Multiagent Software* (LINDOSO, 2006). Dessa forma, a representação dos artefatos de software resultantes dos processos de desenvolvimento de sistemas multiagente descritos na metodologia MADEM (para o reuso) e na metodologia MAAEM (com o reuso) é realizado através da instanciação das correspondentes subclasses de Tarefas de Modelagem, Produtos de Modelagem, Conceitos de Modelagem e Relacionamentos de Modelagem, seguindo a semântica estabelecida na referida ontologia.

No contexto da ONTORMAS, as tarefas e produtos de modelagem estão bastante relacionados, tendo em vista que a realização de uma tarefa sempre origina um respectivo produto. Adicionalmente, tanto as tarefas como os produtos podem ser classificados em simples ou compostos. Produtos compostos são aqueles que agrupam subprodutos – que podem ser, por sua vez, de quaisquer tipos; produtos simples são aqueles que possuem simplesmente conceitos e relacionamentos de modelagem. Em relação às tarefas, estas são simples quando resultam apenas em produtos também simples; e são compostas quando originam produtos compostos e, por conseguinte, possuem subtarefas – que igualmente podem ter quaisquer tipos – relacionados com os componentes de tais produtos.

A utilização de ontologias para representação do conhecimento ocorre, principalmente devido ao fato de serem estruturas de representação adequadas para a especificação de abstrações de software de alto nível, tendo em vista as características relacionadas à univocidade, formalismo, reusabilidade e adaptabilidade (CHANDRASEKARAN, JOSEHSON, 1999) (FALBO, 2002). Nesse contexto, a utilização de ontologias viabilizou a reutilização dos produtos da MADEM como insumos para a MAAEM, uma vez que aqueles também são representados através dessa ontologia, tornando-os compatíveis entre si.

As fases, subfases, passos, produtos e subprodutos da metodologia MAAEM, são mostradas na Tabela 2 (LINDOSO, 2006), as quais serão posteriormente detalhadas no capítulo relativo à modelagem da ferramenta PROPOST.

<table>
<tr><th colspan="2">Fases</th><th>Passos</th><th colspan="2">Produtos</th></tr>
<tr><td colspan="2" rowspan="4">Análise
da
Aplicação</td><td>Modelagem de Conceitos</td><td>Modelo de Conceitos</td><td rowspan="4">Especificação
da
Aplicação</td></tr>
<tr><td>Modelagem de Objetivos</td><td>Modelo de Objetivos</td></tr>
<tr><td>Modelagem de Papéis</td><td>Modelo de Papéis</td></tr>
<tr><td>Modelagem de Interações entre Papéis</td><td>Modelo de Interações entre Papéis</td></tr>
<tr><td rowspan="6">Projeto
da
Aplicação</td><td colspan="2">Modelagem do Conhecimento da Sociedade Multiagente</td><td>Modelo do Conhecimento da Sociedade Multiagente</td><td rowspan="4">Modelo
da
Arquitetura</td><td rowspan="6">Arquitetura
da
Aplicação</td></tr>
<tr><td rowspan="3">Projeto
da
Arquitetura</td><td>Modelagem da Sociedade Multiagente</td><td>Modelo da Sociedade Multiagente</td></tr>
<tr><td>Modelagem das Interações entre Agentes</td><td>Modelo das Interações entre Agentes</td></tr>
<tr><td>Modelagem dos Mecanismos de Cooperação e Coordenação</td><td>Modelo dos Mecanismos de Cooperação e Coordenação</td></tr>
<tr><td rowspan="2">Projeto
do
Agente</td><td>Modelagem do Conhecimento e das Atividades do Agente</td><td>Modelo do Conhecimento e das Atividades do Agente</td><td rowspan="2">Modelo
do
Agente</td></tr>
<tr><td>Modelagem dos Estados do Agente</td><td>Modelo dos Estados do Agente</td></tr>
<tr><td colspan="2" rowspan="2">Implementação
da
Aplicação</td><td>Mapeamento de Agentes do Projeto à Implementação e de Responsabilidades em Comportamentos</td><td>Modelo de Agentes e Comportamentos</td><td colspan="2" rowspan="2">Modelo de Implementação da Aplicação</td></tr>
<tr><td>Mapeamento de Interações entre Agentes em Atos de Comunicação</td><td>Modelo de Atos de Comunicação entre Agentes</td></tr>
</table>

Tabela 2 Fases, subfases, passos, subprodutos e produtos da Metodologia MAEEM

3.3.3 Ontologias e as vantagens da sua utilização

Os produtos resultantes da MADEM bem como os produtos resultantes da MAAEM são representados em ontologias. Esta seção aborda as vantagens resultantes da utilização das ontologias na representação do conhecimento, bem como no processo de desenvolvimento de software.

Os sistemas de informação estão cada vez mais voltados ao suporte do negócio empresarial. Por isso, torna-se cada vez maior a demanda por sistemas que contemplem funcionalidades muitas vezes relacionadas a regras de negócio existentes apenas em forma de conhecimentos tácitos, ou seja, aqueles que por não serem explícitos, são mais difíceis de serem comunicados. Segundo Nonaka (1997): “o conhecimento tácito está enraizado nas ações e experiências de um indivíduo, bem como em suas emoções, valores ou ideais". Por esse motivo, o processo da sua obtenção e especificação, em geral é caro e demorado.

Devido à falta de procedimentos adequados para armazenamento, recuperação e compartilhamento do conhecimento produzido no desenvolvimento dos sistemas de informação, é comum que este não seja reaproveitado em novos projetos, justamente por desconhecimento da existência dos mesmos. Mas assim como a falta de informação é prejudicial, o seu excesso também o é, principalmente quando os mesmos não estão estruturados de forma a permitir uma busca adequada daquilo que se procura, bem como o entendimento da sua semântica. E nesse contexto, as ontologias vêm sendo utilizadas para a representação do conhecimento, a ser utilizado tanto por indivíduos como por sistemas.

De acordo com Gruber (2002) uma ontologia é uma representação explícita dos objetos, conceitos e outras entidades que assumimos existirem em uma área de interesse, além das relações entre esses conceitos e restrições expressos através de axiomas.

Devido às características da estruturação das ontologias, as mesmas proporcionam a captura e armazenamento das informações com maior “riqueza” de informações em relação às bases de dados convencionais, e possuem uma semântica de mais fácil compreensão e implementação (CHANDRASEKARAN e JOSEHSON, 1999). Além disso, estas possuem maior compatibilidade com a forma de interpretação usada nos paradigmas atuais de desenvolvimento: na Orientação a Objetos, por exemplo, podem ser implementadas como modelos de classes, bem como no desenvolvimento multiagente, onde facilitam e viabilizam a comunicação entre os agentes. Devido a isso, as mesmas são consideradas atualmente a forma mais adequada para a representação do conhecimento (GRUBER, 2002). Isso permite não apenas o aumento na produtividade e qualidade da produção de software, mas também a criação de produtos cada vez mais “inteligentes” e capazes de otimizar a execução de tarefas, bem como o suporte na tomada de decisões.

A utilização de ontologias proporciona uma infra-estrutura para integrar sistemas inteligentes no nível do conhecimento, proporcionando as vantagens de compartilhamento do conhecimento (cooperação); integração da informação (interoperação); servir de fontes de consulta (informação) e serem reusáveis na modelagem de sistemas (GIRARDI, 2005).

Existe ainda a vantagem de tornar a busca mais rápida e precisa, pois como esta é baseada na semântica, caso não se encontre “exatamente” aquilo que se procura, pode-se obter como retorno algo mais próximo ao seu significado.

3.3.4 ONTOWUM e ONTOINFO

A seguir são apresentados conceitos básicos das ontologias ONTOINFO (GIRARDI, et. al., 2005) (JANSEN PEREIRA, 2006) e ONTOWUM (MARINHO, 2005) (GIRARDI, LINDOSO, 2005c) (GIRARDI, BALBY, 2006) utilizadas na modelagem da ferramenta PROPOST. Estas ontologias descrevem famílias de produtos de software para o desenvolvimento de aplicações nas áreas de Recuperação e Filtragem de Informações, respectivamente.

A ONTOWUM (Apêndice A) é constituída por modelos de domínio e *frameworks* multiagentes baseados em ontologias para a personalização da *Web*, tendo como base a modelagem de usuários e a mineração de uso. Na prática, ela descreve um conjunto de abstrações de software que podem ser reutilizadas por famílias de aplicações que desejem oferecer serviços personalizados aos seus usuários através da MUW e do paradigma dos SMA (Sistemas Multiagentes). Seu objetivo geral é *prover recomendações através de mineração de uso*. Para a realização deste objetivo, a mesma contempla objetivos específicos referentes a *modelar adaptação através de mineração de uso* e *modelar usuários através de mineração de uso*.

Para alcançar os objetivos específicos acima citados, são necessárias as responsabilidades a serem executadas, a seguir: a *monitoração do usuário*, que objetiva a captura das informações provenientes da navegação do usuário; a *descoberta de padrões de uso*, que objetiva identificar padrões de comportamento do usuário baseado nos dados provenientes da sua monitoração; a *modelagem do usuário corrente*, que objetiva construir um modelo do usuário corrente baseado no seu padrão de uso; a *construção do modelo de adaptação*, que visa construir modelos de adaptação de interface, baseado nos quais a adaptação será realizada; a *classificação do usuário corrente*, que objetiva classificar o usuário corrente em um dos modelos de usuário existentes; a *manutenção de dados de uso*, que objetiva o armazenamento e atualização de um repositório de dados de uso; e finalmente a *adaptação da interface*, que visa prover adaptações de interface *Web* ao usuário, de acordo com seu comportamento de navegação e padrão de uso.

A ONTOINFO (Apêndice B) é constituída por modelos de domínio e *frameworks* multiagentes baseados em ontologias para a recuperação e filtragem de informação. Na prática ela define um conjunto de abstrações de software que possam ser reutilizadas por famílias de aplicações que desejem oferecer serviços de pesquisa aos seus usuários através de técnicas de recuperação e filtragem e do paradigma dos SMA (Sistema Multiagente). No domínio da recuperação e filtragem, o objetivo geral da ONTOINFO é *satisfazer necessidades de informações dos usuários*.

Os objetivos específicos imediatos para alcançar este objetivo são: *satisfazer necessidades pontuais por técnicas de recuperação* e *satisfazer necessidades de longo prazo por técnicas de filtragem*. Este último objetivo se subdivide em três sub-objetivos específicos referentes a cada uma das técnicas de filtragem, a seguir: *baseada no conteúdo, colaborativa e híbrida*.

A reutilização da ONTOINFO na modelagem da PROPOST deu-se no contexto da recuperação da informação. Nesse contexto, para que seja alcançado o objetivo específico *satisfazer necessidades pontuais por técnicas de recuperação*, são necessárias as responsabilidades a serem executadas, a seguir: a *especificação da necessidade pontual de informação*, que trata de receber a entrada da consulta pontual do usuário; a *representação da consulta do usuário*, que corresponde à criação de uma representação interna para a consulta especificada pelo usuário; a *representação e indexação dos itens de informação*, que corresponde à criação de uma representação interna e posterior indexação (na forma de índice invertido) dos itens de informação; a *comparação e análise de similaridade*, que trata da comparação da representação interna da consulta do usuário e itens de informação; e a *apresentação dos itens de informação recuperados*, que trata da entrega dos resultados solicitados ao usuário.

3.3.5 Considerações Finais do Capítulo

Este capítulo teve como objetivo apresentar conceitos relacionados às principais tecnologias utilizadas na modelagem e desenvolvimento de funcionalidades da ferramenta PROPOST. Dessa forma, além de definições relacionadas à recuperação e filtragem de informações, foi também apresentado um panorama geral sobre os conceitos de ontologias e as vantagens da sua utilização, tendo em vista que a PROPOST é uma ferramenta desenvolvida a partir do conhecimento armazenado em ontologias.

Igualmente, foi também apresentada uma breve descrição das metodologias MADEM e MAAEM, utilizadas para a modelagem da referida ferramenta. Tal descrição foi abordada no contexto da Engenharia de Software Multiagente baseada na reutilização e suas respectivas subdivisões relacionadas à Engenharia de Domínio Multiagente, na qual a metodologia MADEM está inserida, e a Engenharia de Aplicações Multiagente, na qual a metodologia MAAEM está inserida. Por último, foram abordados conceitos básicos das ontologias ONTOWUM e ONTOINFO, utilizadas na modelagem da PROPOST.

4 MODELAGEM DA *PROPOST - PROJECT PORTFOLIO SUPPORT TOOL*

Este capítulo apresenta a modelagem da aplicação PROPOST, uma aplicação multiagente destinada a suporte à gestão de portifólio de projetos. Essa ferramenta foi modelada usando a ferramenta dirigida por ontologia ONTORMAS, e com reuso das ontologias ONTOINFO e ONTOWUM, as quais descrevem famílias de produtos de software para o desenvolvimento de aplicações nas áreas de Recuperação e Filtragem da Informação, respectivamente.

A PROPOST objetiva dar suporte a um problema da atualidade enfrentado pelas empresas, referente à redundância na definição do portifólio de projetos, tendo como foco o reaproveitamento dos sistemas existentes. Não obstante, a mesma também proporciona suporte a outras atividades relacionadas à gestão do portifólio (seleção, priorização e avaliação), necessárias à composição do referido portifólio.

Adicionalmente, o desenvolvimento dessa ferramenta também constitui um estudo de caso de uso para avaliação da metodologia MAAEM, bem como demonstra a reutilização dos modelos dirigidos pelas ontologias acima citadas.

4.1 Descrição do Problema

4.1.1 Cenário Atual

Conforme mostrado na Figura 11, o Cenário Atual, representa o processo de definição do portifólio de projetos nas organizações em geral, cujos processos de seleção, priorização e avaliação, tradicionalmente depende em grande parte de experiências isoladas dos analistas.

Mesmo as principais ferramentas atualmente existentes, tais como: *Primaerva Enterprise* (http://www.primavera.com), *Planview Enterprise* (http://www.planview.com), *CA Portifolio Manager* (http://www.niku.com/) *e Microsoft Project Enterprise* (http://www.microsoft.com/brasil/office/project), dentre outras, ainda carecem de mecanismos mais efetivos relacionados ao suporte à seleção e priorização dos novos projetos.

Na definição do portifólio de projetos nas empresas em geral, não são raros os casos em que os envolvidos desconhecem a existência do portifólio atual de sistemas de informação, muitos dos quais poderiam estar sendo reutilizados para atender às necessidades existentes. Devido a isso, é comum haver redundância no desenvolvimento de projetos, ou seja, é desenvolvido um novo projeto de sistema de informação similar a um sistema já existente. E com isso, há desperdício de tempo e dinheiro devido a duplicidades no portifólio de projetos a serem desenvolvidos.

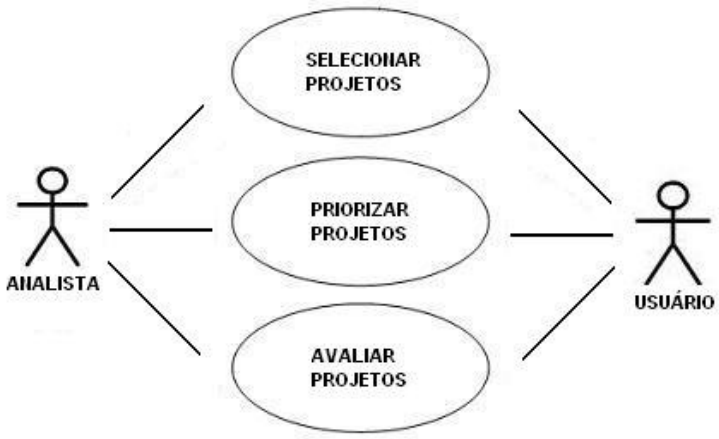


Figura 11 Cenário Atual da definição do portifólio nas empresas

4.1.2 Cenário Proposto

O Cenário Proposto na Figura 12, objetiva agregar valor à solução do problema anteriormente descrito, uma vez que a definição dos projetos a compor o portifólio da organização será realizada não apenas baseada na experiência pessoal dos profissionais diretamente envolvidos, mas também com o suporte da ferramenta PROPOST, descrita neste trabalho.

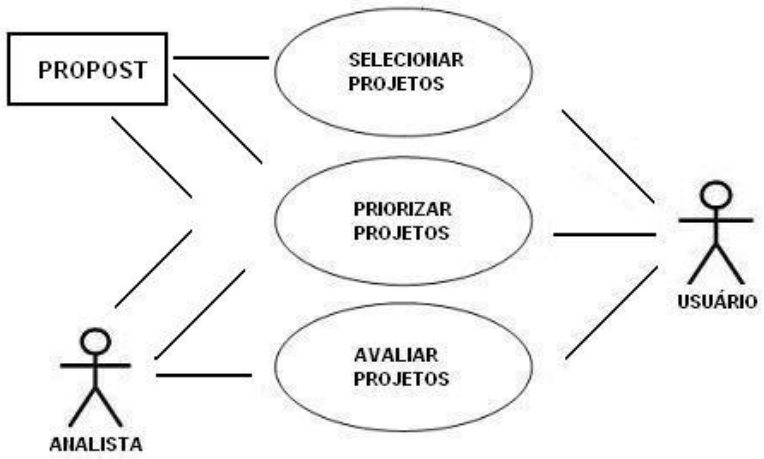


Figura 12 Cenário Proposto da definição do portifólio, incluindo a PROPOST

O suporte proporcionado pela PROPOST considera não apenas a gestão do portifólio propriamente dita (seleção, priorização e avaliação de projetos), mas também o reaproveitamento de sistemas existentes para a composição do referido portifólio. Ou seja, antes de decidir sobre um novo projeto para desenvolvimento de uma *solução de software* [2], a *área gerencial* tem ao seu dispor uma visão de potenciais soluções existentes que eventualmente poderiam estar suprindo suas necessidades.

No contexto anteriormente descrito, o principal foco da ferramenta ocorre no processo da seleção de projetos, onde são providas facilidades relacionadas à recomendação colaborativa de *soluções de software* (sistemas de informação, programas, etc) e recuperação de informações relativas a estas soluções, a partir da reutilização das famílias de produtos ONTOINFO e ONTOWUM, resultantes dos trabalhos desenvolvidos no Grupo de Pesquisa em Engenharia de Software e Conhecimento (GESEC, 2006).

[2] O termo *solução de software* pode ser utilizado em referência a qualquer software utilizado em uma organização. No contexto deste trabalho o mesmo fará referência principalmente aos sistemas de informação utilizados na mesma.

4.2 Fundamentos da Ferramenta PROPOST

Conforme descrito no capítulo 1, a GPP envolve um processo dinâmico de Seleção, Priorização e Avaliação de projetos (PORTER, 2003). A seleção de projetos envolve a identificação e posterior escolha de uma lista de projetos considerados viáveis de execução, a partir de várias alternativas de projetos candidatos; a priorização de projetos contempla a listagem dos projetos selecionados em uma ordem de prioridade, de acordo com critérios previamente estabelecidos pela organização; e a avaliação de projetos está relacionada ao monitoramento constante do status dos projetos em relação ao seu valor agregado ao negócio.

A ferramenta PROPOST provê suporte aos processos da GPP, tendo como principal funcionalidade as *recomendações de soluções de software*, existentes para suprir demandas de informação de *áreas gerenciais* com necessidades similares. Além de facilitar a descoberta de novas demandas de projetos, essa funcionalidade tem como benefício adicional evitar redundâncias em desenvolvimentos de projetos. Tais recomendações estão alinhadas a um dos principais objetivos da gerência de portifólio de projetos, que é a otimização dos recursos existentes, bem como a observância aos princípios da reutilização, no sentido em que, antes de partir para a aquisição de novos recursos, deve-se primeiro buscar o reaproveitamento do que já existe.

De uma maneira geral, o suporte da ferramenta ao processo de seleção de projetos ocorre através das recomendações anteriormente citadas; enquanto o suporte ao processo de priorização dos projetos ocorre através de funcionalidades relacionadas às estimativas e listagem baseada em critérios de pontuação para as soluções de software candidatas à composição do portifólio; e o suporte ao processo de avaliação ocorre através da análise de uso dos sistemas de informação e da análise de status dos projetos existentes.

De acordo com os requisitos descritos para a ferramenta, o acesso ao conhecimento pode ocorrer de duas formas, sendo a principal delas através de recomendações personalizadas de *soluções de software* (sistemas de informação, programas, etc.) consideradas relevantes para a área funcional em questão, de acordo com o perfil dos seus processos de negócio. A outra forma de acesso à informação poderá ocorrer através de consultas explicitamente definidas pelos usuários na base de dados de *soluções de software*. Devido a isso, consideramos que a ferramenta PROPOST pertence a uma família de aplicações nas quais as informações são obtidas por processos de Recuperação e Filtragem de Informações.

A ferramenta PROPOST foi modelada tomando-se por base as fases descritas na metodologia MAAEM (LINDOSO, 2006). Os produtos resultantes dessa modelagem foram obtidos com a reutilização dos modelos dirigidos pelas ontologias: ONTOINFO (GIRARDI, et. al., 2005) (JANSEN PEREIRA, 2006) e ONTOWUM (MARINHO, 2005) (GIRARDI, LINDOSO, 2005c) (GIRARDI, BALBY, 2006), as quais descrevem famílias de produtos de software para o desenvolvimento de aplicações nas áreas de Recuperação e Filtragem de Informações, respectivamente. Adicionalmente foram introduzidos conceitos específicos alusivos ao domínio da Gestão de Portifólio de Projetos.

4.3 Análise da Ferramenta PROPOST

Na fase de Análise da Aplicação, conforme definido na metodologia MAAEM, o principal produto resultante é a especificação da aplicação, a qual é resultante das seguintes atividades:

- Modelagem de Conceitos
- Modelagem de Objetivos
- Modelagem de Papéis
- Modelagem de Interações entre Papéis

4.3.1 Modelagem de Conceitos

A Figura 13 ilustra o Modelo de Conceitos da Ferramenta PROPOST, no qual o objetivo principal resume-se em exibir as principais idéias e requisitos identificados para a aplicação, expressos sob a forma de conceitos semanticamente relacionados. O produto desta fase é um modelo de conceitos, o qual é apresentado sob a forma de uma rede semântica, obtido a partir de refinamentos progressivos em relação aos conceitos obtidos ao longo do levantamento de requisitos da aplicação.

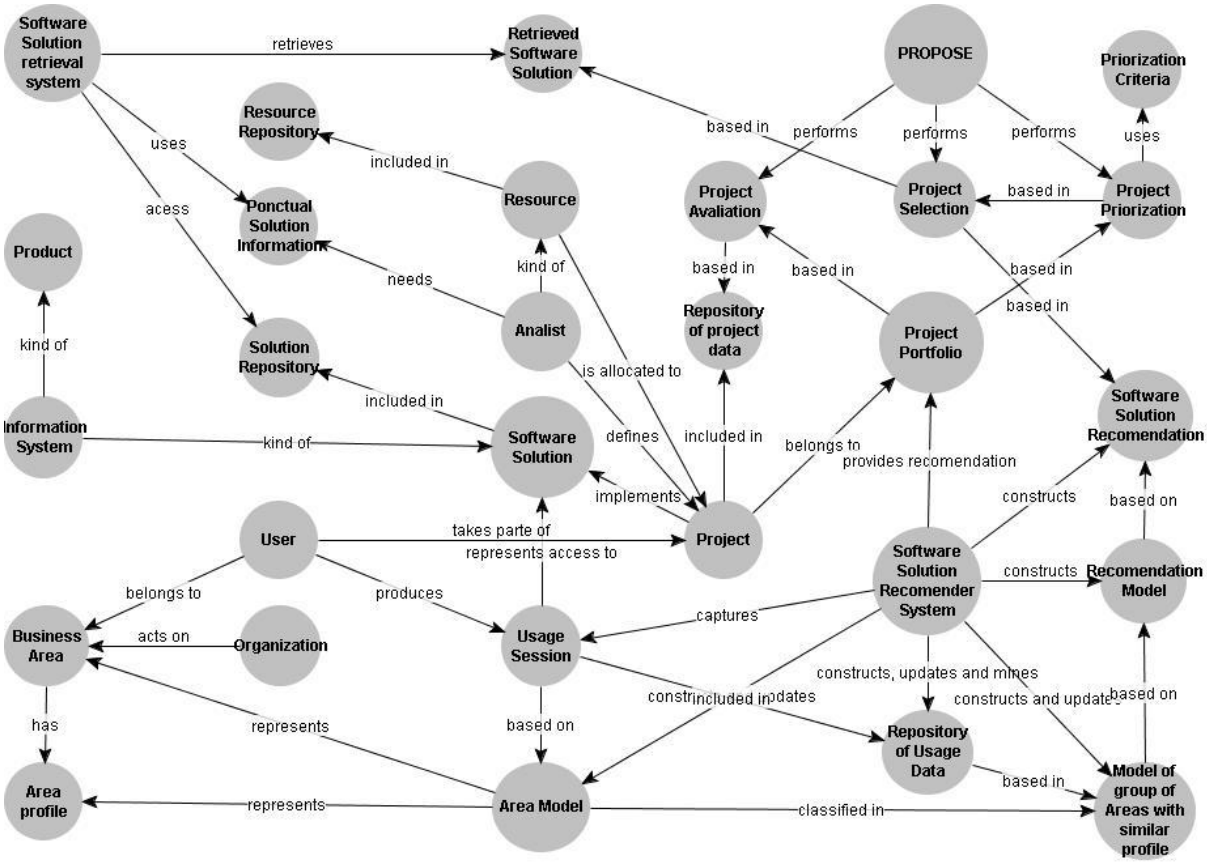


Figura 13 Modelo de Conceitos da ferramenta PROPOST

4.3.2 Glossário dos conceitos utilizados

A seguir são descritos os principais conceitos do modelo de conceitos da ferramenta PROPOST:

- *Organização*: qualquer entidade de estrutura formal de autoridade (empresa pública, privada, etc.), através da qual as subdivisões de trabalho são definidas e coordenadas para viabilizar o alcance de seus objetivos. No contexto deste trabalho uma organização está dividida em várias *áreas gerenciais* (ou gerências).
- *Área Gerencia (ou área)l*: também conhecida como gerência de área ou simplesmente área, representa a unidade gerencial de mais baixo nível em uma escala hierárquica formal na organização.
- *Áreas Similares*: são *áreas gerenciais* consideradas similares à área gerencial em referência devido às mesmas possuírem um comportamento similar de utilização de *soluções de software*.
- *Solução de software (ou solução)*: faze referência principalmente aos sistemas de sistema de informação, podendo também ser estas qualquer outro tipo de software utilizado por uma *área gerencial*, e que seja potencialmente reutilizável por outra *área similar*. Na prática, o uso de uma *solução de software* existente, de uma maneira geral, requer a criação de um novo projeto, no qual serão contemplados desde o estudo das necessidades de infra-estrutura adicional (servidores, redes, licenças, etc) para o uso da solução; o levantamento das necessidades de seleção, adaptação e composição de componentes e funcionalidades; bem como sua implementação propriamente dita.

  Obs: No contexto deste trabalho, o foco das *soluções de software* está voltado principalmente para os sistemas de informação.

- *Recomendação de Solução*: consiste em recomendar o uso de uma (ou mais) *soluções de software* existentes, a serem potencialmente utilizadas para suprir necessidades de informação de *áreas gerenciais* consideradas similares. Na prática, seu principal objetivo é contribuir para que antes de decidir sobre o desenvolvimento de um projeto completamente novo de um sistema de informação, a *área gerencial* deva levar em conta a existência de soluções potencialmente úteis dentre as já existentes no atual portifólio da empresa.
- *Sessão de Uso*: representa a ocorrência da utilização de uma *solução de software* por um usuário. Tendo em vista que cada usuário pertence a somente uma *área gerencial*, pode ser considerado para efeito deste trabalho, que as *sessões de uso* deste usuário também representam a utilização de uma *solução de software* pela *área gerencial* onde o mesmo está alocado.
- *Repositório de Soluções*: é o local onde estão armazenadas informações sobre as *soluções de software* existentes.
- *Perfil da Área*: constitui uma representação da *área*, na qual constam como informações principais a sua identificação e ocorrências de uso de *soluções de software*. Ou seja, o perfil representa um "comportamento de consumo" de *soluções de software* pelas *áreas*, a partir do qual são filtradas e recomendadas outras soluções consideradas relevantes para a mesma.
- *Modelo de Área*: é a representação de um perfil genérico da *área gerencial*, baseado nos padrões extraídos das utilizações de software pela mesma.
- *Repositório de Dados de Uso*: é um repositório no qual estão armazenadas informações sobre todas as *sessões de uso* de *soluções de software* pelas *áreas gerenciais*.
- *Modelo de Grupo de Áreas*: refere-se a um agrupamento de modelos de *áreas similares*, a partir dos vários *modelos de área* existentes.

Na modelagem da PROPOST o reuso das famílias de produtos de software relativos aos conceitos da ONTOWUM e ONTOINFO demandaram adaptações para refletir conceitos peculiares à Gestão de Portifólio. Adicionalmente, outros conceitos novos referentes a este domínio foram introduzidos para complementar a necessidade de outras funcionalidades requeridas pela ferramenta.

4.3.3 Conceitos Reutilizados da ONTOWUM e ONTOINFO

No modelo de conceitos da PROPOST foram reutilizados vários conceitos originários da ONTOINFO e ONTOWUM, bem como introduzidos outros particulares do domínio da gerência de portifólio de projetos.

A Figura 14 mostra o modelo de conceitos da ONTOWUM, no sentido de proporcionar melhor visualização sobre as reutilizações realizadas.

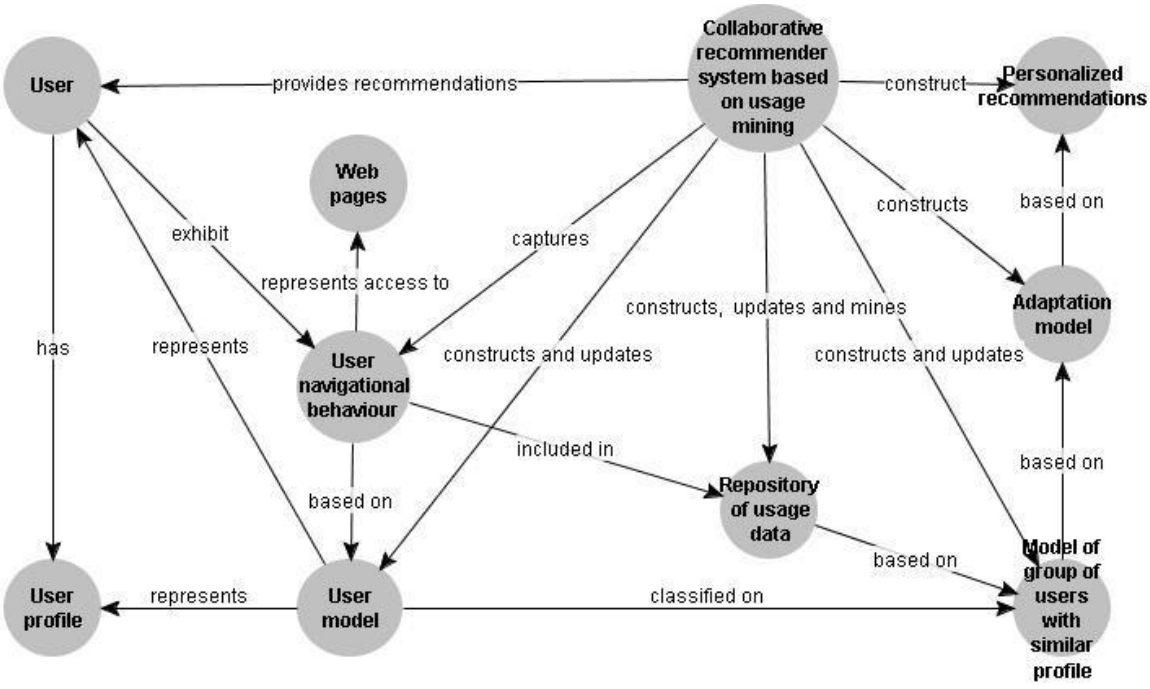


Figura 14 ONTOWUM: Modelo de Conceitos (GIRARDI, BALBY, 2006)

Assim, por apresentarem variabilidade do tipo mandatória, os seguintes conceitos existentes na ONTOWUM foram obrigatoriamente reutilizados: *sistema de recomendações colaborativas baseadas em mineração de uso, comportamento de navegação do usuário, páginas da Web, modelo de usuário, perfil de usuário, recomendações personalizadas, modelo de adaptação, modelo de grupo de usuários com perfis similares*.

De tal reuso, em função da especialização para o domínio da gerência de portifólio de projetos, surgiram então, nessa ordem de correspondência, os conceitos a seguir: *sistema de recomendação de soluções de software, sessão de uso, repositório de solução de software, modelo da área, perfil da área, recomendação de soluções, modelo de recomendação personalizada, modelos de grupo de áreas com perfis similares*.

Os conceitos de *usuário* e *repositório de dados de uso* foram reusados sem adaptação, porém, baseado no conceito do *usuário*, foi também especializado o conceito do *analista*. Todos os demais conceitos não relacionados acima são novos.

Em relação aos relacionamentos, o ponto de partida é o conceito de *sistema de recomendação de soluções de software*, o qual visa prover *soluções de software*, e o que faz capturando as definições de uso das *áreas*, através das quais são construídos e atualizados os *modelos da área*, e construídos os *modelos de recomendação personalizada*.

As *sessões de uso* de *solução de software* são baseadas nas utilizações, pelos usuários, das soluções já existentes no repositório de soluções da organização. Por outro lado, o *perfil da área*, que é representado pelo *modelo da área*, tem como partes a identificação da área e os dados de uso das soluções pelas áreas, os quais são armazenados em um *repositório de dados de uso*. A extração destes dados possibilita a criação dos *modelos de grupo de áreas*, em um dos quais é classificado cada *modelo de área gerencial*. O *perfil da área* representa um comportamento de consumo de *soluções de software* e, a partir do mesmo, são recomendadas *soluções de software* para as *áreas gerenciais*.

A Figura 15 mostra o modelo de conceitos da ONTOINFO. Desse modelo foram reusados os conceitos relativos a *necessidade dinâmica de informação* e *item de informação*.

De tal reuso, em função da especialização para o domínio da gerência de portifólio de projetos, estes conceitos foram especializados, respectivamente, para o conceito de *necessidade de informação pontual* e conceito de *solução de software*. A necessidade de informação pontual, na prática, está relacionada às consultas eventuais que um analista poderá fazer *ao repositório de soluções de software* durante a fase de seleção de projetos, no sentido de obter informações mais detalhadas sobre determinadas soluções existentes. A *solução de software* representa os sistemas de informação existentes.

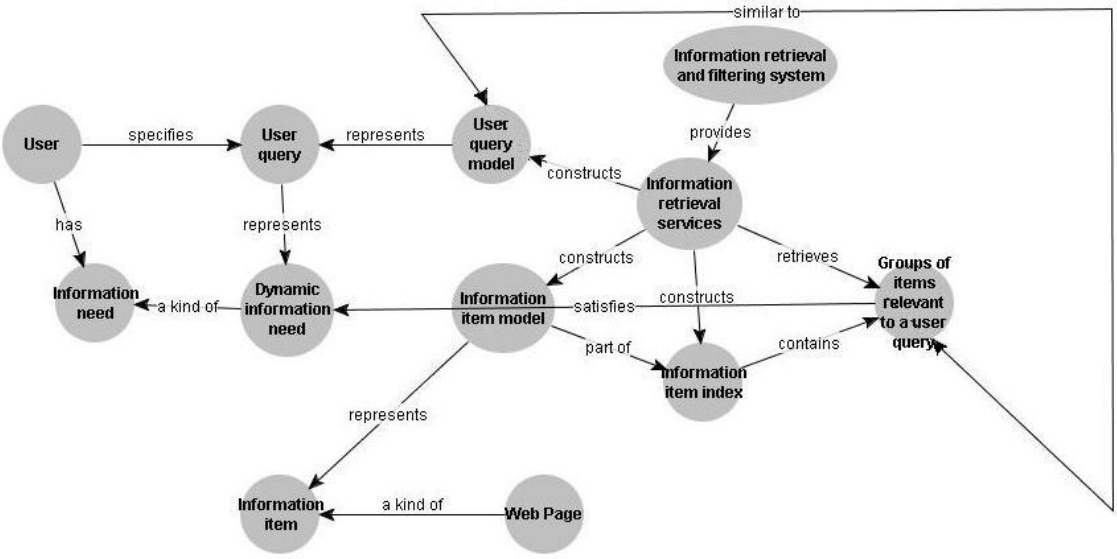


Figura 15 ONTOINFO: Modelo de Conceitos (JANSEN PEREIRA, 2006)

4.3.4 Modelagem de Objetivos

O Modelo de Objetivos especifica uma hierarquia entre o objetivo geral, os objetivos específicos resultantes do mesmo, e as responsabilidades necessárias para alcançar esses objetivos. Na prática, esse modelo representa os requisitos da aplicação em alto nível de abstração.

Constam também nesse modelo as entidades externas que interagem com a aplicação ao longo da sua utilização.

A Figura 16 e Figura 17 mostram, respectivamente, os modelo de objetivos da ONTOWUM e da ONTOINFO, no sentido de proporcionar melhor visualização sobre as reutilizações realizadas.

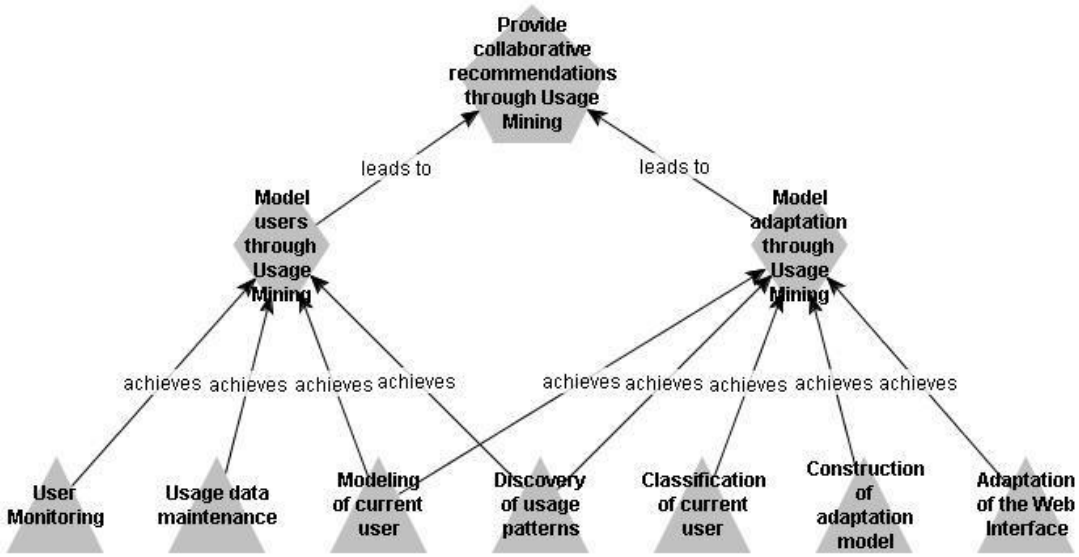


Figura 16 ONTOWUM: Modelo de Objetivos (GIRARDI, BALBY, 2006)

A Figura 18 apresenta o modelo de objetivos da ferramenta PROPOST. Nesse modelo são reusados requisitos dos modelos de domínio das famílias ONTOWUM e ONTOINFO. Sendo assim, de forma análoga como procedemos na modelagem de conceitos, partimos da seleção, adaptação e composição de objetivos, responsabilidades e entidades externas presentes no modelo correspondente nas referidas ontologias.

O objetivo geral da ONTOWUM em *prover recomendações através de mineração de uso* foi reutilizado com adaptação em termos da gerência de portifólio, resultando no objetivo *prover suporte na seleção de projetos*. Deste objetivo foram especializados os correspondentes objetivos específicos referentes a *modelar adaptação através de mineração de uso* e *modelar usuários através de mineração de uso* resultando, respectivamente, em *modelar recomendações* e *modelar áreas*.

Em relação às responsabilidades reutilizadas da ONTOWUM, todas aquelas encontradas no modelo de objetivos reutilizado foram aproveitadas, visto possuírem variabilidade mandatória, sendo elas: *construção do modelo de adaptação, adaptação da interface da Web, classificação do usuário corrente, descoberta de padrões de uso, modelagem do usuário corrente, monitoração do usuário e manutenção de dados de uso*.

Dessa forma, à exceção da última, as restantes sofreram especializações em decorrência do domínio enfocado, originando, respectivamente, as seguintes: *construção do modelo de recomendação, recomendação de soluções, classificação da área corrente, descoberta de padrões de uso de solução, modelagem da área corrente, monitoração de uso*.

Em relação à reutilização de objetivos da ONTOINFO (Figura 17), o objetivo de *satisfazer necessidade pontual de informação* foi especializado no objetivo de prover *informação de solução de informação*. No tocante às suas responsabilidades, as mesmas foram reutilizadas da maneira como estavam, sendo elas: *representação da consulta do usuário, representação e indexação da solução e comparação e recuperação*.

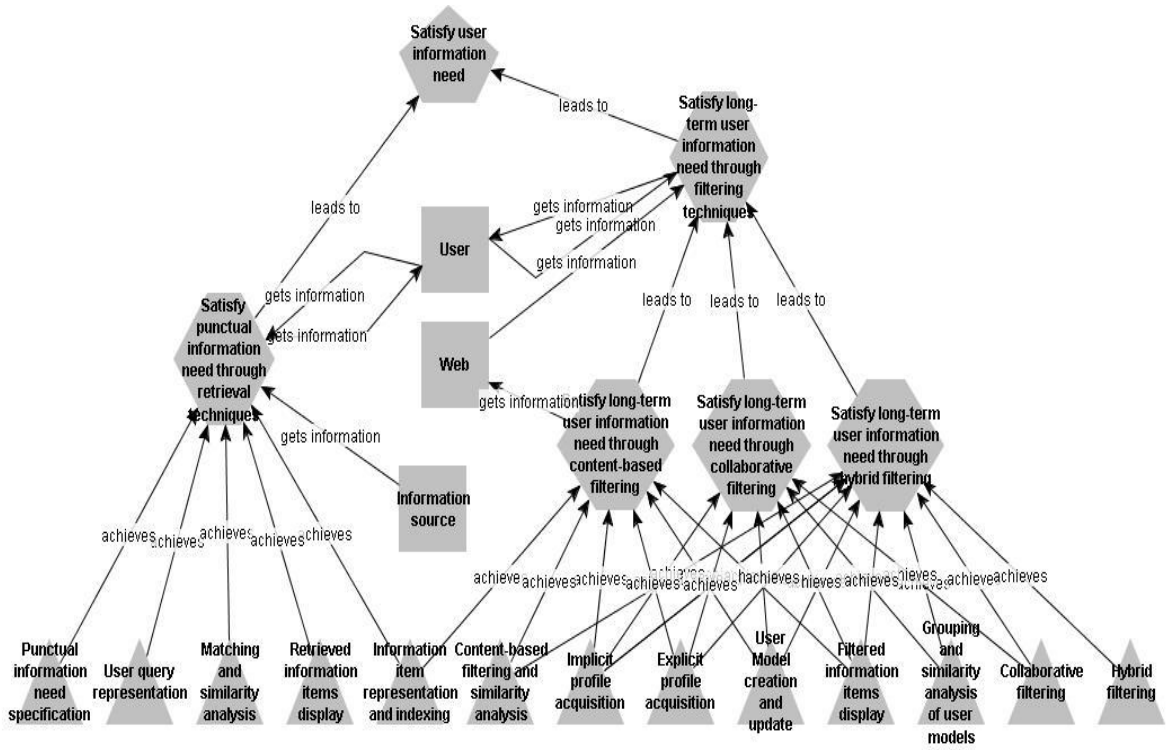


Figura 17 ONTOINFO: Modelo de Objetivos (JANSEN PEREIRA, 2006)

No modelo de objetivos da PROPOST (Figura 18), além dos objetivos e responsabilidades anteriormente citados, foram criados outros peculiares ao domínio da gestão de portifólio. Em relação aos novos objetivos, foram criados: *prover suporte à priorização de projetos e prover suporte à avaliação de projetos*. Dentre as responsabilidades criadas, temos: *elaboração de estimativas de projeto, elaboração da priorização de projetos, elaboração da simulação do portifólio, elaboração da análise de uso de sistemas e elaboração da análise do status dos projetos*.

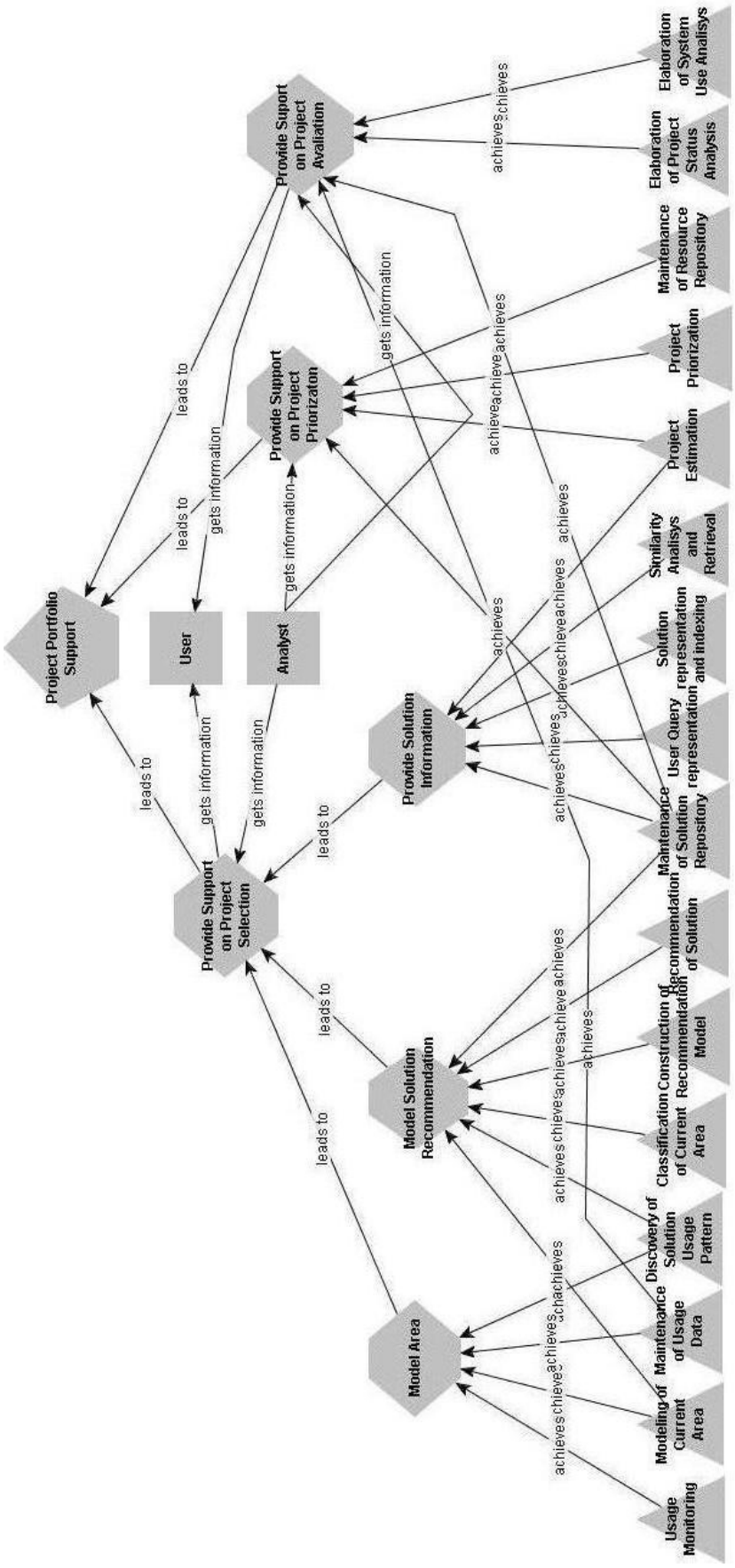


Figura 18 PROPOST: Modelo de Objetivos

4.3.5 Modelagem de papéis

O modelo de papéis da ferramenta PROPOST é apresentado nas seguintes figuras: Figura 19, Figura 20, Figura 21 e Figura 22. O principal objetivo deste modelo é a atribuição das responsabilidades anteriormente identificadas no modelo de objetivos, a papéis a serem desempenhados na aplicação. O desempenho destes papéis envolve o uso e a produção de determinados conhecimentos, o atendimento de pré-condições e de pós-condições, e de habilidades específicas para esse fim. Pode haver casos em que uma responsabilidade seja abrangente demais, devendo ser repartida em duas ou mais atividades de mesma natureza que a original.

Definição dos papéis

Na modelagem de papéis da ferramenta PROPOST, e a partir das responsabilidades reutilizadas dos modelos de domínio da ONTOWUM e ONTOINFO foram selecionados os papéis, entidades externas, conhecimentos e condições correspondentes. Os elementos presentes nesses modelos que não apresentam variabilidades, e portanto constituem conceitos fixos, são reutilizados acompanhando o reuso das respectivas responsabilidades. Em relação às atividades e destrezas, as mesmas são reusadas conforme sejam mandatórias, alternativas ou opcionais. Por fim, acrescentamos elementos novos referentes ao domínio específico em questão, de acordo com as necessidades demandadas pela ferramenta.

Desta forma, quanto aos papéis existentes na ONTOWUM, a título de exemplo, visto que em relação aos demais conceitos de modelagem procedeu-se de maneira bastante similar, foram selecionados os seguintes, para serem representados no modelo: *monitor de usuário, modelador de usuário, classificador, personalizador, interface do usuário, aquisitor de dados e minerador de uso*. E, ao serem adaptados para o domínio em questão por meio da especialização, com exceção dos dois últimos, resultaram, nesta ordem, os papéis: *monitor de uso, modelador de área, classificador de áreas, recomendador de soluções e interface do usuário*.

Dentre os papéis existentes na ONTOINFO, foram selecionados e reusados dessa mesma forma os papéis de *modelador de consultas, modelador de soluções e recuperador de informações*.

Após a reutilização de todos os papéis acima citados, foram definidos exclusivamente para a PROPOST os papéis de *analisador de sistemas de informação, estimador de projetos, simulador de portifólio e analisador de projeto*. Todos estes papéis, conforme citado anteriormente, são relativos ao domínio da gestão de portifólio de projetos.

Reutilizações provenientes da ONTOWUM

Conforme citado anteriormente, o desempenho dos papéis está associado a responsabilidades, habilidades, pré e pós-condições. As responsabilidades e demais conceitos de modelagem a seguir foram selecionados na ONTOWUM visando a reutilização, sendo que alguns deles precisaram passar por adaptações antes da composição final.

Dessa forma, a primeira responsabilidade (Figura 19) definida foi a *monitoração de uso*. Na prática, esse papel corresponde ao monitoramento do uso do sistema pelos usuários, sendo de responsabilidade do papel *monitor de uso*. Sendo assim, este papel usa como conhecimento a interação do usuário com o sistema de informação a qual, conforme citado anteriormente, também representa a utilização das *soluções de software* pela área na qual o usuário está alocado. Também são consideradas as *sessões de uso* de *soluções de software* recuperadas para a referida *área*, as quais estão armazenadas no *repositório de uso de dados*. Essas informações levam à identificação do consumo de informações pela área, e a sua recuperação requer técnicas para captura de informações sobre utilizações de *soluções de software,* como por exemplo a captura implícita do perfil do usuário. A pré-condição para isto é que um usuário tenha entrado no sistema, e a pós-condição é que as informações sobre a utilização de soluções tenham sido capturadas.

A segunda responsabilidade (Figura 19), que é a *modelagem da área corrente*, atribuída ao papel *modelador de uso*, utiliza o conhecimento gerado pelo papel anterior, bem como tem a pós-condição deste como sua pré-condição, e produz o *modelo da área corrente*. Isso resulta em que o *modelo da área corrente* esteja disponível como pós-condição, através de técnicas para construção de modelos de usuário, como por exemplo, o modelo de matrizes de características (SHAHABI, BANAEI-KASHANI, 2003).

A terceira responsabilidade (Figura 19), consiste na *manutenção de dados de uso*, e tem como responsável o papel *aquisitor de dados*, o qual utiliza informações relativas ao uso do sistema pelo usuário para originar/manter um *repositório de dados de uso*. Para isso são requeridas técnicas para *manutenção de dados de uso*, como por exemplo, os grafos semânticos em RDF (LASSILA, SWICK, 1999), o que da margem à pós-condição de que o *repositório de dados de uso* esteja atualizado.

A quarta responsabilidade (Figura 19) é a *descoberta de padrões de consumo*, cujo encarregado é o papel *minerador de uso.* A mesma consiste em descobrir padrões de uso dos sistemas de informação pelas áreas gerenciais, e para isso utiliza as informações provenientes do *repositório de dados de uso*, tendo como pré-condição que estes dados estejam atualizados. Para a sua realização, são necessárias técnicas de descoberta de padrões de uso, como por exemplo, o algoritmo *K-Means* (SHAHABI, BANAEI-KASHANI, 2003), resultando como pós-condição que os *modelos de grupos de área* sejam disponibilizados.

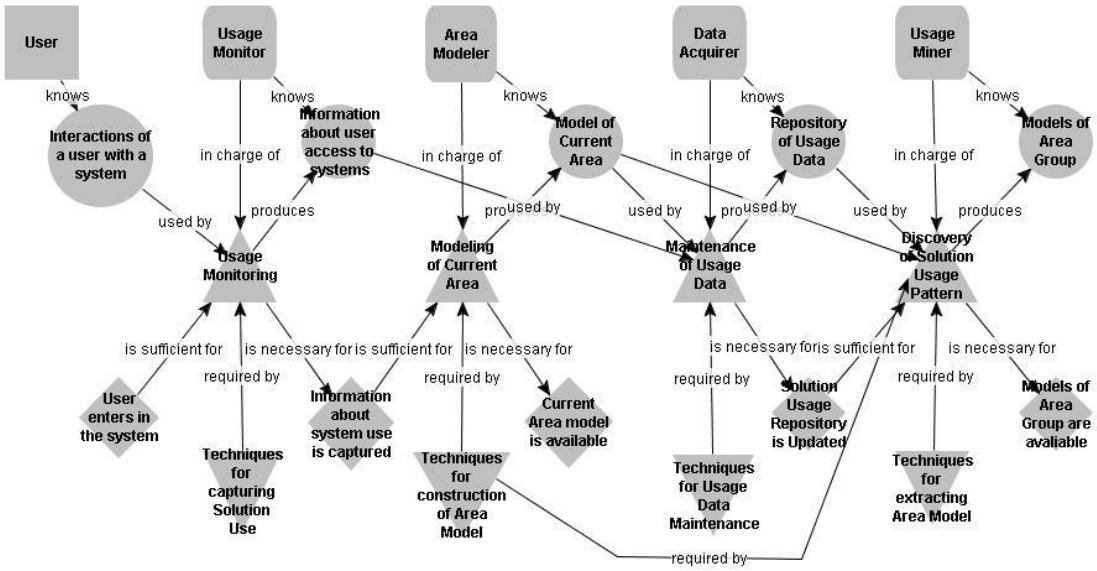


Figura 19 Modelo de Papeis da PROPOST Parte 01

A quinta responsabilidade (Figura 20), se refere à *classificação da área corrente*, e é incumbência do papel *classificador de área*, que identifica o grupo similar ao *perfil da área corrente* a partir do produto advindo do papel antecedente e do *modelo da área corrente*. Para isso são necessárias técnicas para classificação de áreas em grupos, como por exemplo, o algoritmo *K-Means* (SHAHABI, BANAEI-KASHANI, 2003). Também deve ser atendida a pré-condição de que o *modelo da área corrente* esteja disponível, dando lugar à pós-condição de que o *grupo com perfil similar* ao da *área corrente* esteja disponível.

A sexta responsabilidade (Figura 20), que diz respeito à *construção do modelo de recomendação*, tem o papel *recomendador de soluções de software* como encarregado, que produz o *modelo de recomendação* usando o produto resultante da responsabilidade anterior. Para isso, são necessárias técnicas para construção de *modelos de recomendação*, como por exemplo a filtragem colaborativa. A pré-condição para tanto equivale à pós-condição precedente, e tendo sua pós-condição que o *modelo de recomendação* esteja pronto.

A sétima responsabilidade (Figura 20), relativa à recomendação de *soluções de software*, é incumbência do papel *interface do usuário*, que usa os resultados da recomendação e produz *soluções de software* a serem recomendadas, tendo como pré-condição a pós-condição que a antecede e como pós-condição que as recomendações tenham sido entregues. Para isso são necessárias técnicas para entrega de informações personalizadas, como por exemplo a apresentação através de formulários em janelas.

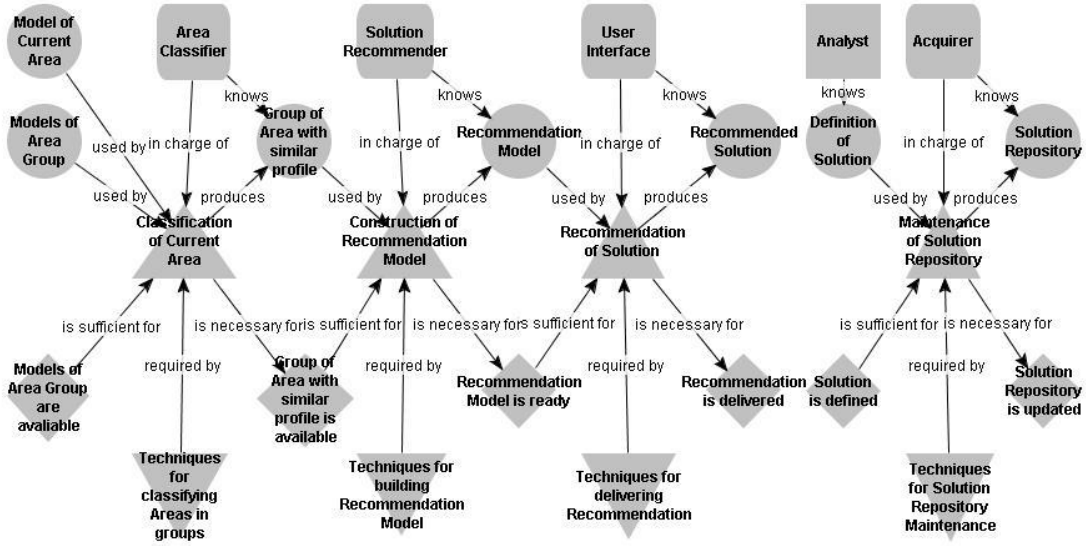


Figura 20 Modelo de Papeis da PROPOST Parte 02

Reutilizações provenientes da ONTOINFO

As responsabilidades e demais conceitos de modelagem a seguir foram selecionados na ONTOINFO visando à reutilização, sendo que alguns deles precisaram passar por adaptações antes da composição final.

Dessa forma, a oitava responsabilidade (Figura 21) relacionada no modelo é a representação da *consulta do usuário*, sendo encargo do papel *modelador de consultas*. Ele usa a necessidade de informação do analista, através da qual produz uma representação interna da mesma, sendo necessárias técnicas para a representação de consultas, como por exemplo a representação em um vetor de palavras-chave.

A pré-condição para esta responsabilidade ocorrer é que o analista defina a sua necessidade de informação, e a pós-condição é a consulta tendo sido representada.

A nona responsabilidade (Figura 21) é a *representação e indexação das soluções de software*, pela qual responde o papel *modelador* de soluções. Ele usa as informações contidas nas *soluções de software* (contidas no repositório de soluções), através das quais produz uma representação e indexação interna das mesmas, sendo para isso necessárias técnicas para a *representação e indexação,* como por exemplo os vetores de palavras-chave. A pré-condição para isto ocorra é que a informação contida na *solução de software* esteja disponível, e a pós-condição é que a representação da solução.

A décima responsabilidade (Figura 21) é a *análise de similaridade e recuperação de soluções de software*, tendo como responsável o papel *recuperador de informação*. Ele usa a *representação da consulta do analista* e a *representação da solução de software*, através das quais produz a recuperação da *solução de software* desejada. Para isso, são necessárias técnicas para a recuperação de informações, como por exemplo as técnicas baseadas no modelo espaço-vetorial (BAEZA, RIBEIRO NETO, 2000). A pré-condição para isto é que tanto a *consulta* como a *solução de software* estejam representadas, e a pós-condição é que a recuperação da informação solicitada.

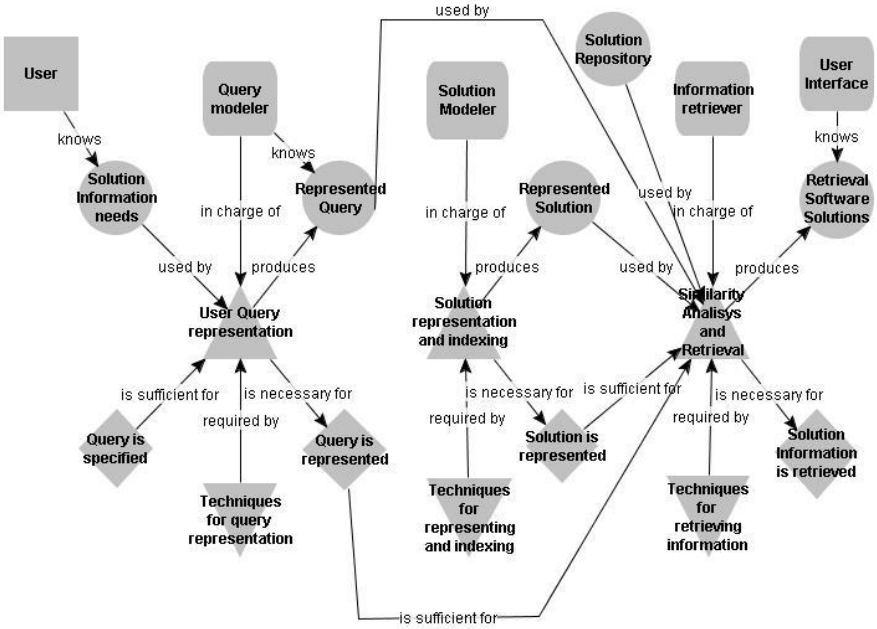


Figura 21 Modelo de Papeis da PROPOST Parte 03

Responsabilidades e demais conceitos novos

As responsabilidades e demais conceitos de modelagem a seguir foram criados especificamente para a ferramenta PROPOST, de acordo com as necessidades identificadas, sendo estas provenientes do domínio da área de gestão de portifólio de projetos.

Dessa forma, a décima primeira responsabilidade (Figura 22) é a *estimativa de projetos*, tendo como responsável o papel *estimador de projetos*. Ele usa a informação de *alocação dos recursos* e de *definição de soluções* para realizar este responsabilidade, para a qual são demandadas técnicas de estimativas de projetos, como por exemplo a análise de pontos de. A pré-condição para isto é que as soluções sejam definidas, tendo como pós condição a estimativa dos projetos.

A décima segunda responsabilidade (Figura 22) é a *priorização dos projetos*, tendo como responsável o papel *priorizador de projetos*. Ele usa a estimativa dos projetos, através das quais produz a priorização dos projetos a serem realizados. Para essa responsabilidade, são necessárias técnicas para a *priorização de projetos*. A pré-condição para isto é que a estimativa dos projetos tenham sido realizadas, e a pós-condição é que a priorização tenha sido realizada.

A décima terceira responsabilidade (Figura 22) é a *manutenção do repositório de recursos*, que é responsabilidade do papel *aquisitor de dados*. Ele utiliza informação sobre a *priorização dos projetos*, para atualizar as informações dos recursos. A pré-condição para isso é que os projetos sejam priorizados, tendo como pós-condição a atualização do *repositório de recursos*.

A décima quarta responsabilidade (Figura 22) é a *elaboração da análise do status dos projetos*, tendo como responsável o papel *analisador de projetos*. Ele usa as informações sobre os projetos, através das quais produz a *análise do status dos projetos*. Para isso, são necessárias técnicas para análise da situação atual de projetos A pré-condição para isto é que as informações sobre os projetos estejam disponíveis no repositório de dados dos projetos, e a pós-condição é que a realização da *análise do status dos projetos*.

A décima quinta responsabilidade (Figura 22) é a *elaboração da análise de uso das soluções de software*, tendo como responsável o papel *analisador de soluções de software*. Ele usa a informação de acesso aos sistemas de informação pelo usuário contidas no *repositório de uso de dados*, através da qual produz a análise de uso dos sistemas de informação. Para isso, são necessárias técnicas para a elaboração de análises de uso das *soluções de software* pelos usuários. A pré-condição para isto é que o a informação do uso esteja disponível no *repositório de uso de dados*, e a pós-condição é que a referida análise seja realizada.

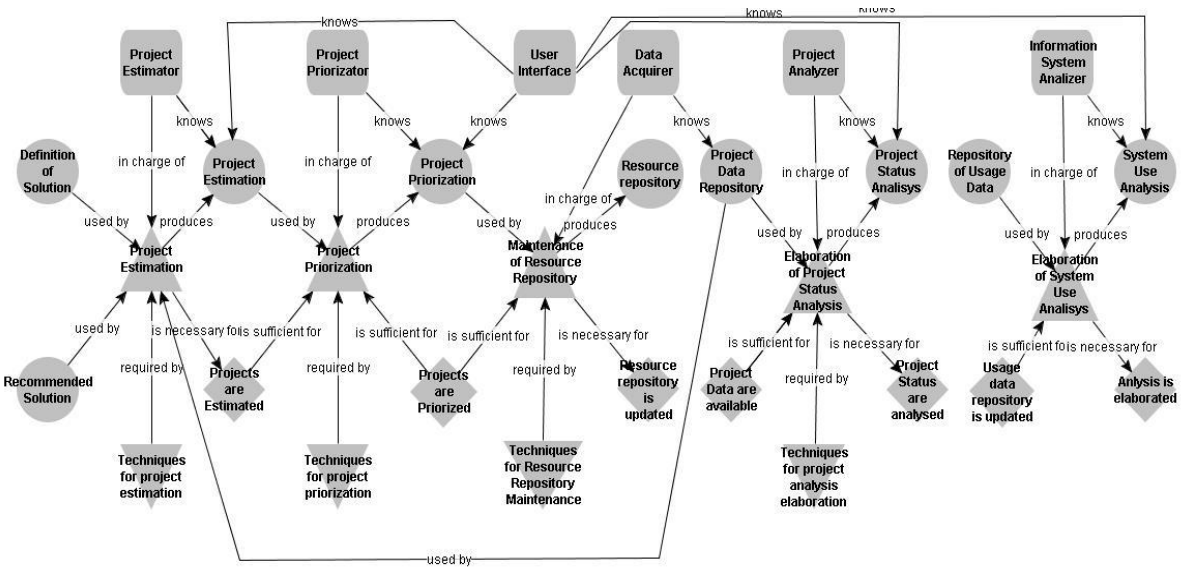


Figura 22 Modelo de Papeis da PROPOST Parte 04

4.3.6 Modelagem de interações entre papéis

Para cada objetivo específico definido no modelo de objetivos existe um modelo de interações entre papéis correspondente, que ilustra de maneira seqüencial como os papéis e as entidades externas da aplicação interagem, através da troca de mensagens, invocando responsabilidades ou atividades, bem como indicando que conhecimentos são passados como parâmetros.

Os modelos de interações entre papéis da ferramenta PROPOST são apresentados nas seguintes figuras: Figura 23, Figura 24, Figura 25, Figura 26 e Figura 27. Nestes modelos constam os objetivos específicos *modelar áreas, modelar recomendações de solução, prover informações sobre solução de software, prover suporte à priorização de projetos e prover suporte à avaliação de projetos*, respectivamente.

Reuso de interações provenientes da ONTOWUM

- **Interação referente ao objetivo específico Modelar Áreas gerenciais**

Esse modelo foi desenvolvido baseado na reutilização do modelo de interações, referente ao objetivo específico *modelar usuários através de mineração de uso*. Nesse modelo, todas as interações foram reutilizadas, tendo sido as mesmas devidamente adaptadas do modelo de domínio na ONTOWUM-DM para o domínio em referência. O referido modelo encontra-se representado na Figura 23, e suas respectivas interações são descritas a seguir.

A primeira interação reutilizada (Figura 23) é a invocação da responsabilidade *monitoração de uso* (interações do usuário com os sistemas de informação) ocorrendo entre o usuário e o papel *monitor de uso*. Em seguida, ocorre a segunda interação, que é a invocação da *modelagem da área corrente* (Figura 23), através da qual o papel de *modelador da área gerencial* recebe do papel *monitor de uso* as informações sobre as *sessões de uso*, necessárias à confecção do referido modelo, a ser usado posteriormente para as recomendações.

O próximo passo consiste na terceira interação (Figura 23), referente à invocação da responsabilidade *manutenção de dados de uso*, na qual o papel *modelador de área gerencial* envia o *modelo da área corrente* para o *aquisitor de dados*, que armazena tal modelos para oportuno emprego.

Por último, a quarta interação (Figura 23) é a invocação da responsabilidade *descoberta de padrões de consumo*. A mesma ocorre entre os papéis *aquisitor de dados* e *minerador de uso*, quando este pede àquele que forneça os *modelos de área* armazenados no *repositório de dados de uso*, para que seja realizada a mineração dos mesmos e conseqüente *descoberta de padrões de consumo*.

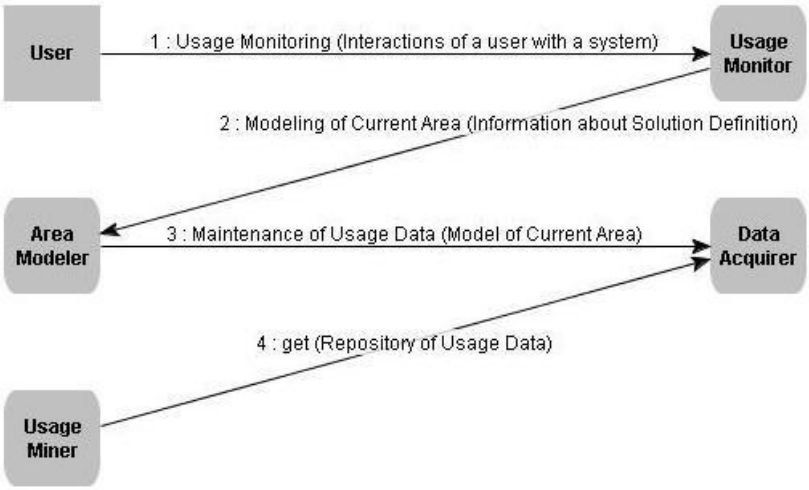


Figura 23 Modelo de Interações entre Papéis referente ao objetivo específico Modelar Área da PROPOST

- **Interação referente ao objetivo específico Modelar Recomendações de Solução de Software**

Em relação à modelagem das *recomendações de solução de software*, houve reuso de todas as interações presentes no correspondente modelo da ONTOWUM, tendo ocorrido a respectiva seleção, adaptação e composição necessárias. O referido modelo encontra-se representado na Figura 24, e suas interações são descritas a seguir.

A primeira interação (Figura 24) é a invocação da responsabilidade *classificação da área corrente*, ocorrendo entre o papel *modelador de área* e o *classificador de área*, onde o primeiro envia para o segundo o *modelo da área corrente*, para que seja realizada a referida classificação.

Em seguida, ocorre a segunda interação (Figura 24), na qual o papel *classificador de área* recebe do *minerador de uso* os *modelo de grupos de área*, e de posse do *modelo da área corrente* e dos *modelos de grupos de área,* o papel classificador de *áreas gerenciais* pode então realizar a a responsabilidade *classificação da área* corrente.

Logo após, ocorre a terceira interação (Figura 24), na qual o papel *recomendador de solução* recebe do papel *classificador de área* o *grupo de áreas com perfil similar* ao *perfil da área corrente*. De posse dessa informação, o *recomendador de solução* pode então realizar a responsabilidade *construção do modelo de recomendação*.

Na quarta interação, o papel *recomendador de solução* envia para o papel *interface do usuário* o *modelo de recomendação* (Figura 24), necessário para que seja realizada a referida recomendação. E por último, ocorre a quinta interação (Figura 24), na qual o papel *interface do usuário*, após receber o *modelo de recomendação de soluções*, providencia a entrega da referida *solução de software* para o usuário.

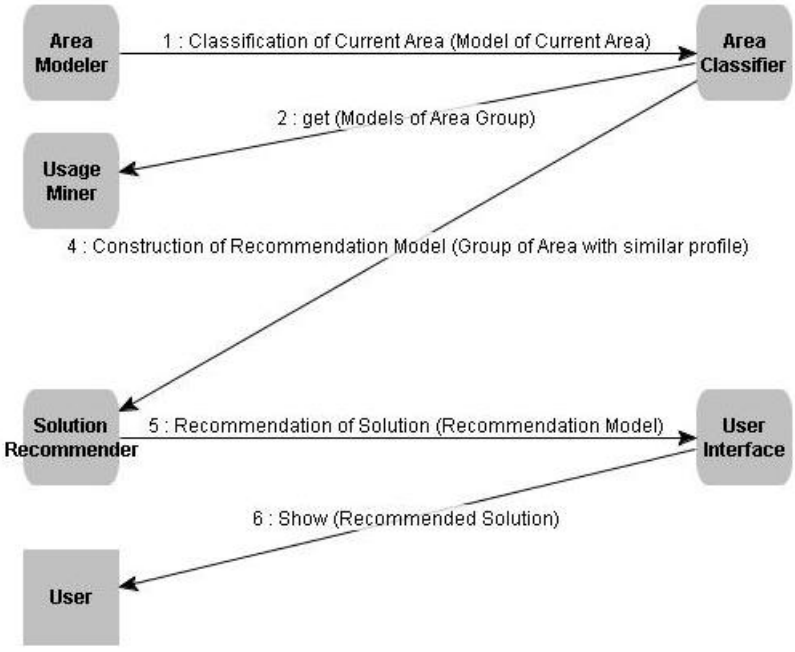


Figura 24 Modelo de Interações entre Papéis referente ao objetivo Modelar Recomendações de Solução da PROPOST

Reuso de interações provenientes da ONTOWUM

- **Interação referente ao objetivo específico Prover Informações de Solução**

Esse modelo foi desenvolvido baseado na reutilização do modelo de interações referentes ao objetivo específico *prover informações pontuais*. Nesse modelo, todas as interações foram reutilizadas, tendo sido as mesmas devidamente especificadas para o domínio em referência. O referido modelo encontra-se representado na Figura 25, e suas interações são descritas a seguir.

A primeira interação (Figura 25) ocorre quando o papel *analista* envia para o papel *modelador de consultas* a definição da sua *necessidade de informação*. De posse dessa informação, o *modelador de consultas* pode realizar a responsabilidade *representação interna da consulta*

Em seguida, a segunda interação (Figura 25) ocorre quando o *modelador de consultas* envia para o papel *recuperador de informação* a *representação da consulta*, necessária para recuperar a referida informação. Após receber essa informação, ocorre a terceira interação, quando o papel *recuperador de informação* solicita ao papel *modelador de solução* a representação interna das *soluções de software*, contidas no repositório de soluções. De posse dessas duas informações, o papel *recuperador de informações* pode recuperar as informações referentes às *soluções de software*, solicitadas.

A quarta interação (Figura 25) ocorre quando o papel *recuperador de informação* entrega ao papel *interface do usuário* as informações solicitadas. E por último, ocorre a quinta interação, quando o papel *interface com o usuário* exibe as referidas informações para o papel *analista*.

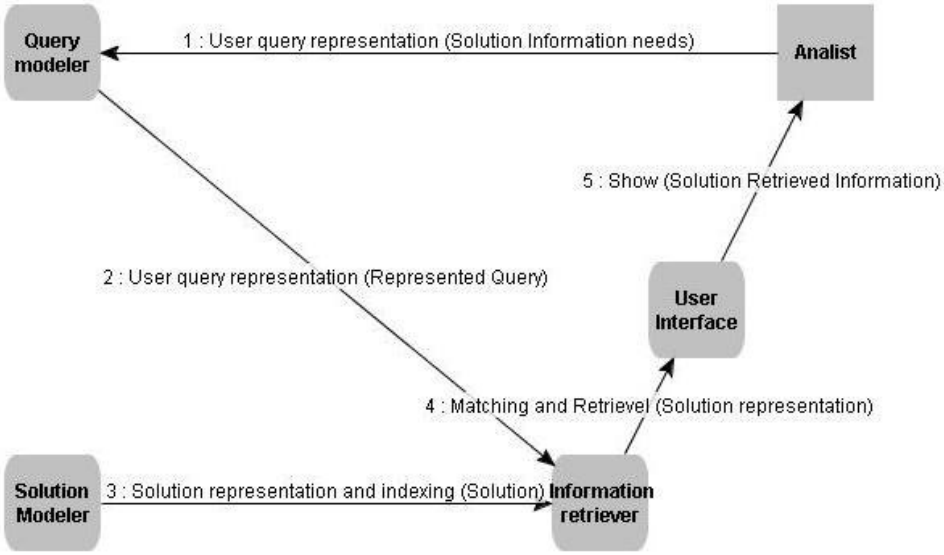


Figura 25 Modelo de Interações entre Papéis referente ao objetivo específico Prover Informação de Solução da PROPOST

Interações complementares

Após a realização da seleção, adaptação e composição de todos as interações possíveis de serem reutilizadas nas ontologias ONTOWUM e ONTOINFO, procedeu-se à criação das interações complementares, de acordo com as necessidades da ferramenta PROPOST, as quais estão abaixo relacionadas.

- **Interação referente ao objetivo específico Prover Suporte à Avaliação de Projetos.**

As interações referentes a esse modelo estão relacionadas principalmente às responsabilidades de elaboração da *análise do status dos projetos* e da *análise do uso dos sistemas de informação*, a partir das quais o analista recebe informações necessárias para a avaliação em questão. O referido modelo encontra-se representado na Figura 26, e suas respectivas interações são descritas a seguir.

A primeira interação (Figura 26) referente a esse modelo, ocorre quando o papel *analisador de projetos* recebe do papel *aquisitor de dados* as informações sobre os projetos, existentes no *repositório de dados* dos projetos. De posse dessas informações, ele realiza a responsabilidade de *análise de status dos projetos*. A partir daí, ocorre a segunda interação, quando o papel *analisador de projetos* envia para o papel *interface do usuário* a referida análise.

A terceira interação (Figura 26) ocorre quando o papel *analisador de soluções de software* recebe do papel *aquisitor de dados* as informações referentes aos usos das *soluções de software* contidas no *repositório de dados de uso*. De posse dessas informações ele realiza a responsabilidade de *análise de uso dos sistemas de informação*. Em seguida ocorre a quarta interação, quando o papel *analisador de informações de sistemas* envia a referida análise ao papel *interface do usuário*.

Por último, ocorre a quinta interação (Figura 26), quando o papel *interface do usuário* mostra para o papel *analista* a *análise do status dos projetos* e a *análise de uso dos sistemas de informação*.

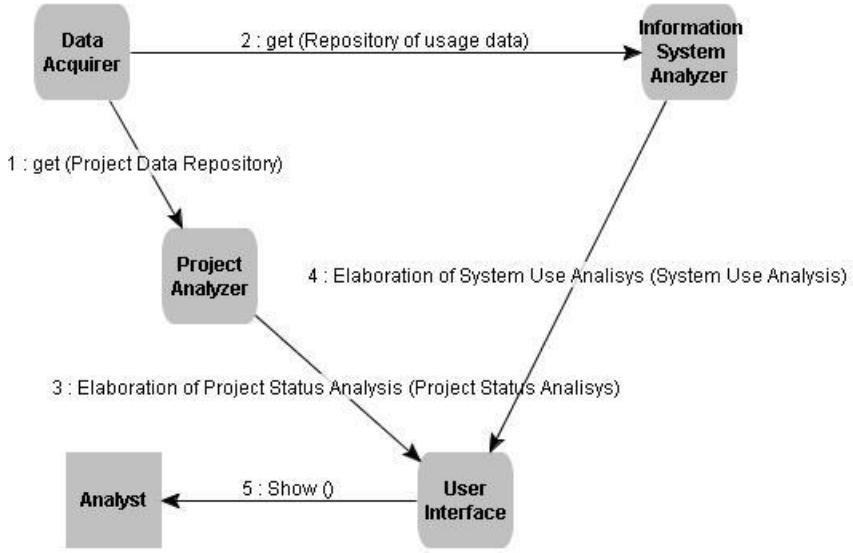


Figura 26 Modelo de Interações entre Papéis referente ao objetivo Prover Suporte na Avaliação de Projetos da PROPOST

- **Interação referente ao objetivo específico Prover Suporte à Priorização de Projetos**

As interações referentes a esse modelo dizem estão relacionadas principalmente às atividades de *elaboração das estimativas dos projetos*, *elaboração da priorização dos projetos* e *elaboração da simulação do portifólio de projetos* resultante das priorizações realizadas. O referido modelo encontra-se representado na Figura 27, e suas interações são descritas a seguir.

A primeira interação (Figura 27) referente a esse modelo, ocorre quando o papel *estimador de projetos* recebe do papel *aquisitor de dados* as informações sobre as *soluções de software* recomendadas para a *área gerencial*. De posse dessas informações, ele pode realizar as estimativas dos referidos projetos.

Em seguida, ocorre a segunda interação (Figura 27), quando o papel *priorizador de projetos* recebe do papel *estimador de projetos* as estimativas para os projetos correspondentes às *soluções de software* recomendadas. De posse dessas informações, ele realiza a priorização dos referidos projetos que irão compor o portifólio em questão.

E, por último, ocorre a sexta interação (Figura 27), quando o papel *interface do usuário* mostra exibe as informações sobre *estimativa de projetos* e *priorização de projetos* ao *analista.*

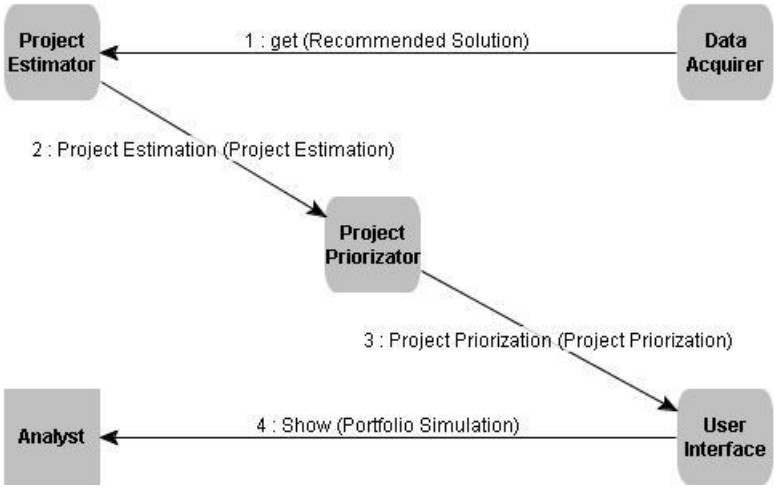


Figura 27 Modelo de Interações entre Papéis referente ao objetivo Prover Suporte na Priorização de Projetos da PROPOST

4.4 Projeto da Ferramenta PROPOST

Segundo a metodologia MAAEM, a fase de *Projeto da Aplicação* consiste de três subfases, que são: o *Projeto da Arquitetura*, que define a visão global da sociedade multiagente, sobretudo, passando pela escolha de seus mecanismos de cooperação e coordenação; o *Projeto do Agente*, que estabelece a visão detalhada e interna de cada agente, modelando sua estrutura e seu comportamento; e a *Modelagem do Conhecimento da Sociedade Multiagente*, que identifica os conceitos compartilhados por todos os agentes em sua comunicação. A tarefa principal do desenvolvedor é reusar soluções de projeto relativas a uma família de aplicações em um domínio, complementando-as com outras referentes apenas à aplicação em construção, gerando a arquitetura específica e seus respectivos detalhamentos que vão orientar a futura implementação. As soluções de projeto reusadas na PROPOST são relativas ao *framework* da ONTOWUM e ONTOINFO.

A subfase de Projeto da Arquitetura é composta das seguintes atividades:

- Modelagem do Conhecimento da Sociedade Multiagente
- Modelagem da Sociedade Multiagente
- Modelagem das Interações entre Agentes
- Modelagem de dos Mecanismos de Cooperação e Coordenação

A subfase de Projeto do Agente é composta das seguintes atividades:

- Modelagem do Conhecimento e das Atividades do Agente
- Modelagem dos Estados dos Agentes

4.4.1 Modelagem do Conhecimento da Sociedade Multiagente

A Figura 28 ilustra o *Modelo do Conhecimento da Sociedade Multiagente* da PROPOST, o qual constitui uma base para a construção da ontologia de comunicação dos agentes, reunindo os conceitos carregados nas mensagens trocadas entre eles.

No caso da *recomendação de soluções*, os agentes da aplicação irão se comunicar mediante a troca dos seguintes conhecimentos: *sessões de uso*, *modelo da área*, *repositório de dados de uso*, *grupo da área*, *modelo de recomendação*, *análise de uso, análise de status* e *repositório de soluções e repositório de dados de projetos.*

As *sessões de uso* dos sistemas pelo usuário indicam o *perfil de uso da área* – representado por um *modelo de área*, e fornecem subsídios para a geração de *recomendações de soluções de software* de interesse a serem reutilizadas, as quais são armazenadas em um *repositório de solução*. Um *repositório de dados de uso* mantém as *sessões de uso* das áreas, que são parte do *modelo da área*. Este modelo possui a *identificação da área*, sendo classificado em um *grupo de áreas*, o qual possui um *número de grupo* e uma *cardinalidade de grupo*.

Um *modelo de recomendação* é baseado em um *grupo de área* e indica um conjunto de *soluções de software* que são recomendáveis à *área gerencial*. Tais soluções possuem seu devido *identificador* e *descrição*. Por fim, *análises de status de projetos* e *análises de uso de sistemas* utilizam informações do *repositório de dados de projeto* e do *repositório de dados de uso*, respectivamente para serem realizadas.

Após a conclusão de todas as subfases e todos os passos da fase de projeto da aplicação, chega-se à implementação, em que haverá o mapeamento dos elementos aqui modelados para a elaboração de modelos relativos a uma tecnologia particular.

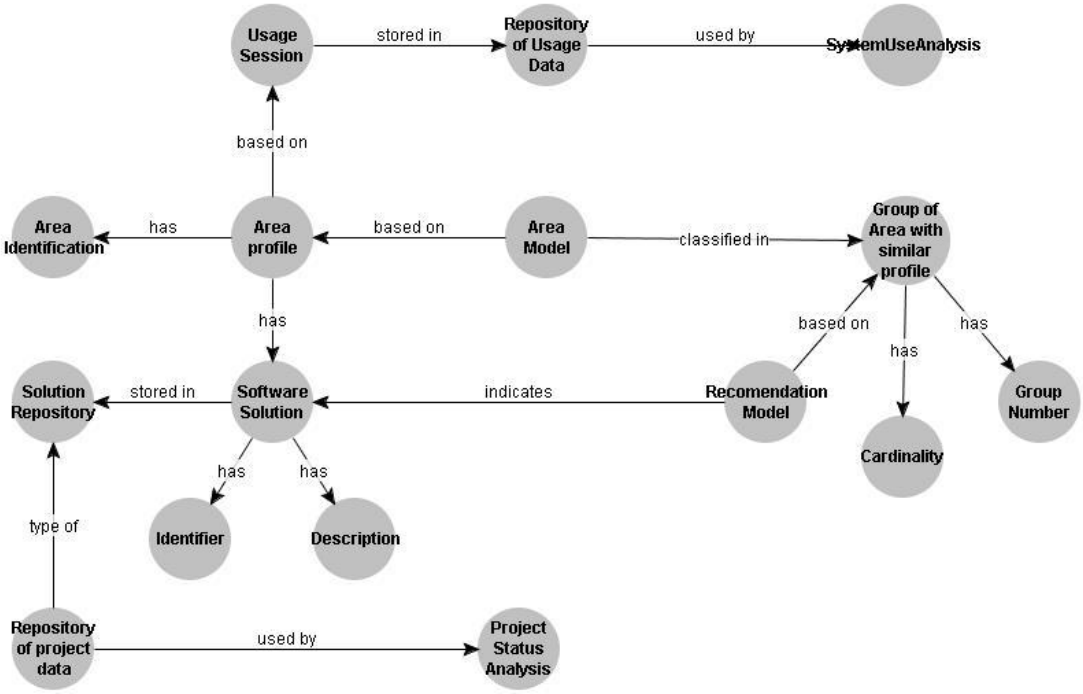


Figura 28 Modelo do Conhecimento da Sociedade Multiagente da PROPOST

4.4.2 Modelagem da Sociedade Multiagente

Nessa atividade, o ponto de partida são os papéis identificados na *Especificação da Aplicação*, que servem como critério para seleção de agentes nos *frameworks* multiagente reutilizados, os quais devem contemplar o problema abordado. O produto desse passo é o *Modelo da Sociedade Multiagente*, representado graficamente por um diagrama organizacional de três níveis.

Assim, feita a seleção de agentes que casem com os papéis previamente reutilizados, a adaptação consistirá em refazer o mapeamento dos agentes em face das responsabilidades, já que os papéis estabeleciam relações de um para um, mas agora se pode ter um daqueles desempenhando mais de um destes, pois regras de coesão funcional, semelhança de responsabilidades ou interações excessivas podem indicar a conveniência em se fundir vários papéis em um único agente. A Tabela 3 ilustra os mapeamentos de papéis para os respectivos agentes na PROPOST.

Papel	**Agente**
monitor de uso	*interfaceador*
modelador de área	*modelador*
aquisitor de dados	*aquisitor*
minerador de uso	*minerador*
classificador de área	
recomendador de soluções	*interfaceador*
interfaceador com o usuário	
modelador de consulta	
modelador de solução	
recuperador de informação	
analisador de sistemas	*analisador*
estimador de projetos	
priorizador de projetos	
analisador de projetos	

Tabela 3 Mapeamento de papéis para agentes na PROPOST

Quanto ao reuso dos demais elementos associados aos agentes, tais como conhecimentos, condições e destrezas, nenhuma grande empreitada é requerida, senão por estas últimas, que lá na fase de análise apenas comportavam possíveis técnicas aptas a auxiliar no cumprimento da responsabilidade ou atividade correspondente e aqui na fase de projeto são tratados com mais cuidado, sendo caracterizadas em geral por um algoritmo, procedimento ou tecnologia própria que tenha sido escolhida.

Os modelos da Sociedade Multiagente da PROPOST são apresentados nas seguintes figuras a seguir: Figura 29, Figura 30, Figura 31 e Figura 32. A Figura 29 e Figura 30 apresentam as reutilizações provenientes do modelo correspondente na ONTOWUM; a Figura 31 apresenta as reutilizações provenientes do modelo da ONTOINFO e ONTOWUM; e a Figura 32 apresenta os elementos correspondentes aos novos conceitos adicionados à modelagem da PROPOST, relevantes ao domínio da gestão de portifólio.

No projeto, para cada responsabilidade ou atividade, é selecionada uma destreza específica dentre aquelas que foram levantadas genericamente na fase de análise. Assim, em vez de várias técnicas que se prestam em potencial ao fim esperado, é definido um modo particular de se realizar a ação atribuída ao agente.

Com isso, para a monitoração de *áreas gerenciais* pelo agente *interfaceador* (Figura 29), foi adotada a captura implícita de usos de sistemas pelos usuários. Esta captura é feita no momento em que o usuário interage com o sistema de informação. Tendo em vista que cada usuário pertence a somente uma *área gerencial*, o uso de sistemas de informação pelos usuários pode conseqüentemente indicar o uso de sistemas pelas *áreas gerenciais* nas quais o usuário está alocado, representando o *comportamento de consumo* da referida área.

Com base nesse *comportamento de consumo*, o agente *modelador* poderá realizar a *modelagem da área corrente* (Figura 29). A técnica escolhida para representar os *modelos de área* foi o modelo de *matrizes de características* (SHAHABI, BANAEI-KASHANI, 2003). Neste modelo, características indicam as informações contidas numa *sessão de uso* da área. Para este caso, as características consideradas são o sistema usado e a freqüência de uso do mesmo pelo usuário. Assim, para cada característica da *sessão de uso* é criada uma matriz, cujo conjunto, constitui o *modelo da área corrente*.

Para a *manutenção de dados de uso* (Figura 29) realizada pelo agente *aquisitor*, são usados *grafos semânticos* em RDF (LASSILA, SWICK, 1999). Quando uma *sessão de uso* é encerrada, o agente *aquisitor* atribui a ela um identificador e a grava em um arquivo RDF. Este formato de arquivo é usado de forma a facilitar a análise desses dados no futuro.

A *descoberta de padrões de consumo* (Figura 29) realizada pelo agente *minerador*, envolve a extração de *modelos da área* e a aplicação de algoritmos de mineração. Para a extração de *modelos de área*, escolheu-se novamente o modelo de *matrizes de características*. Deste modo, os *modelos de área* são dados pelas *matrizes de características* que são extraídas do arquivo RDF gravado anteriormente. Já para a aplicação de algoritmos de mineração, foi escolhido o uso do algoritmo de agrupamento *K-Means* (SHAHABI, BANAEI-KASHANI, 2003).

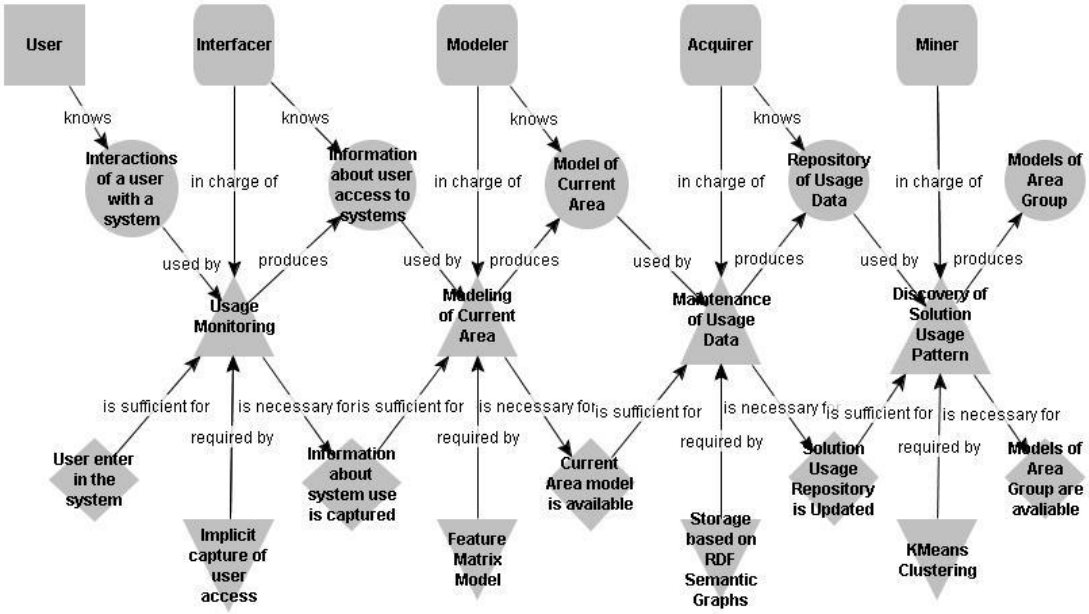


Figura 29 Modelo da Sociedade Multiagente da PROPOST Parte 1

Para a *classificação da área corrente* (Figura 30), realizada pelo agente *minerador*, foi também utilizado o algoritmo *K-Means* (SHAHABI, BANAEI-KASHANI, 2003). Nessa responsabilidade a *área gerencial* é classificada em um dos *modelos de grupo de área similar*, e tal classificação servirá de base para posterior geração da recomendação.

Para a *construção do modelo de recomendação* (Figura 30), realizada pelo agente *modelador* optou-se por utilizar a abordagem da *filtragem colaborativa* (HERLOCKER et al., 2004), de forma a se produzir uma lista de *soluções de software* relativa, neste caso, a sistemas de informação existentes que serão recomendados para potencial uso da *área gerencial* corrente, tomando por base a utilização de sistemas de informação por outras áreas que estão no grupo do qual a *área corrente* faz parte.

Na seqüência, para a *recomendação da solução* (Figura 30), a ser realizada pelo agente interfaceador, foi escolhida a apresentação de uma listagem de sistemas de informação (*soluções de software*) existentes em uma janela de formulário a ser exibida na interface da ferramenta PROPOST. Partindo-se do *modelo de recomendação* anteriormente gerado, os sistemas de informação mais recomendados são exibidos em uma janela assim que o analista requisitar a referida recomendação.

Para a manutenção do repositório de soluções (Figura 30), realizada pelo agente aquisitor são usados *grafos semânticos* em RDF (LASSILA, SWICK, 1999). Quando uma *solução de software* é definida, o agente *aquisitor* atribui a ela um identificador e a grava em um arquivo RDF. Este formato de arquivo é usado de forma a facilitar a análise desses dados no futuro.

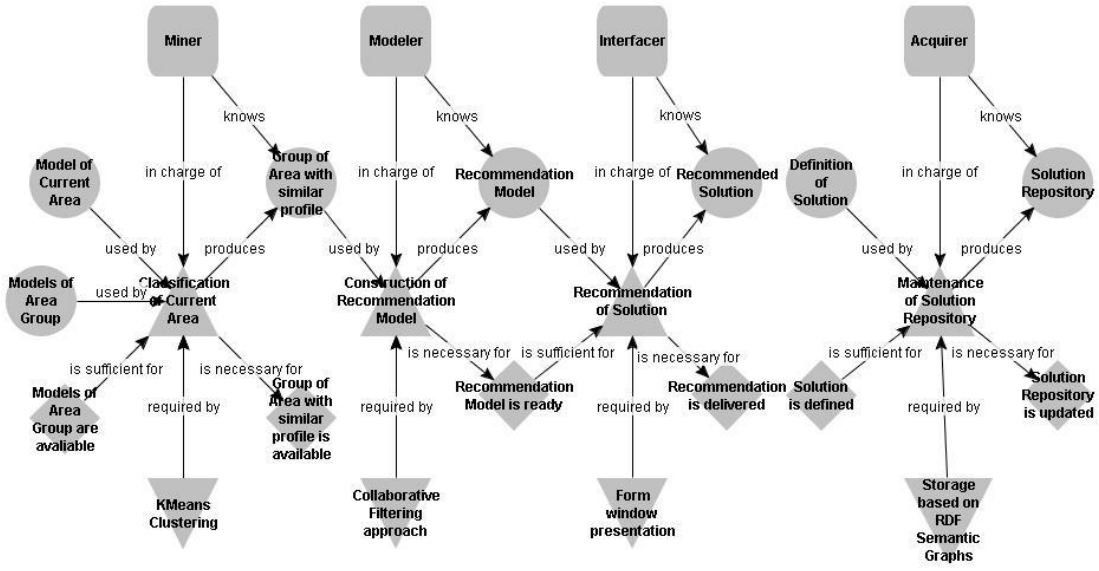


Figura 30 Modelo da Sociedade Multiagente da PROPOST Parte 2

A *representação da consulta* (Figura 31), realizada pelo agente *interfaceador* é realizada através de um vetor de palavras-chave (*vetor da consulta*) (BAEZA, RIBEIRO NETO,2000). A representação das informações contidas nas soluções de software, igualmente realizada pelo agente *interfaceador*, também é realizada através de vetores de palavras-chave (*vetores de soluções*).

Para a *análise de similaridade e recuperação* das informações relacionadas às *soluções de software* (Figura 31), realizada pelo agente *interface*ador, a ferramenta utiliza técnicas de Recuperação da Informação baseadas no modelo *espaço-vetorial* (BAEZA, RIBEIRO NETO,2000). Nesse modelo os *vetores de soluções* são comparados com o *vetor da consulta* e, através do cálculo e comparação da similaridade entre estes vetores é possível representar graficamente a relação de similaridade entre os mesmos, para a posterior obtenção das melhores respostas que possam atender às necessidades de informação do usuário.

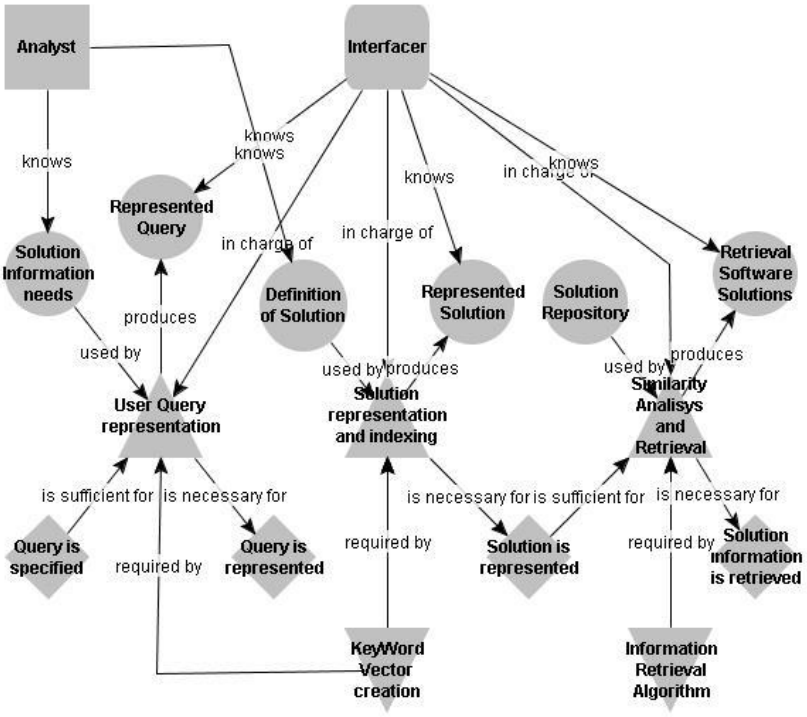


Figura 31 Modelo da Sociedade Multiagente da PROPOST Parte 3

Para a *estimativa de projetos* (Figura 32), realizada pelo agente *analisador*, foram utilizadas técnicas de Pontos de Função (PRESSMAN, 2001), devido a ser estas as mais largamente utilizadas para esta responsabilidade.

Para a *priorização de projetos* (Figura 32), realizada pelo agente *analisador*, foi utilizado um modelo de *scores*, o qual foi especialmente elaborado para este trabalho. Nesse modelo o usuário pode definir critérios de pontuação e definir pesos para estes critérios, com a flexibilidade que os mesmos podem ser genéricos ou específicos para determinadas soluções de software. Baseado nessas informações, o agente analisador realiza o cálculo final das priorizações através do somatório da média ponderada entre as pontuações de cada solução nos critérios estabelecidos e o peso dos mesmos, resultando em uma nota final para cada solução, a qual servirá para definir a sua priorização.

Após a priorização dos projetos (Figura 32), o agente *aquisitor* pode realizar a responsabilidade de *manutenção do repositório de recursos*, na qual são usados *grafos semânticos* em RDF (LASSILA, SWICK, 1999), e é realizada a atualização das informações de quais recursos foram alocados em quais projetos.

Para a *análise do status dos projetos* (Figura 32), realizada pelo agente *analisador*, foi considerada a técnica do *Earned Value* (PMBOK, 2004). Esta é a técnica mais largamente utilizada na atualidade no ambiente de gestão de projetos, em relação à medição da performance dos mesmos, e integra informações de prazo, custo e escopo para gerar indicadores relativos à evolução do projeto. A mesma requer que as medidas de desembolso e desempenho do projeto sejam estabelecidas dentro de um cronograma físico, ou seja, o resultado da medição é estabelecido integrando-se as variáveis de custo e prazo, proporcionando uma análise mais precisa do que se estes controles fossem realizados e analisados separadamente, como é comum em muitos projetos onde esta técnica não é utilizada. Essas informações são obtidas através do *repositório de dados dos projetos*.

Para a *análise do status de utilização dos sistemas de informação* (Figura 32), realizada pelo agente analisador, a ferramenta exibe uma listagem com todos os sistemas existentes, contendo para cada sistema indicadores relativos ao seu percentual atual de uso, disponibilidade do sistema na rede e nível de aderência da arquitetura técnica (*hardware e software*) do sistema em relação à plataforma alvo adotada pela empresa. Dessa forma, a referida análise consiste na comparação destes valores atuais com valores correspondentes previamente estabelecidos, os quais refletem a expectativa da empresa em relação aos sistemas.

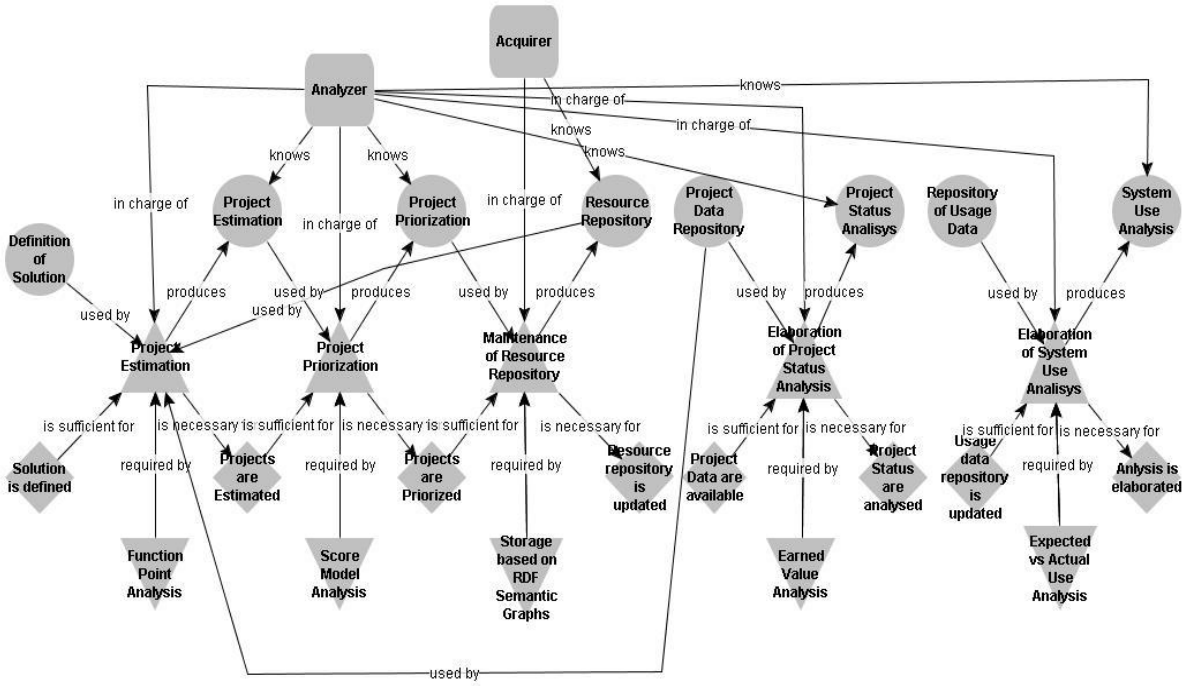


Figura 32 Modelo da Sociedade Multiagente da PROPOST Parte 4

4.4.3 Modelagem das Interações entre Agentes

O produto dessa *subtarefa* é um modelo de interações que especifica as interações que ocorrem entre os agentes do *framework*. As mensagens são rotuladas com performativas da linguagem de comunicação entre agentes FIPA-ACL (http://www.fipa.org). A FIPA-ACL é um padrão internacional para a interoperabilidade entre os agentes, esse padrão define vários atributos, em particular:

- O *remetente* da mensagem;
- A lista de *destinatários*;
- A intenção da comunicação (também chamada de "performativa") que indica a intenção do remetente ao enviar a mensagem. A performativa pode ser REQUEST, se o remetente deseja que o destinatário execute uma ação, INFORM, se o remetente quer informar o destinatário sobre um fato, QUERY_IF, se o remetente deseja saber se determinada condição é verdadeira, CFP, ("*call for proposal*"), PROPOSE, ACCEPT_PROPOSAL, REJECT_PROPOSAL, se o remetente e o destinatário estão engajados em alguma negociação;
- O *conteúdo*, que é a informação incluída na mensagem (ex: a ação a ser executada em uma mensagem REQUEST, ou o fato a ser informado na mensagem INFORM);
- A *ontologia*, que é o vocabulário usado no conteúdo da mensagem indicando o seu significado (tanto o remetente quanto o destinatário devem atribuir o mesmo significado às mensagens trocadas para que a comunicação possa ser efetiva);

Por último, existem alguns atributos usados para o controle de conversas concorrentes entre os agentes e limites de tempo para o recebimento de respostas tais como *conversation-id*, *reply-with*, *in-reply-to* e *reply-by*.

A Figura 33 mostra o *Modelo de Interações entre Agentes* da PROPOST, no qual os agentes e as entidades externas interagem ao longo de suas linhas de vida, seja os primeiros trocando mensagens entre si, ou estes sendo notificados de eventos externos pelas segundos.

Assim, inicialmente ocorre a notificação de que um "*usuário entrou no sistema*", feita ao agente *interfaceador*, sendo essa interação que permite o início do funcionamento da aplicação em benefício dele, posto que é a partir dos usos de sistemas por usuários de uma determinada área que pode-se obter o modelo da mesma e, a partir daí, prosseguir às recomendações.

Em seguida, uma mensagem "*INFORM_REF (informações sobre os usos de sistemas)*" é passada do agente *interfaceador* para o *modelador*. Tendo em vista que cada usuário está lotado em apenas uma *área*, é possível construir o *modelo da área corrente*, o qual será usado para classificação desta área. Posteriormente, caso hajam outras áreas modeladas, a aplicação já poderá *classificar a área corrente*, o que faz após o agente *minerador* receber uma mensagem "*INFORM-REF (modelo da área corrente)*" do *modelador*.

Logo após, o agente *minerador* requer ao agente *aquisitor*, através de uma mensagem "*QUERY_REF (repositório de dados de uso)*", os dados de uso a partir dos quais são extraídos os *modelos da área* para formação de *grupos de área*, nos quais será classifica a *área corrente*.

Como resultado, o agente *minerador* remete ao *modelador* uma mensagem "*INFORM_REF (grupo de áreas com perfil similar à área corrente)*" contendo o grupo em que se enquadra a *área corrente*, na qual é baseado o processo de *recomendação de soluções de software*.

O próximo passo consiste no *modelador* repassar ao *interfaceador* a mensagem "*INFORM_REF (modelo de recomendação)*" para que ele possa providenciar, com base em tal modelo, a efetiva oferta de *soluções de software* recomendadas à área solicitante, recebendo em seguida uma confirmação de que as "*recomendações de solução de software foram entregues*", indicando que a operação foi bem sucedida.

Não obstante aos eventos acima citados, pode também estar ocorrendo em paralelo, a notificação de que um "*usuário saiu do sistema*", igualmente feita ao agente *Interfaceador*, originando outra mensagem "*INFORM_REF (informações sobre usos de sistemas)*" dele para o *modelador* e, por fim, uma "*INFORM-REF (modelo da área corrente)*" ao *aquisitor*, o que possibilita o armazenamento dos dados da presente *sessão de uso*, os quais, juntamente com outras sessões passadas e futuras, constituem outra parte do subsídio demandado pela classificação.

Outra interação que poderá estar ocorrendo entre o usuário e o sistema será a busca por determinadas *soluções de software*, a partir de palavras-chaves contidas na descrição das mesmas. Neste caso, ocorre uma necessidade de informação pontual, cuja especificação é realizada pelo usuário via interface e, consequentemente, reconhecida pelo agente *interfaceador*. Este mesmo agente se encarrega de enviar ao *aquisitor*, através de uma mensagem "*QUERY_REF (representação da consulta)*", na qual ele recupera as *soluções de software* relevantes à necessidade de informação e as entrega ao usuário.

Por último, podem ocorrer ainda as interações referentes às informações sobre *análise de status dos projetos* e *análise de uso de sistemas de informação*, as quais são carregadas automaticamente pelo sistema sempre que um usuário acessa o mesmo. Essas análises são realizadas baseadas em requisições do agente *analisador* para o agente *aquisitor*, respectivamente enviando as mensagens "QUERY_REF (busca no *repositório de dados de uso*)" e "QUERY_REF (busca *no repositório de dados de projetos*)".

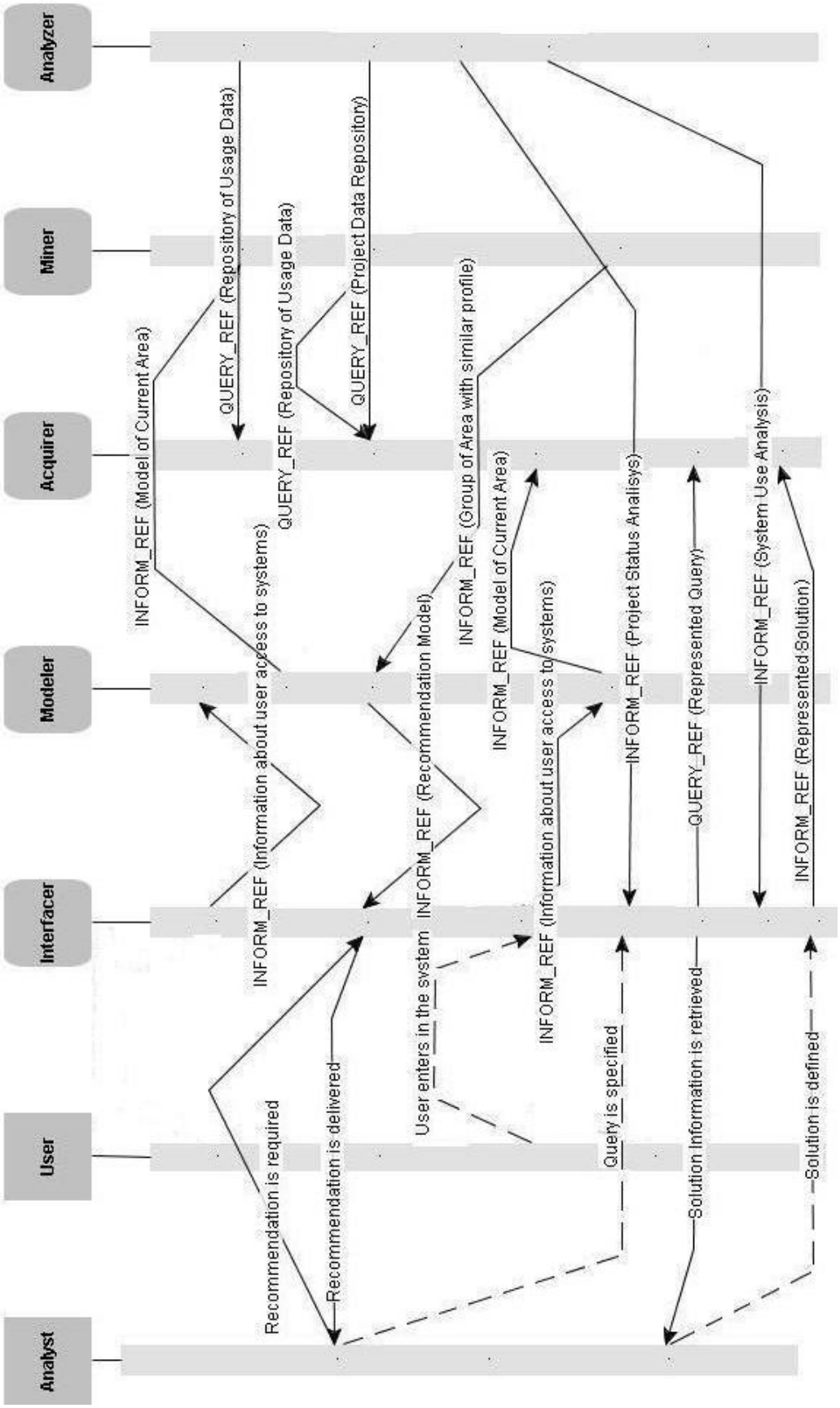


Figura 33 Modelo das Interações entre Agentes

4.4.4 Modelagem dos Mecanismos de Cooperação e Coordenação

Partindo do *Modelo da Sociedade Multiagente* e do *Modelo das Interações entre Agentes*, considerados até então apenas como um esboço arquitetural da aplicação, e segundo requisitos funcionais e não-funcionais previamente analisados, deve-se agora organizá-la através da seleção e adaptação de mecanismos de cooperação e coordenação entre agentes (HUHNS, STEPHENS,1999) (JENNINGS, 1993). O produto desse passo é o *Modelo dos Mecanismos de Cooperação e Coordenação*, representando a arquitetura adotada.

De acordo com a solução definida, a adaptação decorrente de sua adoção pode resultar no agrupamento – em camadas ou federações, por exemplo – de agentes, o que exigirá que se adaptem também os outros elementos de modelagem associados. Também se escolhe um protocolo de comunicação de agentes – como o FIPA-ACL – para a descrição das interações.

A Figura 34 exibe o *Modelo dos Mecanismos de Cooperação e Coordenação da PROPOST*, que segue o padrão arquitetural em camadas para sistemas multiagente denominado *Multi-agent Layer* (GIRARDI et al., 2005b), exatamente como foi feito na modelagem dos mecanismos de cooperação e coordenação da ONTOWUM, reutilizada por esta aplicação.

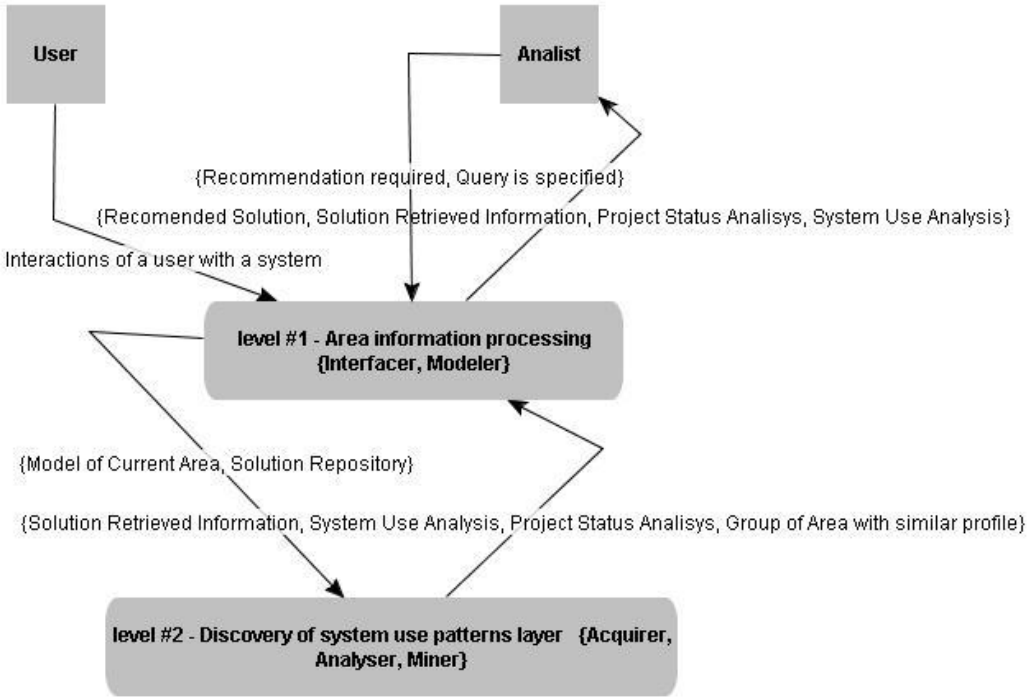


Figura 34 Modelo dos Mecanismos de Cooperação e Coordenação da PROPOST

No nível de abstração “1”, encontra-se a camada de *processamento de informações da área*, da qual fazem parte os agentes i*nterfaceador e modelador.* Esta camada faz a interface com o usuário e analista, monitora o uso de sistemas de informação pelas áreas e envia informações para a camada inferior referente aos *modelos de área* e acessos de pesquisa ao *repositório de soluções*.

No nível de abstração “2”, situa-se a camada de *descoberta de padrões de uso de sistemas de informação,* participando da mesma os agentes: *analisador*, *aquisitor e minerador*. Esses agentes concentram as atividades de processar *recomendações de soluções de software, informações sobre as soluções, análise de status de projetos e análise de uso de sistemas* , utilizando informações provenientes da camada superior. Após realizar as referidas atividades, esta camada envia essas informações processadas à camada superior, para sua conseqüente exibição ao usuário e/ou analista.

Interação entre agentes e camadas no contexto da recomendação

A PROPOST seguiu o mesmo modelo de camadas da ONTOWUM. A ONTOMUW se baseou na arquitetura genérica definida por Mobasher et al. (2000) para os sistemas de personalização na *Web* baseados na MUW. Dessa forma, na PROPOST existe uma camada responsável pelo monitoramento, modelagem da área corrente e execução da recomendação e outra camada responsável pelo pré-processamento dos dados de uso e descoberta de padrões. A camada de descoberta de padrões fornece serviços para a primeira camada, que são justamente os padrões descobertos, necessários para a realização da recomendação.

A divisão dos agentes nas camadas se dá de acordo com a similaridade entre suas responsabilidades. Sendo assim, similarmente à ONTOWUM, na PROPOST tem-se duas camadas, uma chamada de camada de processamento de informações da área, a qual contém os agentes Interfaceador e Modelador, e outra chamada de camada de descoberta de padrões de navegação, a qual contém os agentes Aquisitor, Minerador e Analisador. A Figura 35 ilustra essas camadas.

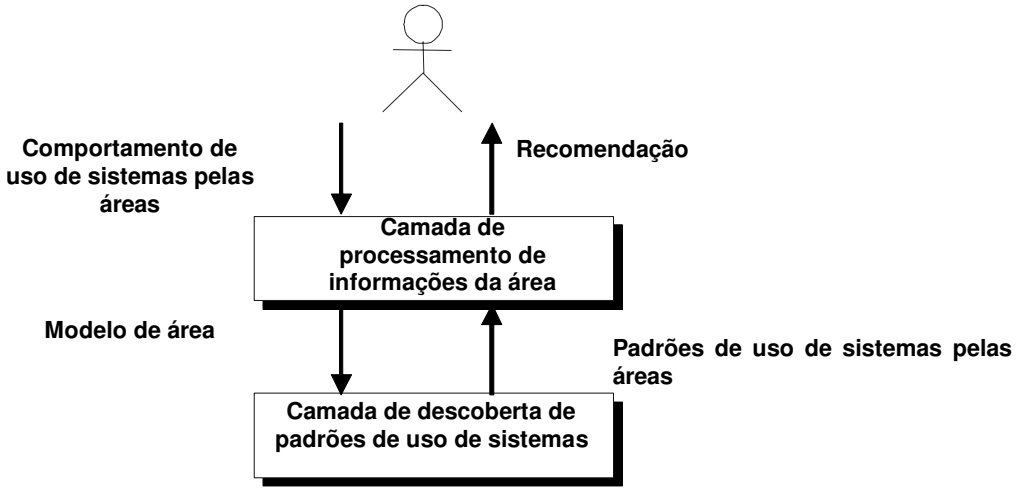


Figura 35 Estrutura de camadas da arquitetura da PROPOST

No contexto da Recomendação de Soluções de Software, a interação dos agentes nessas camadas pode ocorrer em dois eventos básicos, mostrados na Figura 36 .

a) Evento 1

O agente *interfaceador* roda todo final de dia, e coleta dados sobre sistemas de informação acessados pelas *áreas gerenciais*. Em seguida ele envia estes dados para o agente *modelador* construir os *modelos de área*.

Após construir os *modelos de área*, o agente *modelador* os envia para o *aquisitor* armazenar os mesmos em um *repositório de dados de uso*. Ao armazenar estes dados, na prática, o *aquisitor* pode tanto realizar uma atualização nos modelos já existentes como também incluir novos modelos, dependendo da ocorrência em questão.

b) Evento 2

O usuário acessa a ferramenta, cuja interface lhe é apresentada pelo agente *interfaceador*. Após isso, o usuário solicita recomendação para uma determinada *área* de sua escolha (*área corrente*). De posse da identificação da área, o agente *interfaceador* faz uma atualização da coleta dos dados de acesso recentemente realizados pela área (ocorridos desde a última coleta até o momento atual), e em seguida, o agente *interfaceador* envia os dados recentemente coletados para o *modelador* construir o *modelo da área corrente*.

O próximo passo deste evento consiste no agente *modelador* enviar a identificação da *área corrente* para o *minerador.* De posse deste modelo, o agente *minerador* realiza dinamicamente a *descoberta de padrões de uso*, necessária para a construção de *grupos de áreas*, através da mineração em relação aos *modelos de área* existentes no repositório de dados de uso. Em seguida, o *minerador classifica a área corrente* em um dos *grupos de área* construídos*,* o qual terá perfil semelhante ao seu.

Após classificar a *área corrente*, o agente *minerador* envia o *modelo de grupo de áreas semelhantes* para agente *modelador*, para que este possa fazer o *modelo de recomendação*. E após fazer o *modelo de recomendação*, o agente *modelador* envia este modelo ao agente *interfaceador*, o qual exibe as recomendações ao usuário.

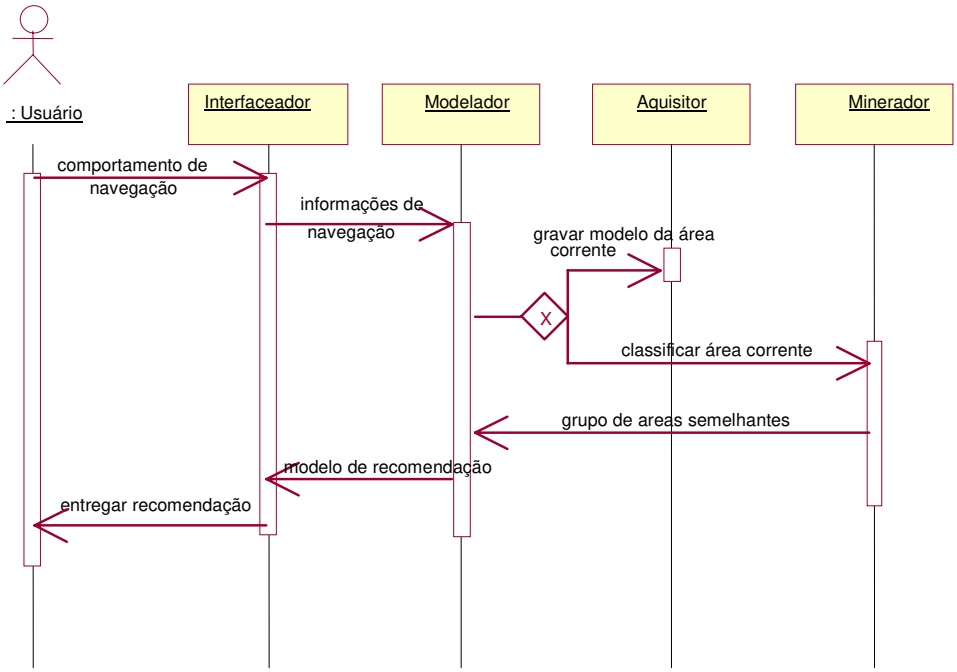


Figura 36 Interações entre agentes para recomendação da PROPOST

4.4.5 Modelagem do Conhecimento e das Atividades dos Agentes

a) Modelagem do conhecimento e das atividades do agente Aquisitor

O agente *aquisitor* possui as seguintes responsabilidades: *manutenção do repositório de recursos, manutenção do repositório de soluções de software e manutenção do repositório de dados de uso*.

A manutenção do *repositório de recursos* (Figura 37) é realizada mediante a pré-condição de que os projetos tenham sido priorizados, pois nesse momento os recursos são alocados aos projetos. A partir daí é possível originar e manter um *repositório de recursos*, com base em técnicas para manutenção de dados, o que da margem à pós-condição de que o repositório esteja atualizado. Para a manutenção do *repositório de recursos*, são usados grafos semânticos em RDF (LASSILA, SWICK, 1999).

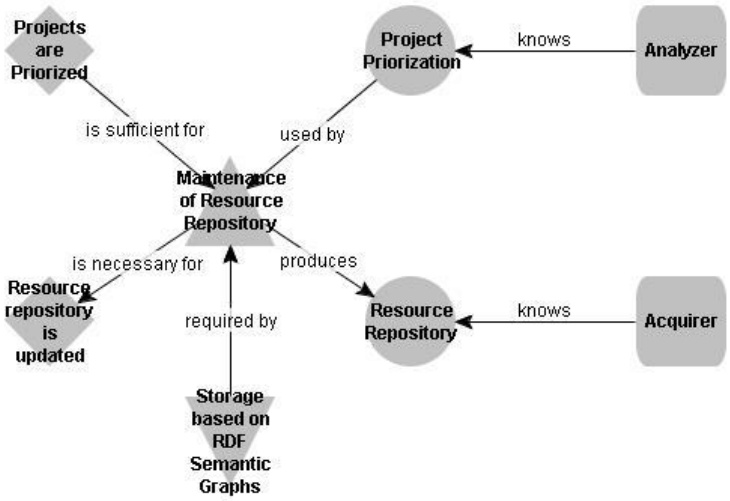


Figura 37 Modelo do Conhecimento e das Atividades do Agente *Aquisitor* para a responsabilidade Manutenção do Repositório de Recursos da PROPOST

A manutenção do *repositório de soluções* (Figura 38) é realizada mediante a pré-condição de que uma *solução de software* tenha sido definida. A partir daí é possível originar e manter um *repositório de soluções*, com base em técnicas para manutenção de dados, o que da margem à pós-condição de que o repositório esteja atualizado. Para a manutenção do *repositório de soluções* são usados grafos semânticos em RDF (LASSILA, SWICK, 1999).

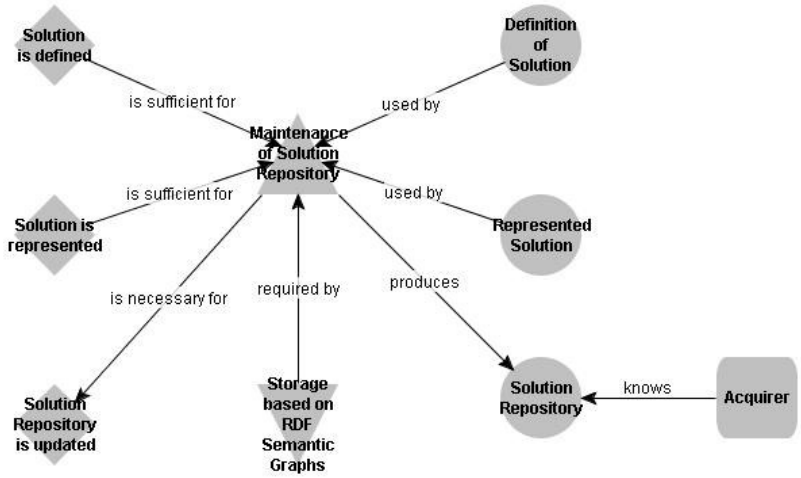


Figura 38 Modelo do Conhecimento e das Atividades do Agente *Aquisitor* para a responsabilidade Manutenção do Repositório de Soluções da PROPOST

A manutenção do *repositório de dados de uso* (Figura 39) é realizada mediante a pré-condição de que o usuário tenha saído do sistema, para originar e manter um *repositório de dados de uso*, com base em técnicas para *manutenção de dados de uso*, o que da margem à pós-condição de que o *repositório de dados de uso* esteja atualizado. Para a *manutenção de dados de uso*, são usados grafos semânticos em RDF (LASSILA, SWICK, 1999).

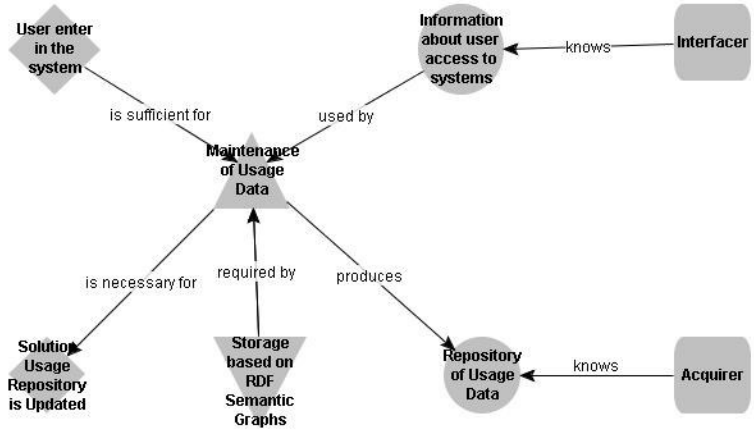


Figura 39 Modelo do Conhecimento e das Atividades do Agente *Aquisitor* para a responsabilidade Manutenção do *Repositório de dados de uso* da PROPOST

b) Modelagem do conhecimento e das atividades do agente Interfaceador

O agente *interfaceador* possui as seguintes responsabilidades: *análise de similaridade e recuperação; recomendação da solução; representação e indexação; monitoramento de uso; e representação da consulta.*

Para a análise de similaridade e recuperação (Figura 40), é suficiente que a consulta esteja representada, sendo tais necessárias para que *soluções de software* sejam recuperadas, tudo através de um algoritmo de recuperação de informações. Nesse processo, são usadas: a *representação da consulta* e o repositório de *soluções de software*, que é de conhecimento do agente *interfaceador*, sendo produzidas *soluções de software* recuperadas.

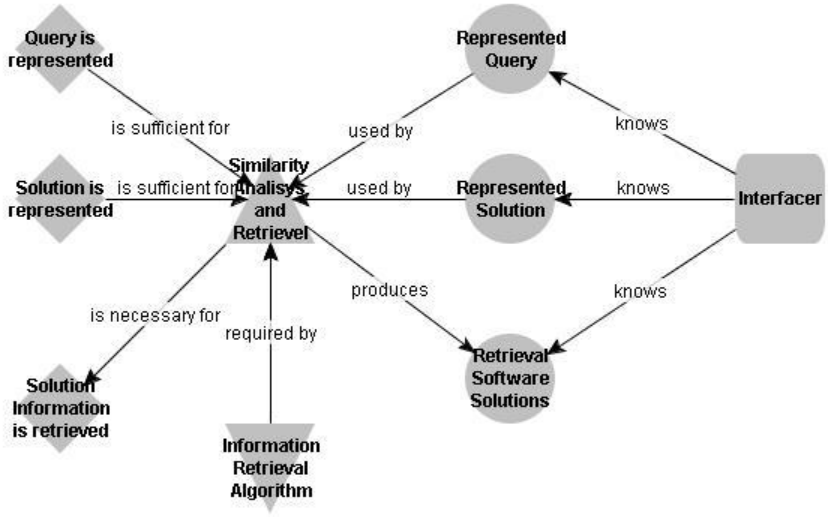


Figura 40 Modelo do Conhecimento e das Atividades do Agente *Interfaceador* para a responsabilidade Análise de Similaridade e Recuperação da PROPOST

Para a realização da *recomendação de soluções* (Figura 41) é suficiente que um *modelo de recomendação* esteja pronto, sendo esta responsabilidade necessária para que recomendações sejam entregues, isso feito através da apresentação de recomendações em uma listagem a ser exibida ao usuário na interface do sistema. Nesse processo, são usados: o *modelo de recomendação* e de *conhecimento do agente modelador*, para produzir recomendações de *soluções de software*, de conhecimento do *interfaceador*.

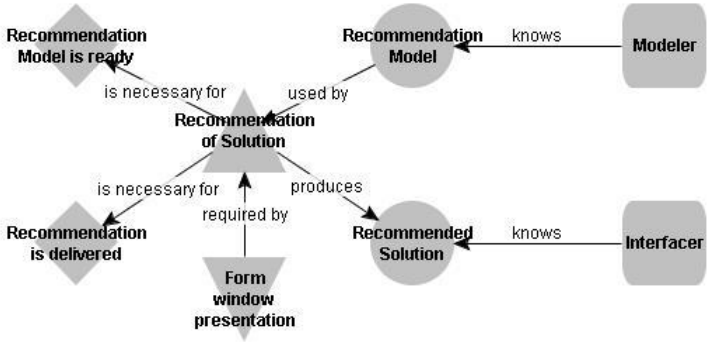


Figura 41 Modelo do Conhecimento e das Atividades do Agente *Interfaceador* para a responsabilidade Recomendação de Solução da PROPOST

Para a realização da *representação* e *indexação* (Figura 42) é suficiente que uma solução tenha sido definida. A partir daí o agente *Interfaceador* pode utilizar técnicas de *indexação e representação*, através da criação de um vetor de palavras-chave, e realizar *a representação e indexação* da *solução de software*. Nesse processo, é usada a *definição da solução de software*, que é de conhecimento do agente *Interfaceador*, para produzir representação da *solução e indexação* da referida solução.

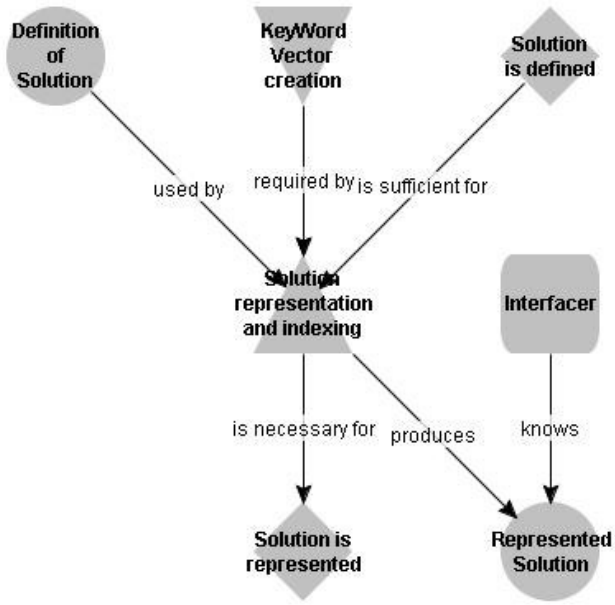


Figura 42 Modelo do Conhecimento e das Atividades do Agente *Interfaceador* para a responsabilidade Representação e Indexação da PROPOST

Para o *monitoramento de uso* (Figura 43), em relação às pré e pós-condições, respectivamente, é suficiente que um usuário entre no sistema, enquanto que a execução de tal responsabilidade, mediada pela captura implícita de usos de sistema, é necessária para que as informações sobre os usos capturados. Nesse processo, para cada de usuário monitorado, são usadas as respectivas informações de acesso, e são produzidas as informações de uso, sendo estes últimos conhecidos pelo agente *interfaceador*.

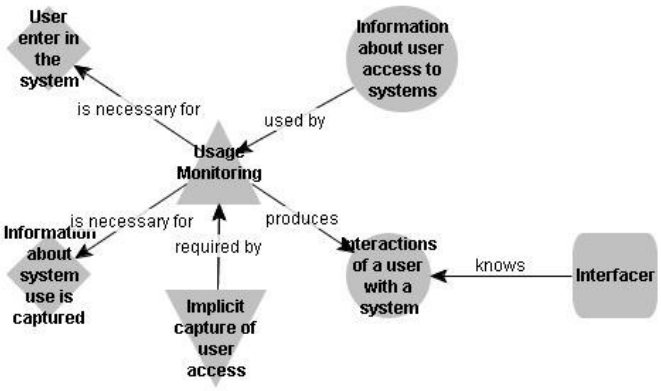


Figura 43 Modelo do Conhecimento e das Atividades do Agente *Interfaceador* para a responsabilidade Monitoramento de Uso da PROPOST

A *representação da consulta* (Figura 44) tem como pré-condição que uma consulta por *soluções de software* tenha sido especificada, e como pós-condição que a consulta esteja representada, sendo a destreza requerida para tanto a criação de vetores de palavras-chave. Na realização dessa responsabilidade ocorre o uso de uma necessidade de *solução de software* e a conseqüente produção da *representação da consulta*, que é de conhecimento do agente *interfaceador*.

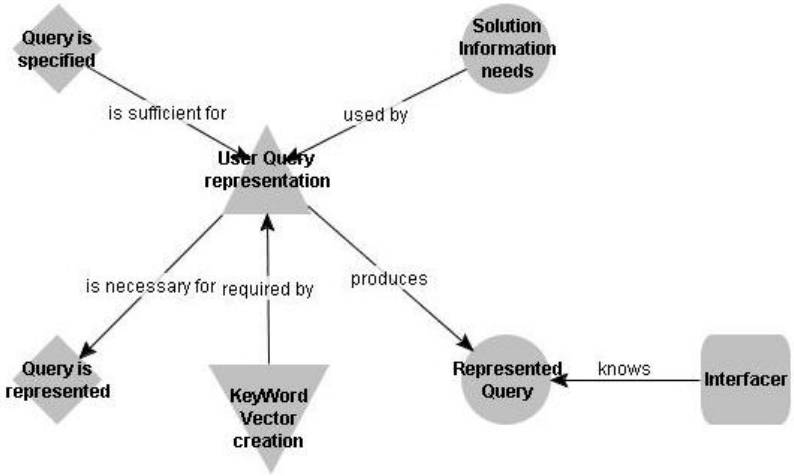


Figura 44 Modelo do Conhecimento e das Atividades do Agente *Interfaceador* para a responsabilidade *Representação da consulta* da PROPOST

c) Modelagem do conhecimento e das atividades do agente Minerador

O agente *minerador* possui as seguintes responsabilidades: *descoberta de padrões de uso; construção de modelos de recomendação e modelagem da área corrente.*

A *classificação da área corrente* (Figura 45) possibilita a identificação do *grupo de áreas com perfil similar* ao *perfil da área* corrente. Isso é possível a partir da comparação entre os *modelos das áreas* existentes e *modelo da área corrente*, o que é realizado através da utilização de técnicas relacionadas à classificação das áreas em grupos.

Para que essa responsabilidade ocorra torna-se é necessário que seja atendida a pré-condição de que o *modelo da área corrente* esteja disponível. Ao término da referida atividade ocorre a pós-condição em que *grupo de áreas com perfil similar* ao *perfil da área* corrente deverá estar disponível. A técnica utilizada nesta atividade consiste na aplicação do algoritmo de *K-Means* (SHAHABI, BANAEI-KASHANI, 2003).

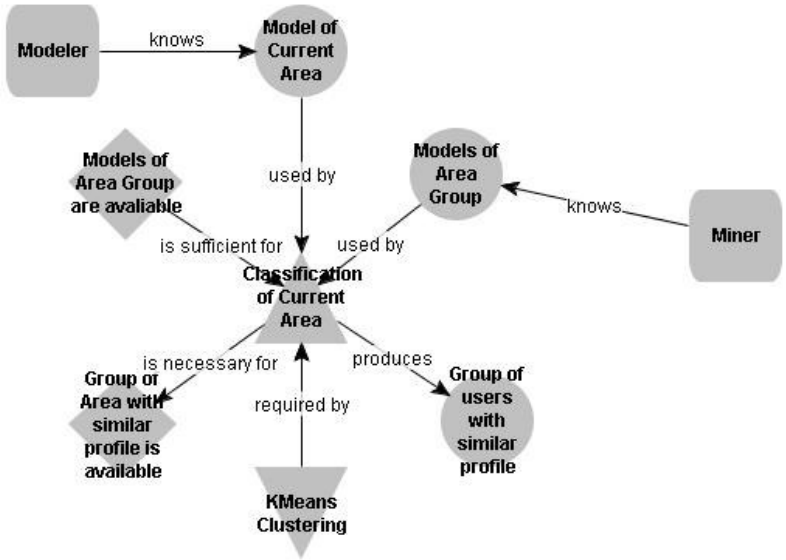


Figura 45 Modelo do Conhecimento e das Atividades do Agente *Minerador* para a responsabilidade *Classificação da área corrente* da PROPOST

A *descoberta de padrões de uso* (Figura 46), devido a sua complexidade, é dividida nas atividades de *extração de modelos de área* e de *aplicação de algoritmos de mineração*. A *extração de modelos de área* tem como pré-condições que um tempo de espera tenha transcorrido e que o *repositório de dados de uso* esteja atualizado e como pós-condição que *modelos de área* tenham sido extraídos, sendo que a destreza requerida para isso é o modelo de matrizes de características. Há, na execução da referida atividade, o uso do *repositório de dados de uso*, o qual deve ser de conhecimento do agente *aquisitor*, e a produção de *modelos de área*, os quais são conhecidos pelo agente *minerador*.

Para a aplicação de algoritmos de mineração, é suficiente que *modelos de área* tenham sido extraídos, sendo que tal é necessária para que *modelos de grupo de área* estejam disponíveis, isto através do algoritmo de agrupamento *KMeans*. Nesta atividade ocorre o uso de *modelos de área* e a produção de *modelos de grupo de área*, ambos de conhecimento do agente *minerador*.

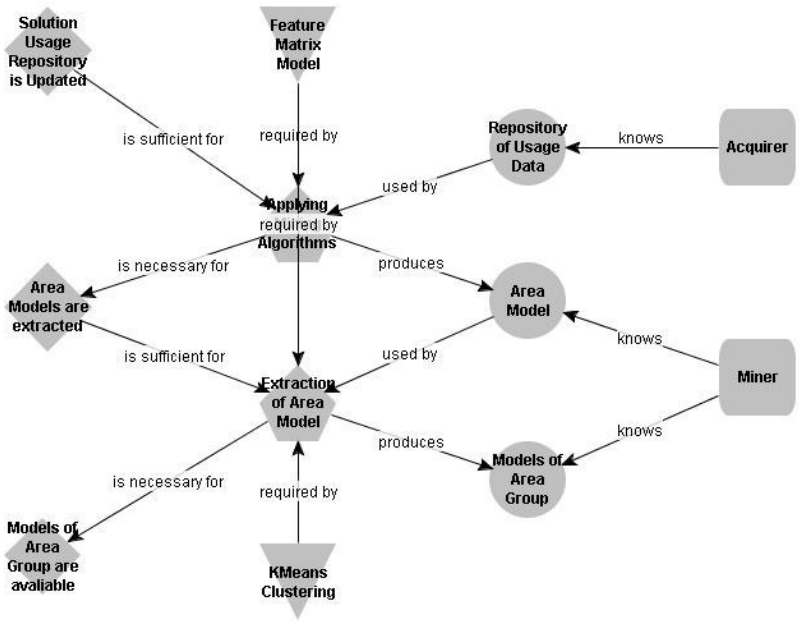


Figura 46 Modelo do Conhecimento e das Atividades do Agente *Minerador* para a responsabilidade Descoberta de Padrão de Uso da PROPOST

A *construção do modelo de recomendação* (Figura 47) tem como pré-condição que o grupo com *perfil similar* ao da *área corrente* esteja disponível, e como pós-condição que o *modelo de recomendação* esteja pronto, sendo a destreza requerida para tanto a abordagem da filtragem colaborativa. Há, na execução dessa responsabilidade, o uso do *grupo similar ao perfil da área* corrente, de conhecimento do agente *minerador*, e a produção do *modelo de recomendação*, conhecido pelo *modelador*.

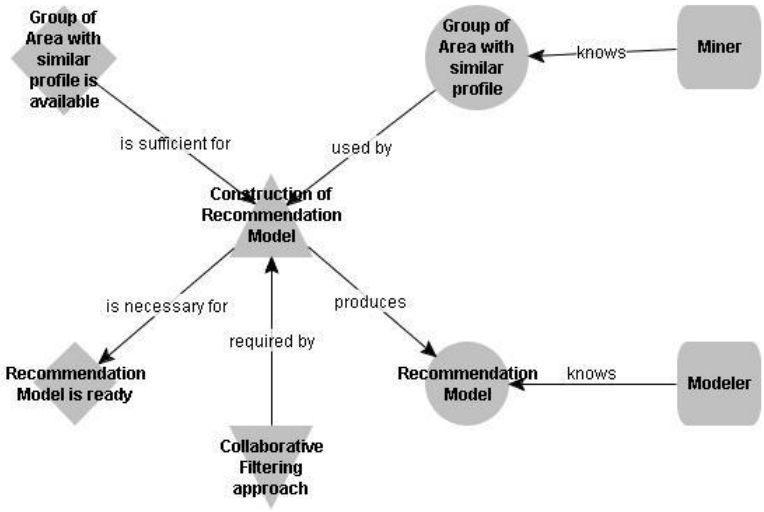


Figura 47 Modelo do Conhecimento e das Atividades do Agente *Modelador* para a responsabilidade *Construção do modelo de recomendação* da PROPOST

Para a *modelagem da área corrente* (Figura 48) é suficiente que informações sobre usos de sistemas pela área tenham sido capturadas, sendo que ela é necessária para que o *modelo da área corrente* esteja disponível, isso através do modelo de matrizes de características. Nesse processo são usadas informações sobre uso de sistemas da *área corrente*, que são de conhecimento do agente *interfaceador*, e é produzido o *modelo da área corrente*, conhecido pelo *modelador*.

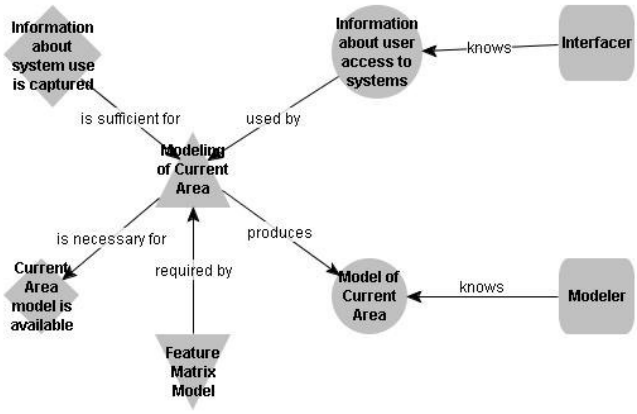


Figura 48 Modelo do Conhecimento e das Atividades do Agente *Modelador* para a responsabilidade *Modelagem da área corrente* da PROPOST

d) Modelagem do conhecimento e das atividades do agente Analisador

O agente *analisador* possui as seguintes responsabilidades: *elaboração de estimativas de projeto; elaboração de priorização de projeto e elaboração de análise de uso*.

A elaboração das *estimativas dos projetos* (Figura 49) utiliza as técnicas de pontos de função para estimar o custo e o prazo dos projetos. A pré-condição para esta responsabilidade é que a *solução de software* relativa ao projeto tenha sido definida, e a pós-condição é a realização da estimativa.

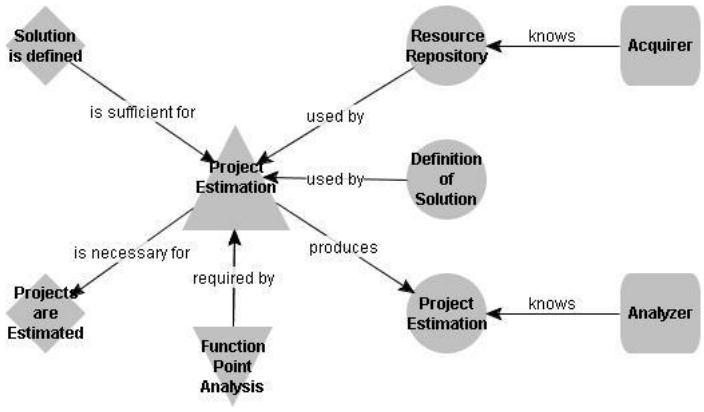


Figura 49 Modelo do Conhecimento e das Atividades do Agente *Analisador* para a responsabilidade Elaboração de Estimativas de Projeto da PROPOST

A *elaboração da priorização dos projetos* usa as estimativas dos projetos (Figura 50), através das quais produz a *priorização dos projetos* a serem realizados. Para isso, são necessárias técnicas para a *priorização de projetos*, nas quais utiliza-se um modelo de priorização baseado em avaliações. A pré-condição para que esta responsabilidade ocorra é que a *estimativa dos projetos* tenha sido realizada, e a pós-condição é a realização da referida priorização.

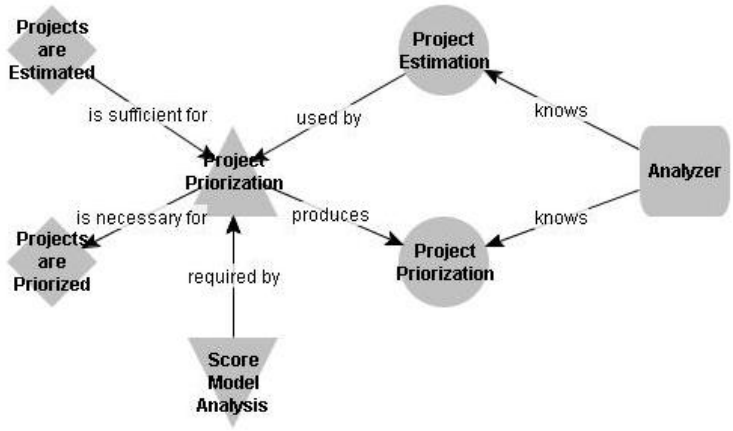


Figura 50 Modelo do Conhecimento e das Atividades do Agente *Analisador* para a responsabilidade Elaboração de Priorização de Projeto da PROPOST

A responsabilidade de *elaboração da análise de uso* (Figura 51) das soluções dos sistemas de informação usa a informação de acesso aos sistemas de informação pelo usuário contidas no *repositório de uso de dados*, através da qual produz a *análise de uso dos sistemas de informação*.

Para a realização desta responsabilidade são necessárias técnicas para a elaboração de *análises de uso dos sistemas de informação* pelos usuários, relativas à comparação dos indicadores atuais de uso com indicadores previamente estabelecidos. A pré-condição para isto é que o a informação do uso esteja disponível no *repositório de uso de dados*, e a pós-condição é a referida análise seja realizada.

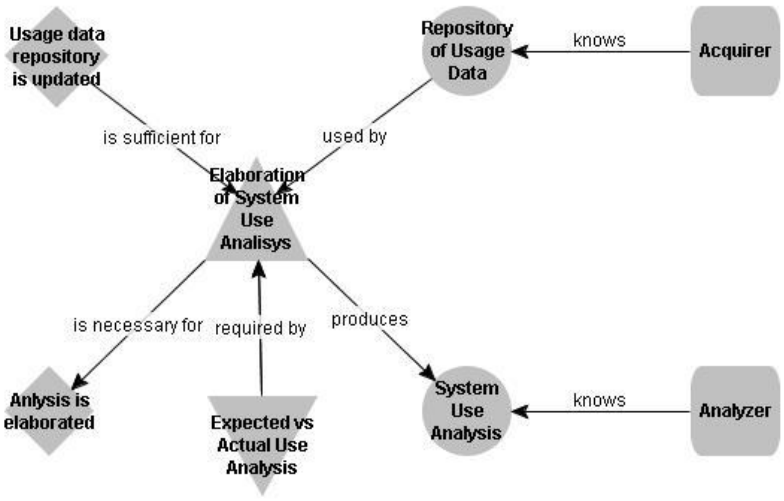


Figura 51 Modelo do Conhecimento e das Atividades do Agente *Analisador* para a responsabilidade Elaboração de Análise de Uso da PROPOST

4.4.6 Modelagem dos Estados dos Agentes

a) Modelagem dos estados do agente Aquisitor

A Figura 52 exibe o *Modelo dos Estados do Agente Aquisitor* da PROPOST, que representa os estados pelos quais o agente passa durante seu funcionamento, bem como as transições que leva de um para outro.

Desse modo, o agente *aquisitor* transita de seu estado inicial quando o *sistema é inicializado* e fica no estado *aguardando informações sobre acesso aos sistemas, priorização de projetos ou definição de solução.*

Quando a *informação sobre uso de sistema* é capturada, ele vai para o estado *atualizando uso de dados no repositório*, após o qual ele retorna para o estado anterior.

Quando a informação sobre *priorização de projetos* é processada, ele vai para o estado *atualizando repositório de recursos*, após o qual ele retorna para o estado anterior; e quando a informação sobre *definição de solução* é processada, ele vai para o estado *atualizando repositório de soluções* após o qual ele retorna para o estado anterior, e deste ou de qualquer outro estado ele pode transitar para o final quando a aplicação for finalizada.

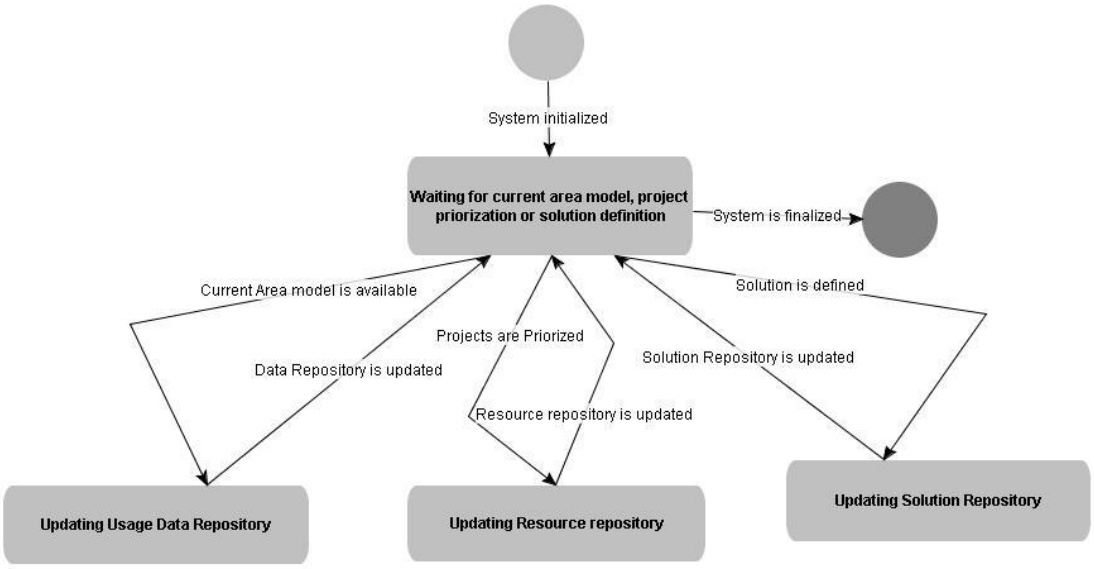


Figura 52 Modelo dos Estados do Agente *Aquisitor* da PROPOST

b) Modelagem dos estados do agente Analisador

A Figura 53 exibe o *Modelo dos Estados do Agente Analisador da* PROPOST, que representa os estados pelos quais o agente passa durante seu funcionamento, bem como as transições que leva de um para outro.

Ao iniciar a aplicação, o agente *analisador* permanece no estado *aguardando informações de definições de soluções e recursos, ou do repositório de dados de projetos, ou sobre uso de sistemas*.

Quando as informações de *definições de soluções e recursos* são processadas, o agente passa ao estado *elaborando estimativas de projetos*. Após elaborar essas estimativas, ele passa ao estado *elaborando priorização de projetos*, após o qual ele retorna ao seu estado inicial.

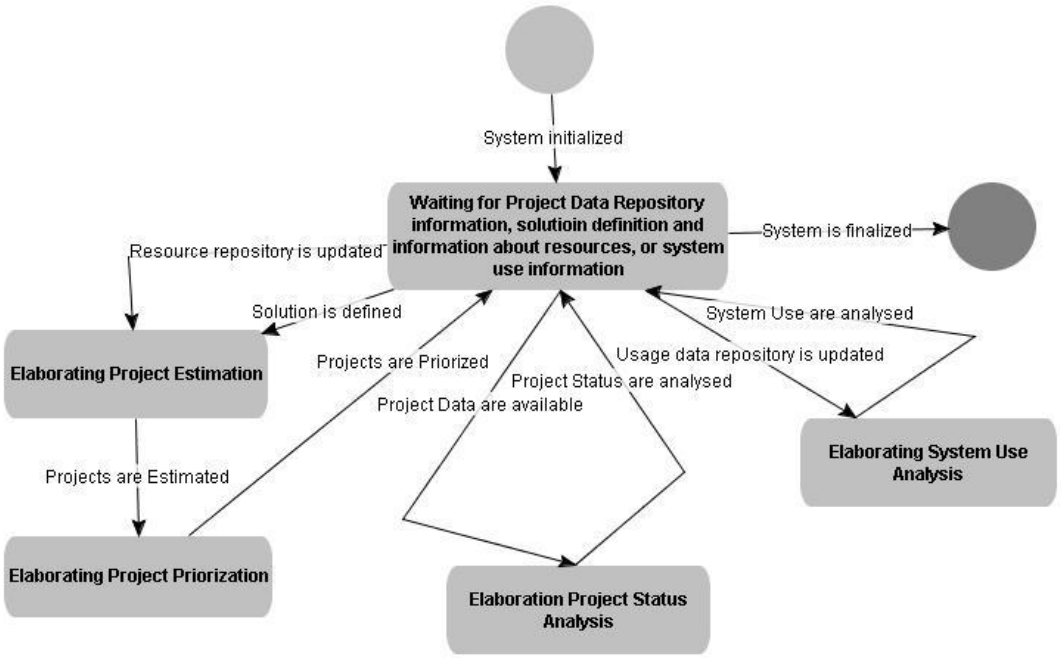


Figura 53 Modelo dos Estados do Agente *Analisador* da PROPOST

Quando as informações sobre o *repositório de dados de projetos* são processadas, o agente passa ao estado *elaborando análise de status de projetos* e, após realizar essa análise ele retorna ao seu estado inicial.

Quando as informações sobre *uso dos sistemas* são processadas, o agente passa ao estado *elaborando análise de uso de sistemas* e, após realizar essa análise, ele retorna ao seu estado inicial, sendo que deste ou de qualquer dos outros estados ele pode transitar para o final quando a aplicação for finalizada.

c) Modelagem dos estados do agente Interfaceador

A Figura 54 exibe o *Modelo dos Estados do Agente Interfaceador* da PROPOST, que representa os estados pelos quais o agente passa durante seu funcionamento, bem como as transições que leva de um para outro.

Dessa forma, uma vez que um usuário entre no sistema, o agente *interfaceador* inicia sua operação, permanecendo no estado *monitorando o comportamento de consumo do usuário* ou aguardando uma consulta por *soluções de software*, ou ainda aguardando a definição de uma nova *solução de software*.

Se uma consulta por *soluções de software* é especificada, então o referido agente assume o estado *representando uma consulta por soluções de software*, no qual ele verifica os dados fornecidos pelo usuário e cria um vetor de palavras-chave a partir da consulta.

Quando a consulta está representada, então o agente *analisador* transita para o estado *comparando* e *recuperando soluções de software*, em que é realizada a comparação entre a consulta e as *soluções de software* contidas no repositório, havendo a recuperação de algumas. No momento em que as *soluções de software* são recuperadas, o agente retorna ao seu estado de monitoração e espera.

Por outro lado, se uma *solução de software* é definida, o agente *interfaceador* assume o estado *representando e* indexando uma solução e, ao final desta, retorna para o estado anterior.

Já quando um *modelo de recomendação* está pronto, então o agente vai para o estado *processando recomendações de soluções de software*, o que é realizado a partir do modelo, fazendo com que ele retorne para o seu estado de monitoração e espera.

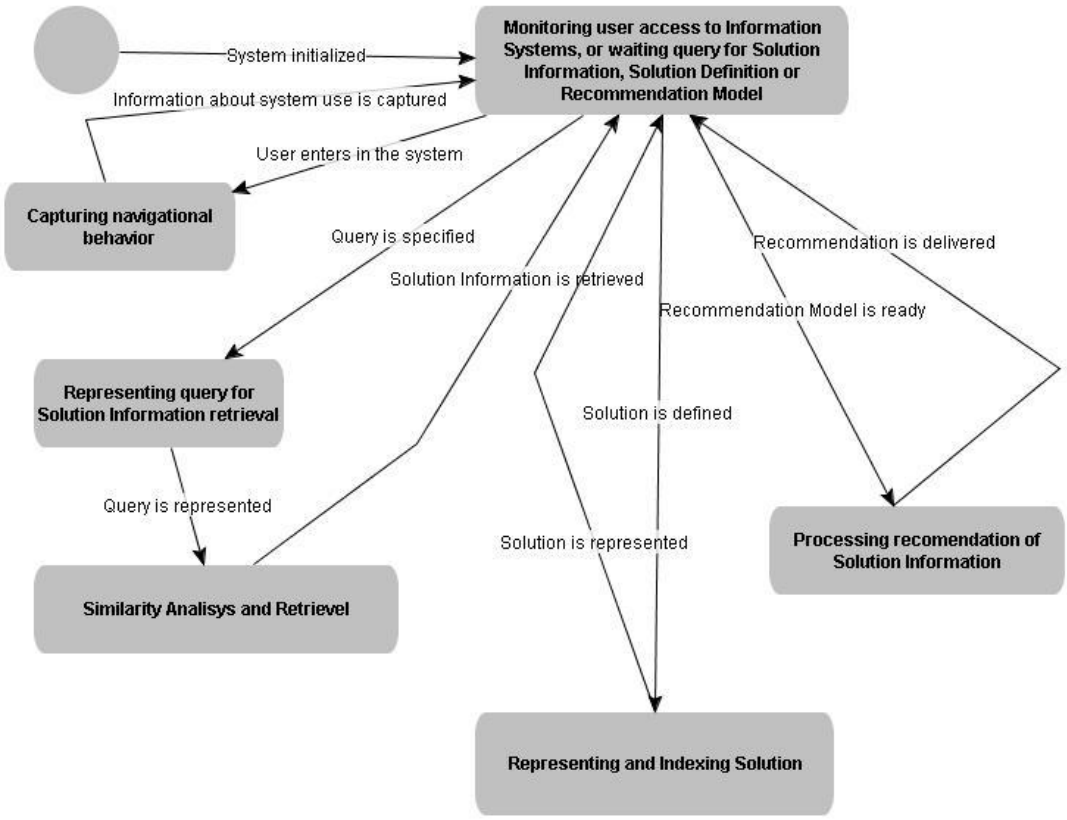


Figura 54 Modelo dos Estados do Agente *Interfaceador* da PROPOST

d) Modelagem dos estados do agente Minerador

A Figura 55 exibe o *Modelo dos Estados do Agente Minerador da* PROPOST, que representa os estados pelos quais o agente passa durante seu funcionamento, bem como as transições que leva de um para outro.

Dessa maneira, quando o sistema é inicializado, o agente *minerador* assume o estado *aguardando a atualização do repositório de dados de uso* ou *modelo da área corrente*. Quando os dados do repositório são atualizados, ele assume o estado *extraindo modelos de área do repositório de dados de uso*, no qual o agente emprega as características de consumo armazenadas para a representação de *modelos de área*.

Assim que *modelos de área* são extraídos, ele entra no estado *criando modelos de grupo de áreas* através de algoritmos de mineração, sendo que é esse agrupamento que possibilita a *classificação da área*, permitindo a recomendação de *soluções de software* por meio de filtragem colaborativa.

Quando os modelos de *grupo de áreas* estão disponíveis, o agente passa para o estado *aguardando o modelo de área corrente* vindo do agente *modelador*. Uma vez chegado tal modelo, o agente entra no estado *classificando a área corrente*, em que a área é classificada em um *grupo de áreas com perfil similar* ao seu, através de algoritmos de agrupamento dinâmico.

Feita a classificação, um *grupo com perfil similar ao da área corrente* fica disponível e, então, o agente retorna para o estado *aguardando o modelo da área corrente* vindo do agente *modelador*, sendo que deste ou de qualquer dos outros estados ele pode transitar para o final quando a aplicação for finalizada.

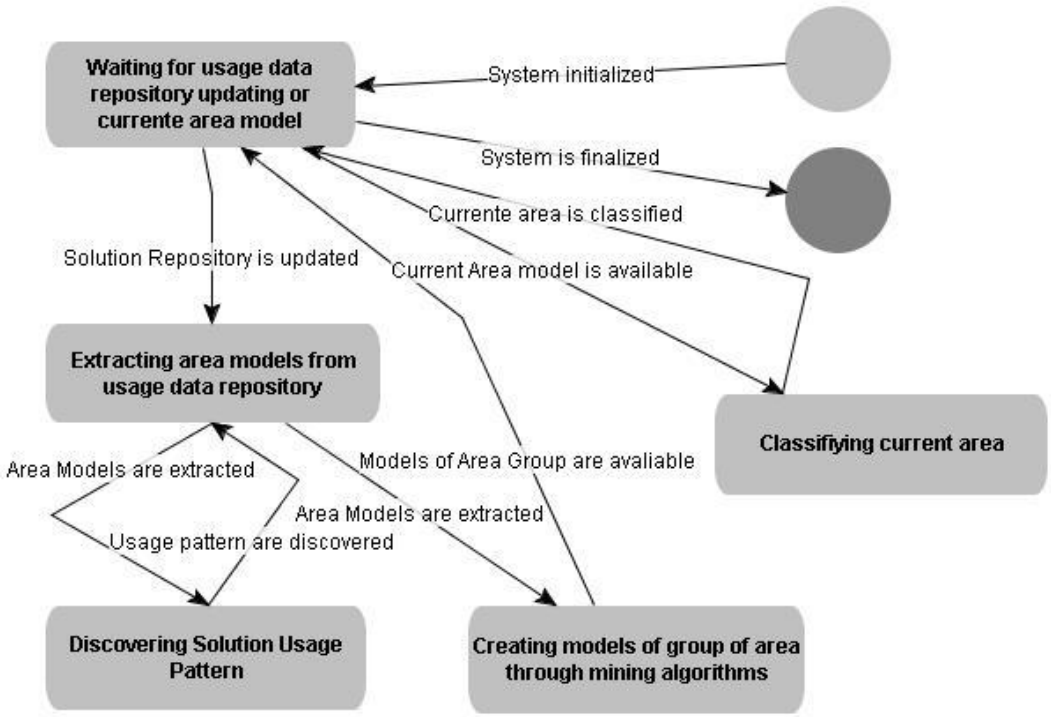


Figura 55 Modelo dos Estados do Agente *Minerador* da PROPOST

e) Modelagem dos estados do agente Modelador

A Figura 56 exibe o *Modelo dos Estados do Agente Modelador da* PROPOST, que representa os estados pelos quais o agente passa durante seu funcionamento, bem como as transições que levam de um para outro.

O agente *modelador* inicia sua execução no estado aguardando informações sobre usos dos sistemas pelos usuários ou por disponibilidade do *grupo de áreas similares*. Quando informações sobre usos dos sistemas são capturadas, o estado dele passa a ser modelando a *área corrente*, em que ele utiliza tais informações para construir um novo modelo ou atualizar os modelos já existentes e, posteriormente, retorna ao seu estado inicial.

Quando o *modelo da área corrente* está disponível, o agente passa ao estado *aguardando a classificação da área corrente* feita pelo agente *minerador*. Uma vez que o grupo com perfil similar ao da *área corrente* está disponível, o agente muda para o estado *construindo modelo de recomendação*, no qual ele identifica as *soluções de software* escolhidas por *áreas com perfis similares* ao da *área corrente*.

Logo que o *modelo de recomendação* está disponível, o agente volta para o estado *aguardando por informações sobre informações sobre acessos de usuários* vindas do agente *interfaceador*, e deste ou qualquer outro estado ele pode passar transitar ao estado final quando a aplicação finalizar.

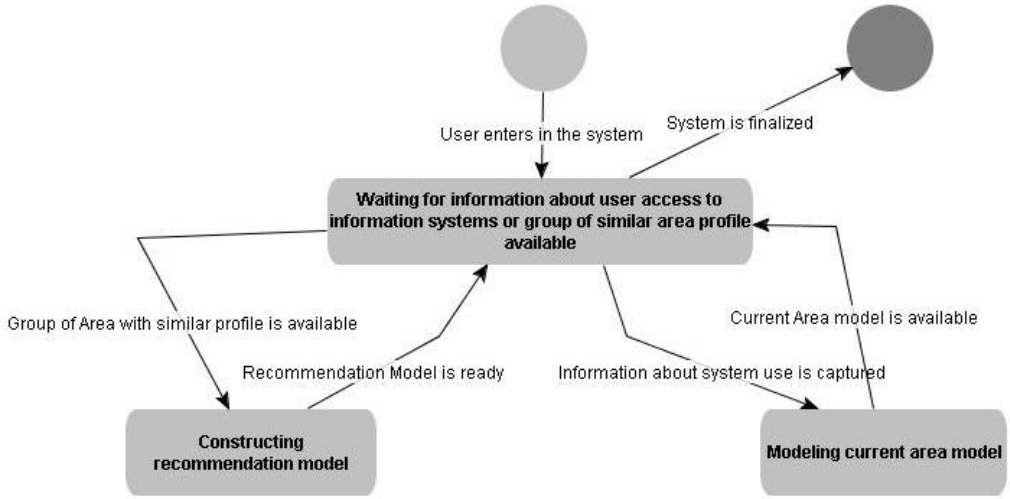


Figura 56 Modelo dos Estados do Agente *Modelador* da PROPOST

4.5 Implementação da Ferramenta PROPOST

Na fase de Implementação da Aplicação, conforme definido na metodologia MAAEM, o principal produto resultante é o *Modelo de Implementação da Aplicação*, o qual é resultante das seguintes atividades:

- Modelagem de Agentes e Comportamento
- Modelagem de Atos de Comunicação entre Agentes

4.5.1 Modelagem de Agentes e Comportamento

A Figura 57 mostra o *Modelo de Agentes e Comportamentos* da PROPOST, o qual relaciona instâncias das classes *agentes* e *comportamento* da plataforma JADE - *Java Agent DEvelopment framework* (Bellifemine, 2003), a tecnologia de implementação adotada, identificadas a partir dos *agentes* e *responsabilidades* projeto, respectivamente.

Assim, foram instanciados os agentes i*nterfaceador*, *modelador, aquisitor*, *minerador* e *analisador*. E, de acordo com as responsabilidades por eles assumidas, procedeu-se a instanciação dos comportamentos de cada um, os quais apresentam diversos tipos, conforme detalhado a seguir.

O agente *interfaceador* desempenha os seguintes comportamentos: *monitorar uso dos sistemas*, que é *cíclico*, pois permanece em constante estado de captura das informações sobre as utilizações de sistemas; *representar consultas*, que é que é atômico, porque é executado uma única vez quando o usuário expressa sob a forma de uma consulta a sua necessidade pontual sobre informações de *soluções de software*; *representação e indexação de soluções de software*, que é cíclico, pois permanece em constante estado de captura de definições de *soluções de software*, a serem representadas e indexadas; *comparação e recuperação*, que é seqüencial, pois ocorre em seqüência a uma *representação de consulta*; e *recomendação de solução*, que é simples, uma vez que é disparado quando há uma requisição de recomendação pelo usuário.

O agente *modelador* fica encarregado dos seguintes comportamentos: *modelar área corrente*, que é *simples*, já que é executado cada vez que sejam repassadas informações sobre escolhas; e *construir modelo de recomendação*, que também é *simples*, uma vez que entra em execução sempre que um grupo esteja disponível.

O agente *aquisitor* cuida dos seguintes comportamentos: *manter dados de uso*, que é simples, porque é executado uma única vez, quando um usuário acessa um sistema, para que sejam armazenados no repositório os dados referentes ao uso; *manter dados de solução de software*, que é simples, porque é executado uma única vez quando uma determinada *solução de software* é definida, para que sejam armazenados no repositório os dados referentes à solução; *manter dados de recursos*, que é simples, porque é executado uma única vez quando um determinado recurso é alocado em um projeto, para que sejam armazenados no repositório os dados referentes à alocação do recurso.

O agente *minerador* executa os seguintes comportamentos: *descobrir padrões de consumo*, que é *seqüencial*, tendo como sub-comportamentos *extrair modelos de área*, que é *cíclico*, pois é executado dentro de intervalos de tempos pré-definidos, e *aplicar algoritmos de mineração*, que é simples, já que é executado sempre que *modelos de área* são extraídos; e *classificar a área corrente*, que é *simples*, pois é executado cada vez que um modelo de *área corrente* esteja disponível.

O agente *analisador* executa os seguintes comportamentos: *elaboração de estimativas de projeto*, que é atômico, pois é executado uma única vez para cada projeto a ser estimado; *elaborar priorização de projetos*, que é atômico, pois é executado uma única vez para cada projeto a ser priorizado; *elaboração de análise de status*, que é atômico, pois é executado uma única vez para cada projeto a ser analisado; e *elaboração de análise de sistemas*, que é atômico, pois é executado uma única vez para cada sistema a ser analisado.

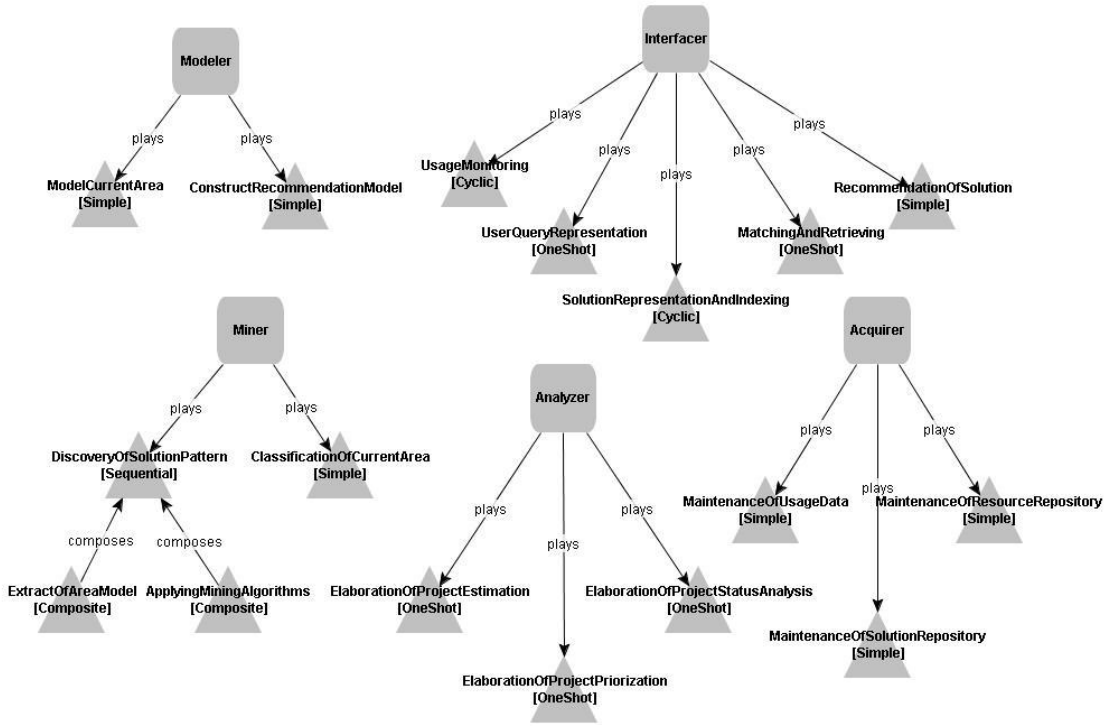


Figura 57 Modelo de Agentes e Comportamento da PROPOST

4.5.2 Modelagem de Atos de Comunicação entre Agentes

A Figura 58 mostra o *Modelo de Atos de Comunicação entre Agentes* da PROPOST, que exibe a forma como o conhecimento da sociedade multiagente é realizado entre os agentes, por meio das mensagens que uns enviam para outros.

Após capturar a informação de uso do sistema por um usuário o agente *interfaceador* envia uma mensagem "1-*INFORM_REF (informações sobre os usos de sistemas)*" para o *Modelador*. Tendo em vista que cada usuário está lotado em apenas uma área, é possível construir o *modelo da área corrente*, o qual será usado para classificação desta área. Posteriormente, caso haja outras áreas modeladas, a aplicação já poderá classificar a área, o que faz após o agente *minerador* receber uma mensagem "2-*INFORM-REF (modelo da área corrente)*" do *modelador*.

Em seguida, o agente *minerador* requer ao *aquisitor*, através de uma mensagem "3-*QUERY_REF (repositório de dados de uso)*", os dados de uso a partir dos quais são extraídos os *modelos da área* para formação de grupos em que se classifica a *área corrente*, por meio da *mineração de uso*. E após isso, o o agente *minerador* remete ao *modelador* uma mensagem "4-*INFORM_REF (grupo de áreas com perfil similar 'a área corrente)*" contendo o grupo em que se enquadra a *área corrente*, na qual é baseado o processo de recomendação de *soluções de software*.

O próximo passo consiste no *modelador* repassar ao *interfaceador* a mensagem "5-*INFORM_REF (modelo de recomendação)*" para que ele possa providenciar, com base em tal modelo, a efetiva oferta à área de *soluções de software* recomendadas, recebendo em seguida uma confirmação de que as "*recomendações de solução de software foram entregues*", indicando que a operação foi bem sucedida.

Além dos eventos acima descritos, pode também estar ocorrendo em paralelo, a notificação de que um "*usuário saiu do sistema*", repassada ao agente *interfaceador*, originando outra mensagem "6-*INFORM_REF (informações sobre usos de sistemas)*" dele para o *modelador* e, por fim, uma "7-*INFORM-REF (modelo da área corrente)*" ao *aquisitor*, o que possibilita o armazenamento dos dados da presente *sessão de uso*, os quais, juntamente com os outras sessões passadas e futuras, constituem outra parte do subsídio demandado pela classificação.

Pode ainda estar ocorrendo uma interação entre o usuário e o sistema, relativa à busca por *soluções de software*, a partir de palavras-chaves contidas na descrição das mesmas. Neste caso, ocorre uma necessidade de informação pontual, cuja especificação é realizada pelo usuário via interface e, reconhecida pelo agente *interfaceador*. Este agente se encarrega de enviar ao *aquisitor*, através de uma mensagem "8-*QUERY_REF (representação da consulta)*", na qual ele recupera as *soluções de software* relevantes à necessidade de informação e as entrega ao usuário.

Outras interações que podem ainda ocorrer são referentes às informações sobre *análise de status de projetos* e *análise de uso de sistemas de informação*, as quais são carregadas automaticamente pelo sistema sempre que um usuário acessa o mesmo. Essas análises são realizadas a partir das requisições “9-QUERY_REF (busca no *repositório de dados de uso*)” e “10-QUERY_REF (busca no repositório de dados de projetos)”, ambas enviadas do agente *analisador* para o agente a*quisitor*.

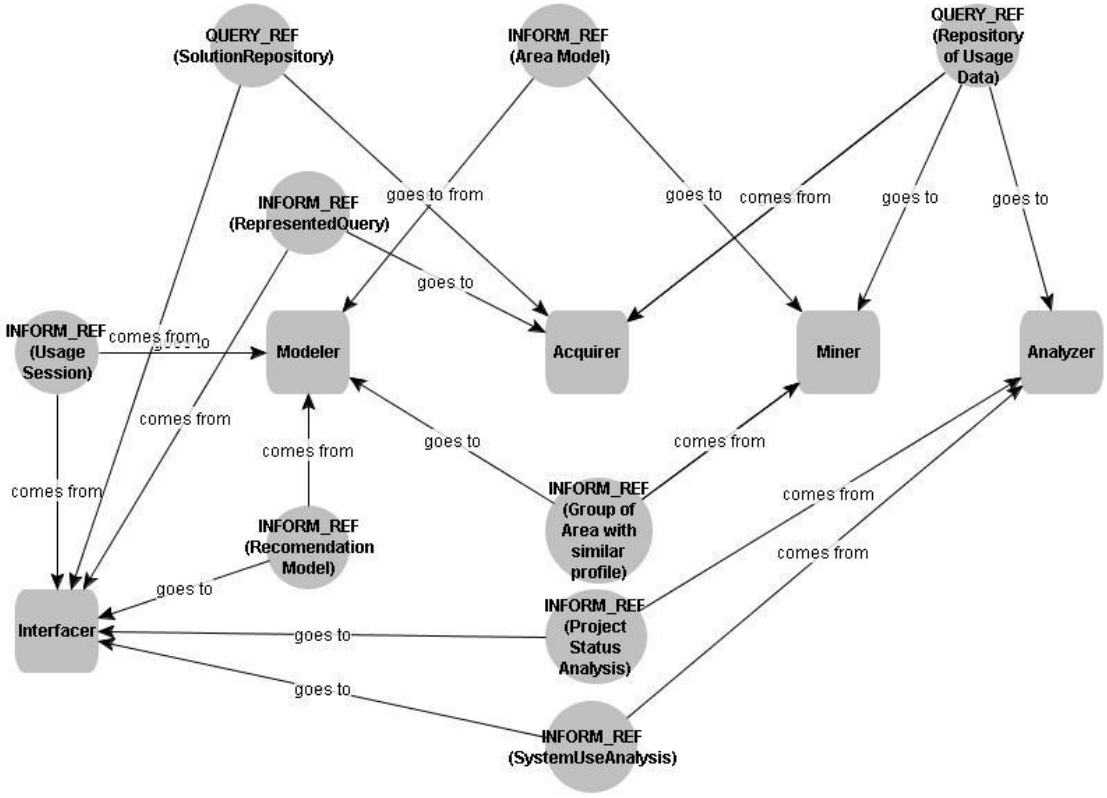


Figura 58 Modelo de Atos de Comunicação entre Agentes da PROPOST

4.6 Benefícios da Modelagem Semântica realizada na PROPOST

A modelagem semântica é um recurso importante na captação de detalhes do mundo real com um nível mais rico em detalhes. Devido a isso, a sua utilização como recurso complementar no desenvolvimento de software é de grande valor, podendo-se observar demandas da utilização de seus conceitos em outras várias áreas, tais como Inteligência Artificial, Web Semântica, Modelagem de Processos de Negócio, Modelagem do Conhecimento, etc.

No desenvolvimento de software em geral (seja ele orientado a objetos, baseado em agentes, em eventos, etc) a captura do conhecimento adquirido proporciona a redução no tempo e custo do desenvolvimento, bem como o aumento na sua qualidade. E nesse particular, as ontologias são uma forma emergente de captura e representação do conhecimento em um determinado domínio. As ontologias são particularmente úteis para representar abstrações de software de alto nível, como modelos de domínio e arquiteturas reutilizáveis, pois fornecem uma terminologia não ambígua que pode ser compartilhada por todos os atores envolvidos em um processo de desenvolvimento (GIRARDI, 2004). Nesse contexto, entendemos abstrações de software reutilizáveis como especificações em alto nível, sucintas, naturais e úteis que correspondem a uma ou mais realizações em um nível mais detalhado de representação.

Conforme já mencionado neste trabalho, a ferramenta PROPOST foi modelada utilizando a ONTORMAS, uma ferramenta dirigida por ontologias, a qual representa o conhecimento descrito nas metodologias MADEM e MAAEM. A ONTORMAS contempla as fases e atividades necessárias para a análise, o projeto e a implementação de aplicações multiagente através da reutilização de artefatos de software produzidos na Engenharia de Domínio Multiagente. Por outro lado, as ontologias ONTOINFO e ONTOWUM descrevem famílias de produtos de software para o desenvolvimento de aplicações nas áreas de Recuperação e Filtragem de Informação, respectivamente. A tarefa de modelagem foi realizada no ambiente Protégé (http://protege.stanford.edu).

Os benefícios obtidos em utilizar a ONTORMAS foi devido a essa ontologia constituir um guia para o processo de modelagem de aplicações multiagentes, proporcionando também a reutilização de modelos de domínio e frameworks multiagentes previamente existentes no seu repositório. Isso facilitou a construção dos modelos a serem gerados, uma vez que os passos a serem seguidos e os produtos a serem gerados estavam formalmente contidos nessa ontologia. A representação dos artefatos de software resultantes foi realizada através da instanciação das correspondentes subclasses de tarefas, produtos, conceitos e relacionamentos de modelagem, de acordo com a semântica representada na ontologia.

Os benefícios obtidos com o reuso das ontologias ONTOWUM e ONTOINFO foi devido às mesmas constituírem artefatos de software relacionados a modelos de domínio e *frameworks* multiagentes reutilizáveis. Dessa forma, seguindo os passos descritos na ONTORMAS, pudemos proceder à reutilização dos elementos contidos nos modelos de domínio durante as fases de modelagem da Análise, bem como reutilizar os elementos contidos nos *frameworks multiagentes* durante as fases de modelagem do Projeto e Implementação, previstas na MAAEM. A Figura 59 ilustra graficamente um exemplo de reutilização ocorrida na PROPOST em relação aos objetivos contidos na ONTOWUM, os quais foram especializados para o domínio da gestão de portifólio. Maiores detalhes sobre o reuso da ONTOWUM e ONTOINFO ocorrido na PROPOST são descritos ao longo do capítulo 4, o qual trata da modelagem da ferramenta.

Os benefícios da modelagem no ambiente *Protégé*, por sua vez, foram decorrentes do mesmo promover relacionamentos semânticos entre os elementos presentes nas ontologias (conceitos, relações, instâncias, etc), o qual proporcionou facilidades para a confecção dos modelos, tais como a busca e recuperação destes elementos, bem como para a incorporação destes aos modelos produzidos. Dessa forma, mudanças em determinados elementos em um modelo, puderam se refletir nos outros modelos onde estes elementos estavam presentes, evitando o retrabalho de mudanças manuais em cada um desses modelos.

As ontologias também foram úteis na implementação da aplicação, tendo em vista que elas proporcionaram uma estrutura de terminologia comum para comunicação e troca de mensagens entre os agentes.

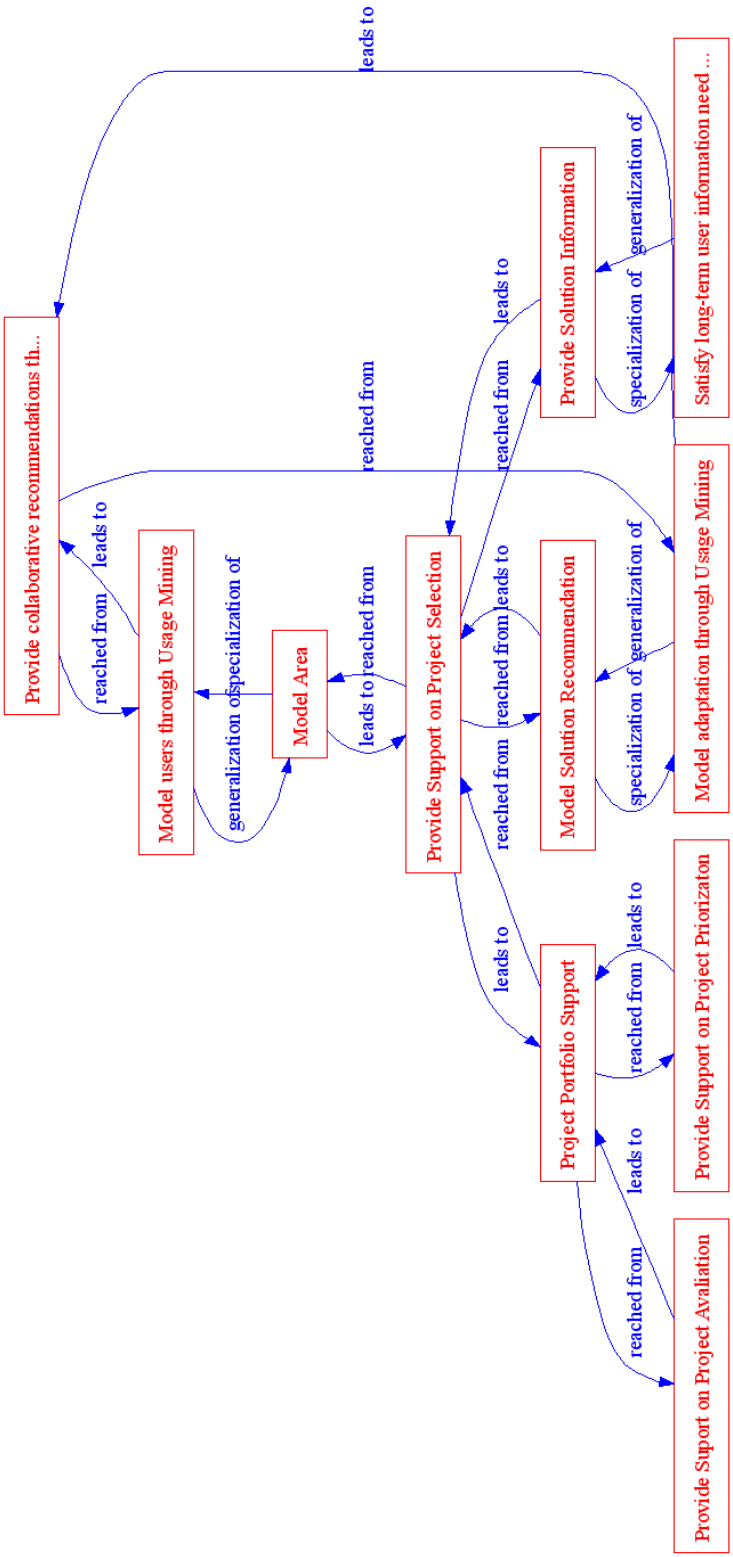


Figura 59 Reuso de objetivos na PROPOST

4.7 Considerações Finais do Capítulo

Este capítulo mostrou todo o processo de modelagem da ferramenta PROPOST, no qual foram utilizadas: a ferramenta dirigida por ontologias ONTORMAS, bem como reutilizados os artefatos descritos nas ontologias ONTOINFO e ONTOWUM. Conforme mostrado anteriormente, a ONTORMAS contempla as fases e atividades necessárias para a análise, o projeto e a implementação de aplicações multiagente através da reutilização de artefatos de software produzidos na Engenharia de Domínio Multiagente. Também foi mencionado que as ontologias ONTOINFO e ONWOWUM descrevem famílias de produtos de software para o desenvolvimento de aplicações nas áreas de Recuperação e Filtragem de Informação, respectivamente.

As ontologias são particularmente úteis para representar abstrações de software de alto nível, como modelos de domínio e arquiteturas reutilizáveis, fornecendo uma terminologia a ser compartilhada pelos atores envolvidos em um processo de desenvolvimento. Tais características puderam ser constatadas ao longo deste capítulo, no demonstramos como a reutilização destes artefatos foi realizada ao longo do desenvolvimento da ferramenta.

Os maiores benefícios obtidos na modelagem da PROPOST ocorreram principalmente devido a facilidades relacionadas à reutilização das abstrações de software representados nas ontologias ONTOINFO e ONTOWUM, bem como pelo uso da ferramenta dirigida por ontologias ONTORMAS como guia para este processo. Esses benefícios tiveram influência direta na redução do tempo de desenvolvimento da ferramenta e, conseqüentemente no trabalho realizado, reduzindo seu custo e proporcionando qualidade, tendo em vista que as abstrações reutilizadas presumidamente constituem componentes previamente testados no âmbito do grupo GESEC.

5 PROTOTIPAÇÃO DA INTERFACE COM O USUÁRIO

5.1 Considerações Iniciais

Neste capítulo apresentamos a prototipação das interfaces da aplicação PROPOST com o usuário. A referida atividade teve como objetivo verificar na prática os benefícios que a prototipação da interface usuário poderia proporcionar à concepção da referida ferramenta e, conseqüentemente, analisar a viabilidade de adotar esta fase como complementar a ser inserida na metodologia MAAEM, utilizada na modelagem do processo de desenvolvimento.

Devido à prototipação proporcionar uma visualização mais clara dos objetivos da aplicação, a mesma possibilita um melhor entendimento dos objetivos relacionados no *modelo de objetivos da aplicação*, podendo inclusive ajudar a identificar eventuais objetivos adicionais que, a princípio, não haviam sido identificados nos requisitos da aplicação.

5.2 Interfaces e objetivos da aplicação

5.2.1 Tela de Login

A Figura 60 mostra a tela de *login* da ferramenta, na qual o usuário pode ser identificado. Tendo em vista que cada usuário está alocado somente a uma área gerencial, essa identificação poderá ser utilizada posteriormente para reconhecer a área na qual o usuário pertence, e gerar as recomendações de soluções de software para a respectiva área, quando houver solicitação desta funcionalidade.

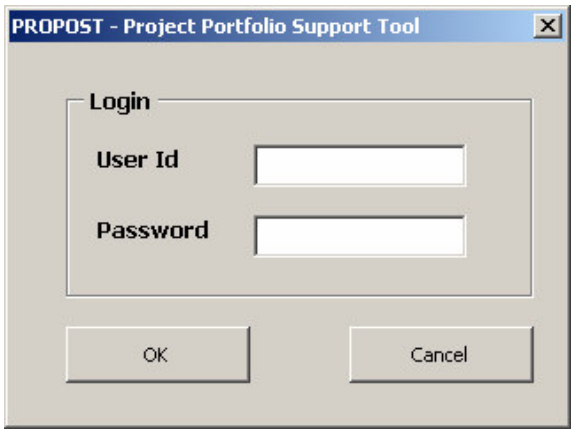


Figura 60 Tela de Login

5.2.2 **Seleção de Projetos**

A Figura 61 mostra a interface referente à funcionalidade de *Seleção de Projetos*. Essa funcionalidade corresponde ao objetivo específico *Prover Suporte à Seleção de Projetos*. Nesta interface pode-se observar as seguintes funcionalidades:

Solution Detail: nessa funcionalidade é possível verificar os detalhes de uma determinada *solução de software*, bem como criar, atualizar ou deletar a alocação de uma *solução de software* para uma determinada *área gerencial*. Essa funcionalidade está relacionada com a responsabilidade *Manutenção de Dados de Uso* no modelo de objetivos.

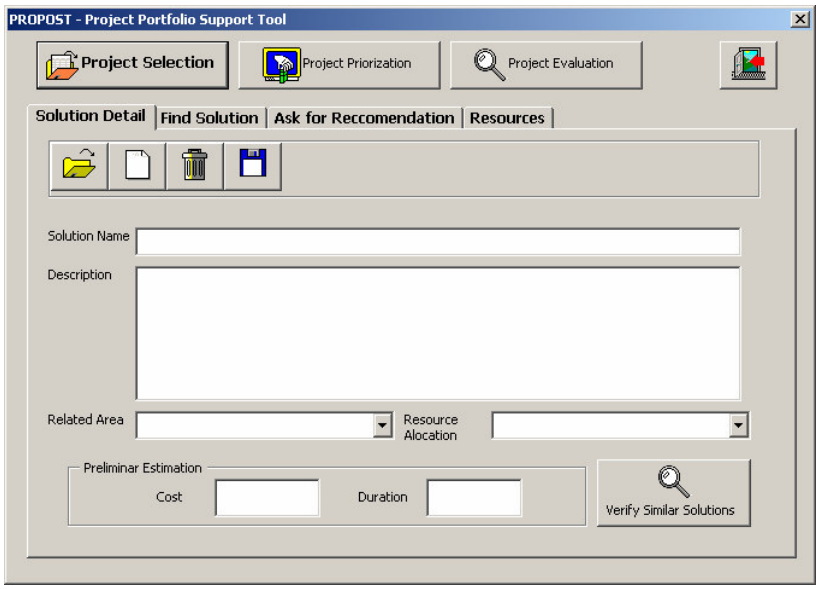


Figura 61 Seleção de Projetos: interface principal

Ainda em relação a essa interface, pode ser usada a funcionalidade *Verify Similar Solutions*, que mostra soluções de software semelhantes à solução corrente (Figura 62), e está relacionada ao objetivo *Prover Informação de Solução* no modelo de objetivos.

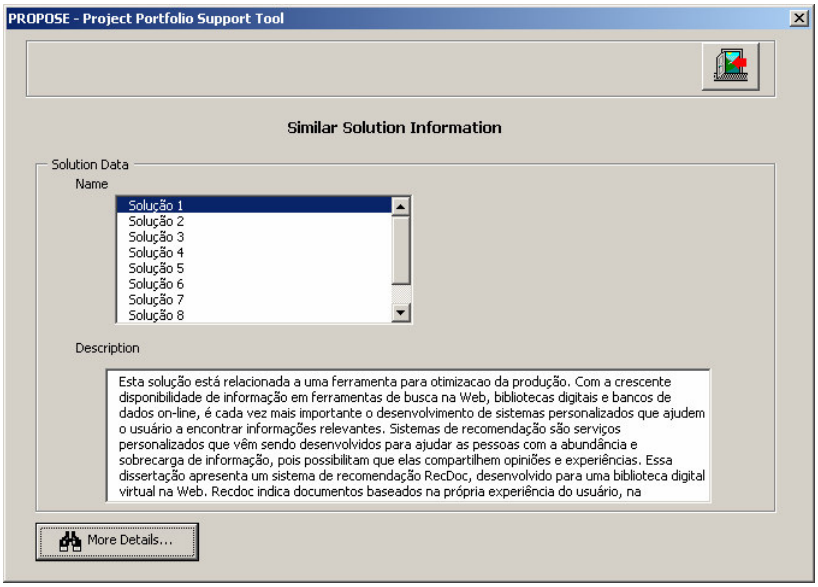


Figura 62 Informações sobre *Soluções de software*

Find Solution: nessa funcionalidade (Figura 63) é possível realizar uma consulta nas *soluções de software* existentes, a partir de critérios (palavras-chave) definidos pelo usuário. A mesma está relacionada com o objetivo de *Prover Informações de Soluções* no modelo de objetivos.

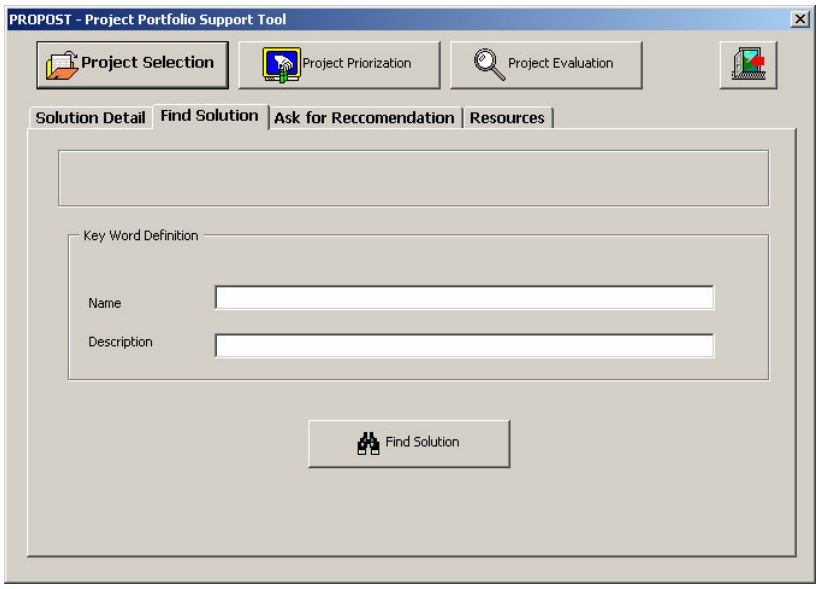


Figura 63 Critérios de Pesquisa de Soluções

Ask for Recomendation: nessa funcionalidade (Figura 64), o usuário pode solicitar uma recomendação de *solução de software* para uma determinada área do seu interesse.

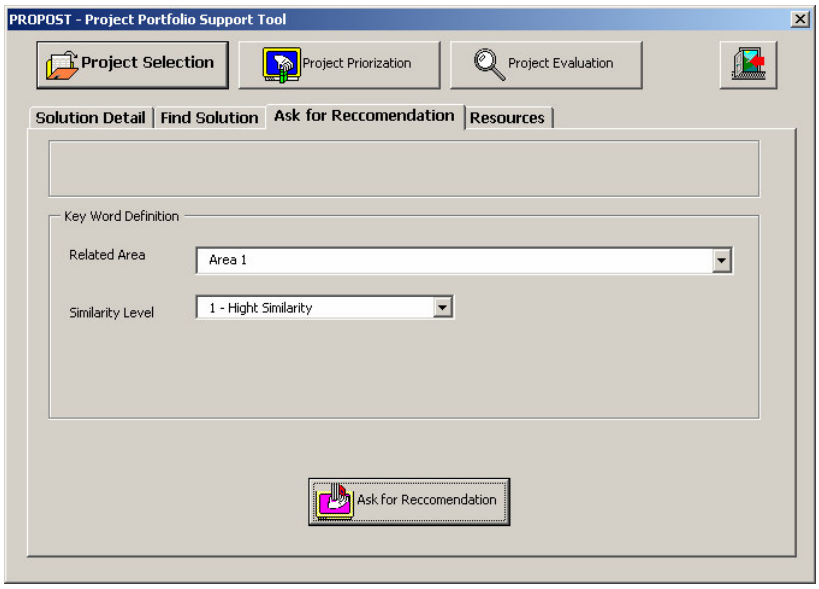


Figura 64 Solicitação de Recomendação

O resultado dessa recomendação é mostrado na Figura 65, na qual pode-se observar uma lista de *soluções de software* exibidas com seus respectivos detalhes. Ao pressionar o botão *More Details*, o usuário pode ainda obter informações mais detalhadas sobre as soluções apresentadas.

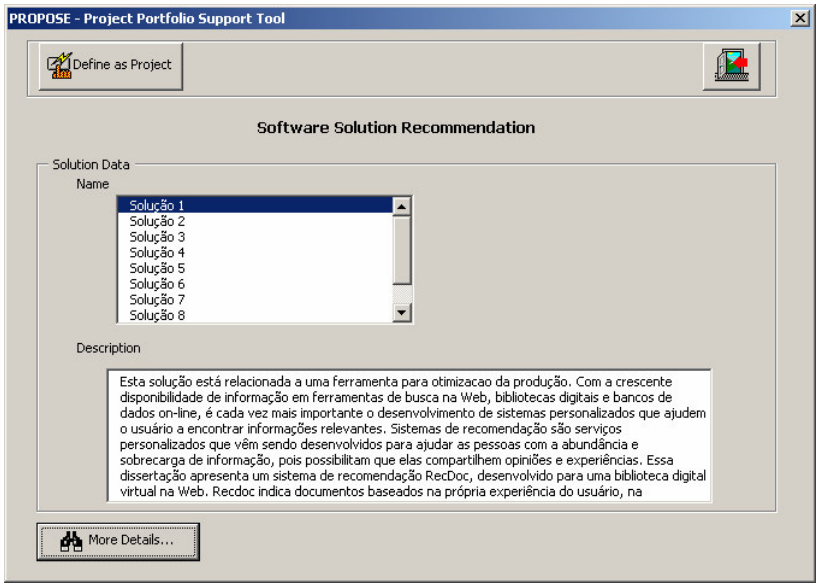


Figura 65 Resultado da Recomendação

Resources: essa é uma funcionalidade (Figura 66) para pesquisa sobre a alocação de recursos nos projetos existentes, sendo útil na realização de estimativas de projetos, e está relacionada com a responsabilidade *Elaboração de Estimativas de Projetos* no modelo de objetivos..

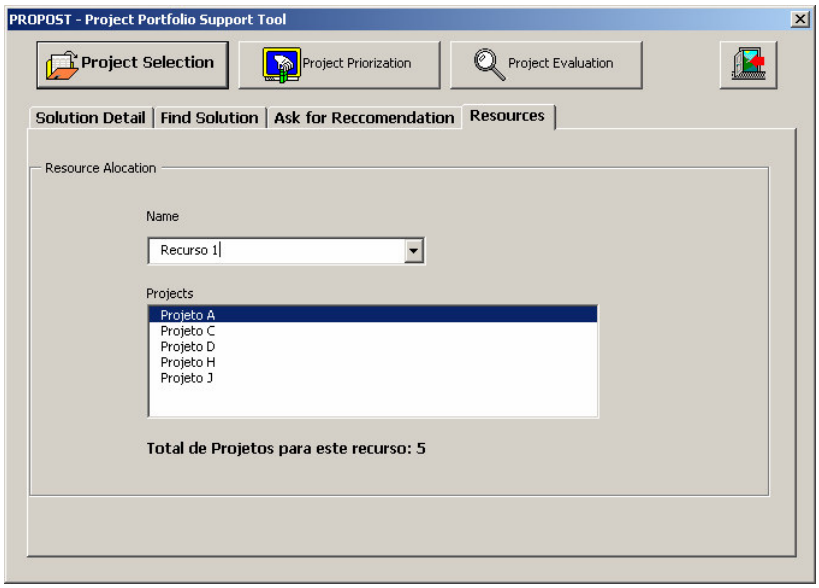


Figura 66 Alocação de Recursos

5.2.3 Priorização de Projetos

A Figura 67 mostra a interface referente à funcionalidade de *Priorização de Projetos*. Essa funcionalidade corresponde ao objetivo específico *Prover Suporte à Priorização de Projetos* no modelo de objetivos..

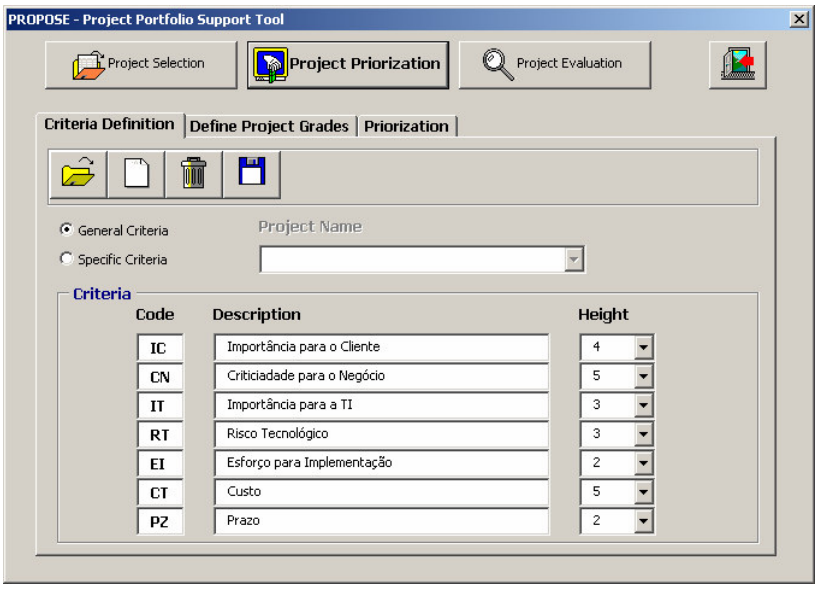


Figura 67 Priorização de Projetos: interface principal

Nesta interface pode-se observar as seguintes funcionalidades da aplicação:

Criteria Definition: nessa funcionalidade (Figura 68) é possível consultar, criar, atualizar e atualizar os critérios que serão usados para o cálculo da *priorização dos projetos*, bem como o peso dos mesmos. A mesma está relacionada com a responsabilidade *Elaboração de Priorização de Projetos* no modelo de objetivos..

Convém observar que podem ser definidos tanto *critérios genéricos*, quanto *critérios específicos* para um determinado projeto. Acreditamos que esse modelo venha a suprir uma limitação comumente observada nos modelos similares ao apresentado os quais, por serem essencialmente genéricos, possuem pouca flexibilidade para a definição de determinadas prioridades, que deveriam ser tratadas de maneira específica.

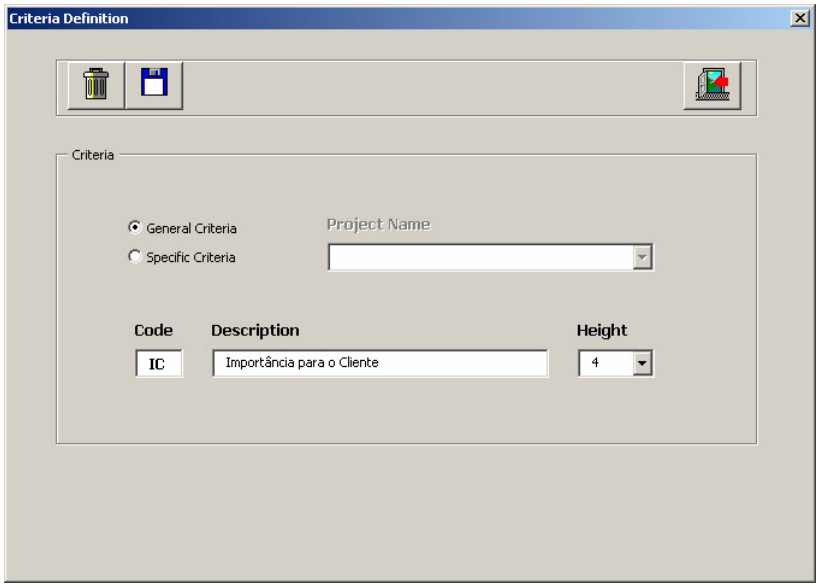


Figura 68 Definição de Critérios de Priorização

Define Project Grades: nessa funcionalidade (Figura 69), o usuário pode definir as pontuações resultantes para cada projeto em cada um dos critérios definidos anteriormente. A mesma está relacionada com a responsabilidade *Elaboração de Priorização de Projetos* no modelo de objetivos..

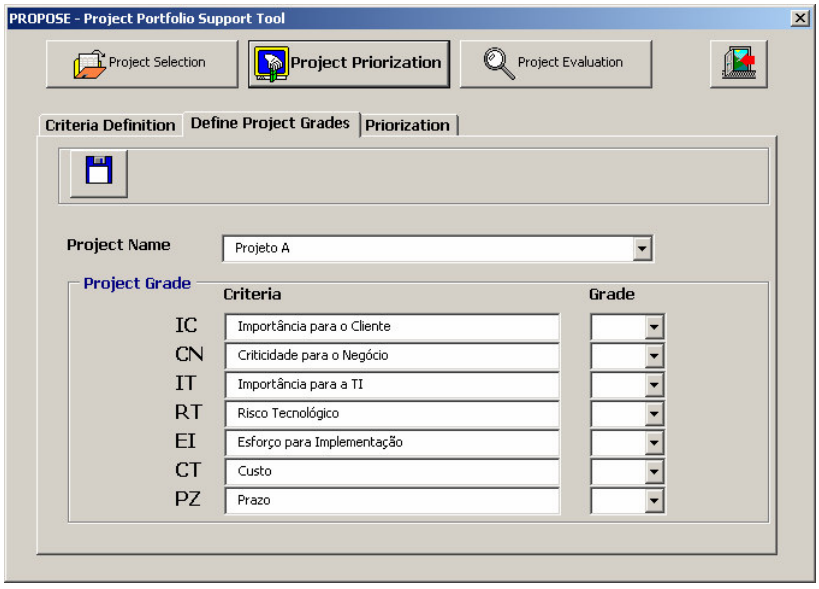


Figura 69 Pontuação dos Projetos

Prioritization: essa funcionalidade mostra o cálculo resultante da *Priorização* propriamente dita (Figura 70), o qual consiste na média ponderada entre as notas dos projetos e os pesos dos critérios definidos. Nessa interface o usuário pode ainda mudar explicitamente uma determinada prioridade, e definir a prioridade final dos projetos. A mesma está relacionada com a responsabilidade *Elaboração de Priorização de Projetos* no modelo de objetivos.

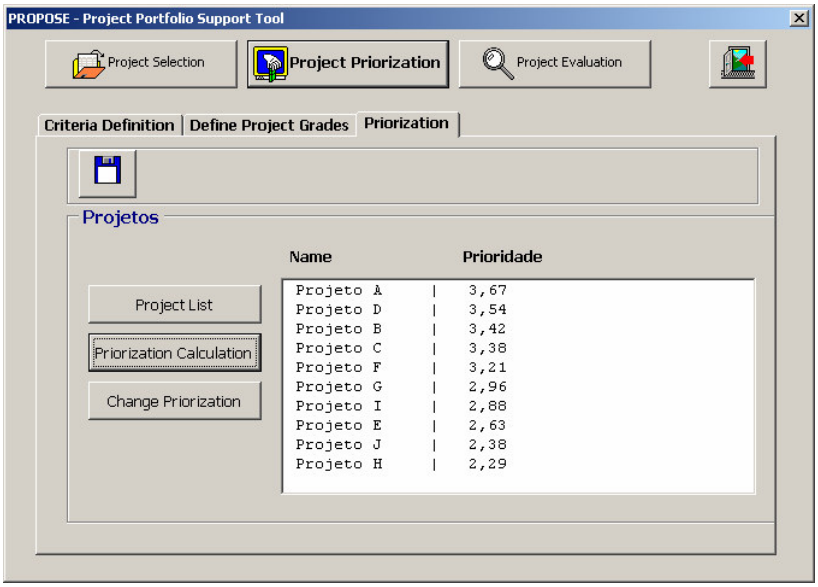


Figura 70 Resultado da Priorização

5.2.4 Avaliação de Projetos

A Figura 71 mostra a interface referente à funcionalidade de *Avaliação de Projetos*. Essa funcionalidade corresponde ao objetivo específico *Prover Suporte à Avaliação de Projetos* no modelo de objetivos. Nesta interface pode-se observar as seguintes funcionalidades:

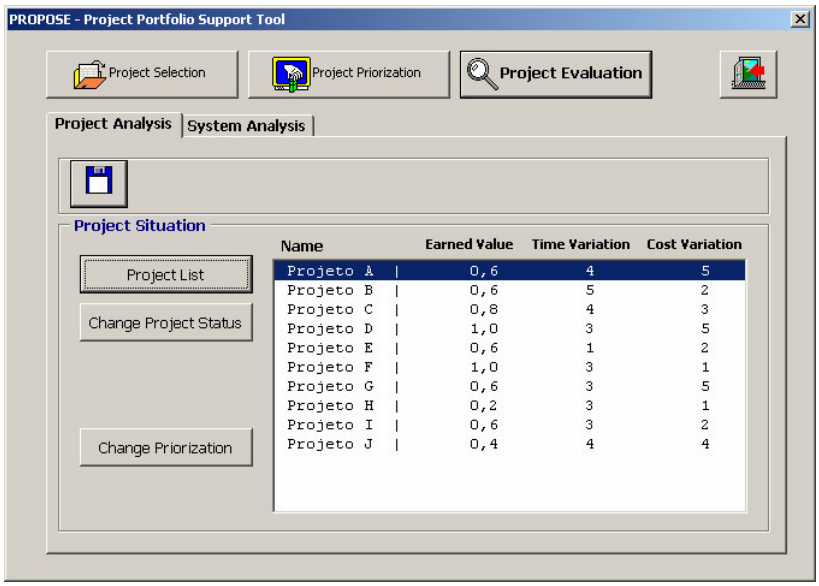


Figura 71 Avaliação de Projetos: interface principal

Project Analysis: nessa funcionalidade (Figura 72), é exibida a lista dos projetos existentes e o resultado de uma análise dos mesmos baseado no método *Earned Value* (PMBOK, 2004). Pode-se ainda mudar o *status* de um projeto, para o qual existem as opções de cancelar, paralisar ou reativar, bem como mudar a prioridade do mesmo. Essa funcionalidade está relacionada com a responsabilidade *Elaboração da Análise da Situação dos Projetos* no modelo de objetivos..

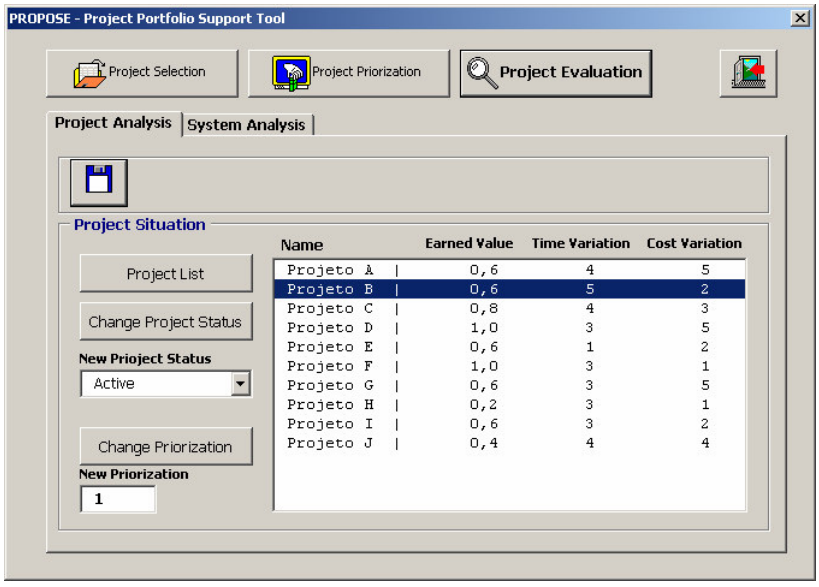


Figura 72 Resultado da Análise da Situação dos Projetos

System Analysis: nessa funcionalidade (Figura 73), é exibida uma *análise da situação atual* do uso dos sistemas existentes, para a qual são considerados fatores tais como: *percentual de uso atual, uso previsto, disponibilidade atual, disponibilidade prevista*, etc. Essa funcionalidade está relacionada com a responsabilidade *Elaboração da análise de uso* dos Sistemas no modelo de objetivos.

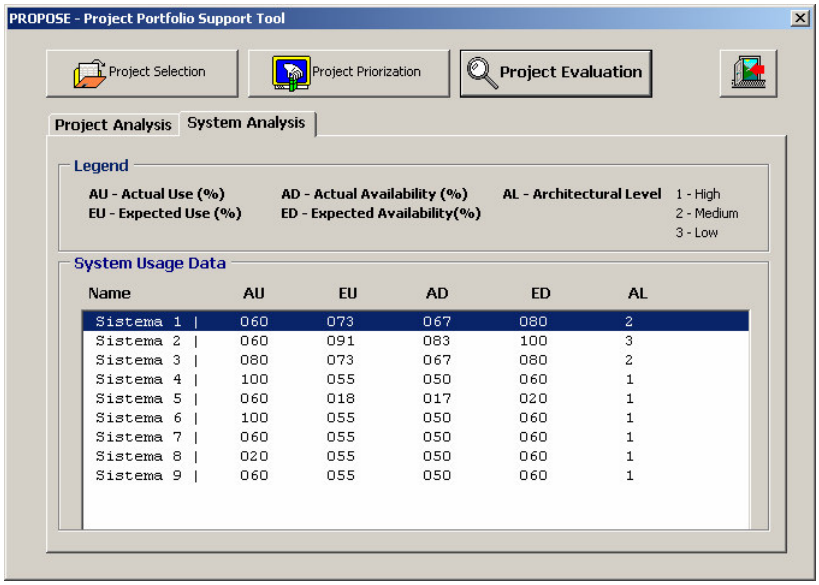


Figura 73 Resultado da Análise de Uso dos Sistemas

5.3 Considerações Finais do Capítulo

Conforme era nossa expectativa, a prototipação das interfaces da aplicação proporcionou uma visualização mais detalhada das funcionalidades a serem contempladas pela referida ferramenta, bem como um maior entendimento do seu funcionamento, tendo em vista que a mesma permitiu uma apresentação da aplicação mais próxima ao seu estágio final.

Tal visualização proporcionou também refinamentos complementares nos modelos desenvolvidos, tais como: correções e ajustes, introdução de novos conceitos e relacionamentos, melhor visibilidade das responsabilidades desempenhadas pelos agentes, etc.

Conseqüentemente, o fato da prototipação ser realizada em conjunto com as demais fases da modelagem, poderá agregar mais valor ao referido processo, e assim tornando os modelos mais ricos em termos das informações apresentadas, bem como mais completos em relação aos objetivos a serem contemplados pela ferramenta.

Ao longo da referida prototipação, observamos que os maiores benefícios foram relacionados principalmente aos modelos da fase de Análise, porém também houve benefícios tanto na fase de Projeto como na própria fase de Implementação da aplicação.

Dentre os benefícios observados com a prototipação, pode-se destacar, por exemplo, que a própria navegação das interfaces do protótipo proporcionou uma maior visibilidade em relação a determinadas trocas de mensagens e interações entre agentes, facilitando o seu entendimento e conseqüente implementação.

Devido ao acima exposto, sugerimos como contribuição de melhoria que a prototipação constitua uma fase a ser incorporada na metodologia MAAEM, a qual deve iniciar após a de Modelagem de Objetivos, devendo ser refinada durante toda a fase de Análise da Aplicação, à proporção que mais funcionalidades forem identificadas para a aplicação em questão e conforme haja necessidade.

A Tabela 4 apresenta a seguir a composição correspondente as fases da MAAEM, incluindo a prototipação da interface usuário, conforme sugerido neste capítulo.

<table>
<tr><th colspan="2">Fases</th><th>Passos</th><th colspan="2">Produtos</th></tr>
<tr><td colspan="2" rowspan="5">Análise
da
Aplicação</td><td>Modelagem de Conceitos</td><td>Modelo de Conceitos</td><td rowspan="5">Especificação
da
Aplicação</td></tr>
<tr><td>Modelagem de Objetivos</td><td>Modelo de Objetivos</td></tr>
<tr><td>Prototipação da Interface Usuário</td><td>Protótipo da Interface Usuário</td></tr>
<tr><td>Modelagem de Papéis</td><td>Modelo de Papéis</td></tr>
<tr><td>Modelagem de Interações entre Papéis</td><td>Modelo de Interações entre Papéis</td></tr>
<tr><td rowspan="6">Projeto
da
Aplicação</td><td colspan="2">Modelagem do Conhecimento da Sociedade Multiagente</td><td colspan="2">Modelo do Conhecimento da Sociedade Multiagente</td><td rowspan="6">Arquitetura
da
Aplicação</td></tr>
<tr><td rowspan="3">Projeto
da
Arquitetura</td><td>Modelagem da Sociedade Multiagente</td><td>Modelo da Sociedade Multiagente</td><td rowspan="3">Modelo
da
Arquitetura</td></tr>
<tr><td>Modelagem das Interações entre Agentes</td><td>Modelo das Interações entre Agentes</td></tr>
<tr><td>Modelagem dos Mecanismos de Cooperação e Coordenação</td><td>Modelo dos Mecanismos de Cooperação e Coordenação</td></tr>
<tr><td rowspan="2">Projeto
do
Agente</td><td>Modelagem do Conhecimento e das Atividades do Agente</td><td>Modelo do Conhecimento e das Atividades do Agente</td><td rowspan="2">Modelo
do
Agente</td></tr>
<tr><td>Modelagem dos Estados do Agente</td><td>Modelo dos Estados do Agente</td></tr>
<tr><td colspan="2" rowspan="2">Implementação
da
Aplicação</td><td>Mapeamento de Agentes do Projeto à Implementação e de Responsabilidades em Comportamentos</td><td>Modelo de Agentes e Comportamentos</td><td colspan="2" rowspan="2">Modelo de Implementação da Aplicação</td></tr>
<tr><td>Mapeamento de Interações entre Agentes em Atos de Comunicação</td><td>Modelo de Atos de Comunicação entre Agentes</td></tr>
</table>

Tabela 4 Fases, subfases, passos, subprodutos e produtos da Metodologia MAEEM, considerando a Prototipação da Interface Usuário (LINDOSO, 2006)

6 ESTUDO DE CASO

6.1 Considerações Iniciais

A ferramenta PROPOST foi desenvolvida em Java (http://www.java.com), tendo como plataforma para implementação dos agentes o JADE - Java Agent DEvelopment framework (Bellifemine, 2003), que consiste em um ambiente para desenvolvimento de aplicações baseadas em agentes conforme as especificações da FIPA6 - Foundation for Intelligent Physical Agents (http://www.fipa.org). Sua modelagem foi realizada no Protégé (http://protege.stanford.edu), um ambiente CASE destinado tanto à modelagem de ontologias e aquisição de conhecimentos, como também oferece serviços de propósito geral (*armazenamento de dados, bibliotecas de componentes, tratamento de eventos*, etc).

Tendo em vista esta ferramenta possuir muitas funcionalidades, tornar-se-ia por demais extenso ao escopo deste trabalho contemplar a implementação de todas estas funcionalidades. Sendo assim, devido a considerarmos a *recomendação de soluções de software* a principal funcionalidade desta ferramenta, foi esta a funcionalidade por nós escolhida para ser implementada ao longo do referido trabalho, ficando, portanto a implementação das demais funcionalidades como sugestão para futuros trabalhos.

A implementação da *recomendação de soluções de software* foi realizada com reutilização de códigos de aplicações previamente desenvolvidas no grupo GESEC relacionadas à recomendação de páginas para usuários da Internet (Marinho, 2004) (Lindoso, 2006), tendo-se adaptado as mesmas para o contexto correspondente à funcionalidade oferecida pela ferramenta PROPOST.

Tendo em vista a impossibilidade de trabalharmos com dados reais acerca dos sistemas de informação existentes na nossa empresa devido à confidencialidade dos mesmos, optamos em apresentar um caso de uso mais simplificado, sem contudo deixar de abordar todos os passos necessários à garantia de que os resultados obtidos estivessem de acordo com as expectativas que pudessem estar associadas a um caso de teste de maior complexidade. Vale ainda frisar que, no nosso entendimento, a simplificação do referido caso de teste também contribui didaticamente para um maior entendimento do problema ora abordado, suas entradas e respectivas saídas.

6.2 Descrição do Teste Realizado

6.2.1 Dados de entrada

Para avaliar a performance da funcionalidade de *Recomendação de Soluções de Software* foram realizados vários estudos de casos, um dos quais descreveremos a seguir. Neste estudo de caso simulamos a utilização de vinte *sistemas de informação* por quinze *áreas gerenciais*, em uma empresa. Após a realização das fases de aquisição e pré-processamento dos dados nos arquivos de *logs* dos servidores, os dados resultantes deste processamento são mostrados na Figura 74.

Essa figura relaciona as *áreas gerenciais* e as respectivas *soluções de software* – neste caso, referentes a sistemas de informação, usados pelas mesmas. O objetivo do referido teste foi a recomendação de *sistemas de informação* para a *área A*, sendo esta, portanto, considerada a *área corrente*.

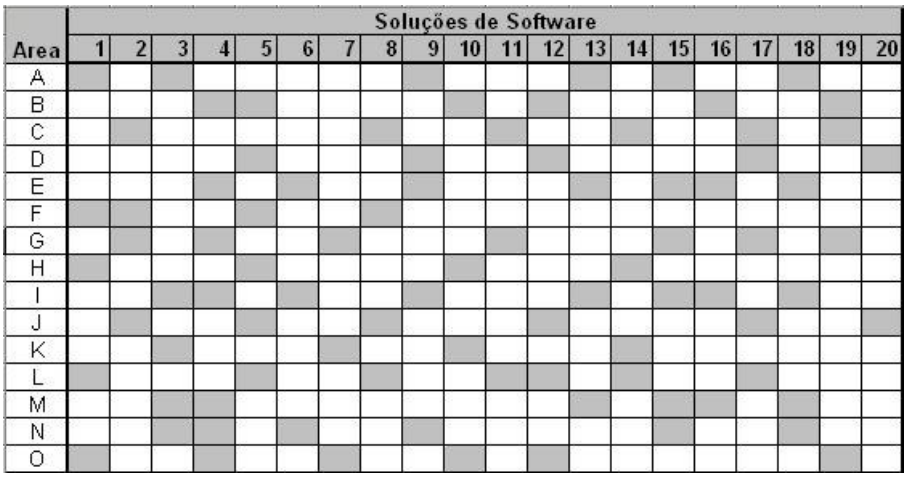


Figura 74 Exibição gráfica das *Soluções de software* usadas nas áreas

6.2.2 Solicitação da Recomendação

A Figura 75 mostra a interface para solicitações de recomendações, implementada para a aplicação. Nesta figura o usuário solicita à ferramenta que sejam recomendadas *soluções de software* para a *área corrente* "A". Essa solicitação é realizada escolhendo-se uma área na caixa de opções *Related Area* e pressionando-se o botão *Ask for Recomendation*.

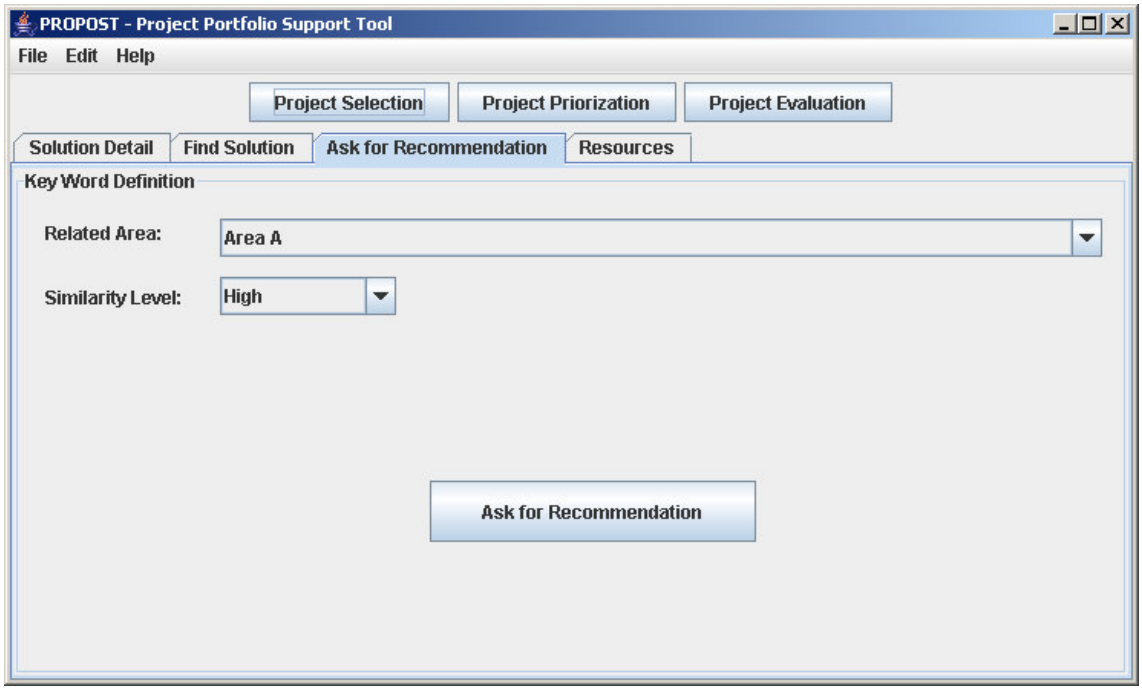


Figura 75 Interface para Solicitação de Recomendações da PROPOST

6.2.3 Trocas de Mensagens entre os agentes

Para proporcionar uma visualização gráfica das trocas de mensagens ocorridas entre os agentes ao longo do teste realizado, usaremos a ferramenta RMA - *Remote Management Agent* , que consiste em um console gráfico para o controle e gerenciamento da plataforma JADE. Esta ferramenta possui um recurso de depuração denominado *SnifferAgent*, o qual mostra a troca de mensagens graficamente em notação similar a diagramas de seqüências UML entre os agentes.

A Figura 76 exibe o console do RMA, no qual o recurso *SnifferAgent* mostra a passagem de mensagens entre o agente *interfaceador* e o agente *modelador*, onde as mesmas são representadas através de setas horizontais, cujo sentido indica sua origem e o seu destino, respectivamente. Essas mensagens equivalem às informações de acesso das *áreas gerenciais* aos sistemas de informação (equivalentes às páginas *Web* na ONTOWUM). Essas informações são repassadas ao agente *modelador* e, a partir das mesmas, esse agente pode construir o *modelo da área corrente*, o qual contém as características de uso de sistemas pela referida área.

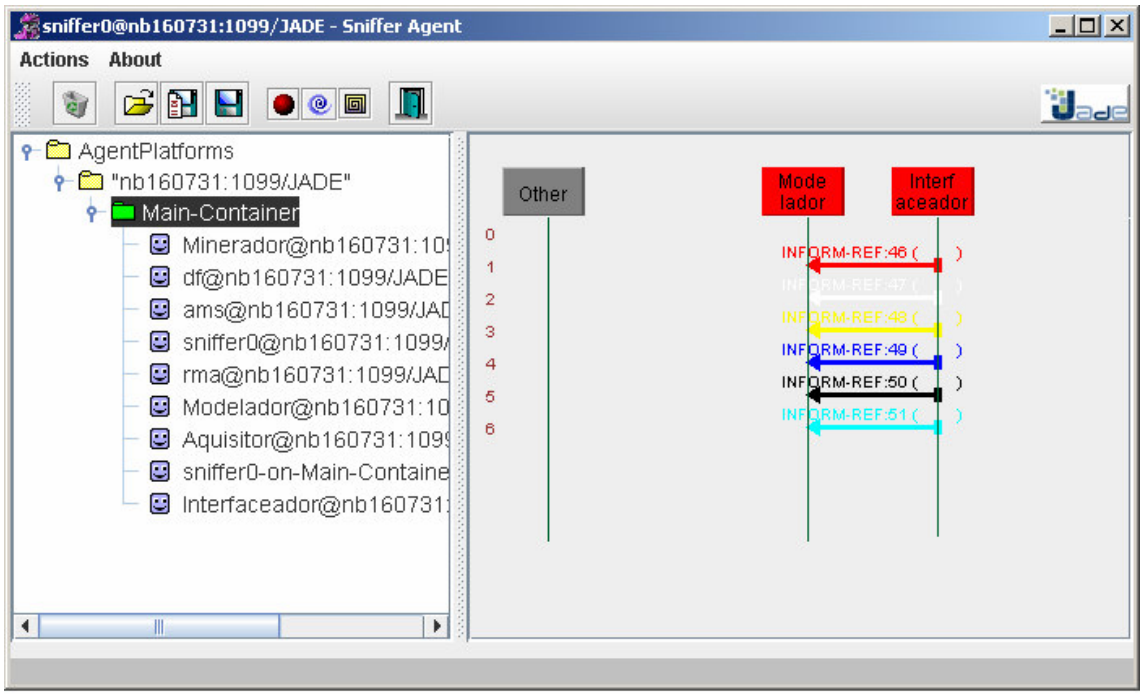


Figura 76 Troca de mensagens entre os agentes *interfaceador* e *modelador*

Para obtermos informação sobre o conteúdo das mensagens enviadas, deve-se clicar sobre as setas horizontais exibidas no diagrama.

A Figura 77 exemplifica o conteúdo de uma das mensagens enviadas pelo agente *interfaceador* ao agente *modelador*. Nesta mensagem pode-se identificar, por exemplo, qual o agente que enviou a mensagem (*sender*), qual o agente que recebeu a mesma (*receiver* ou *receivers* – caso mais de um agente esteja recebendo a mensagem), qual foi o conteúdo da mensagem (*content*) a linguagem utilizada na comunicação (*language*), a ontologia que foi utilizada na comunicação entre os agentes (*ontology*), etc.

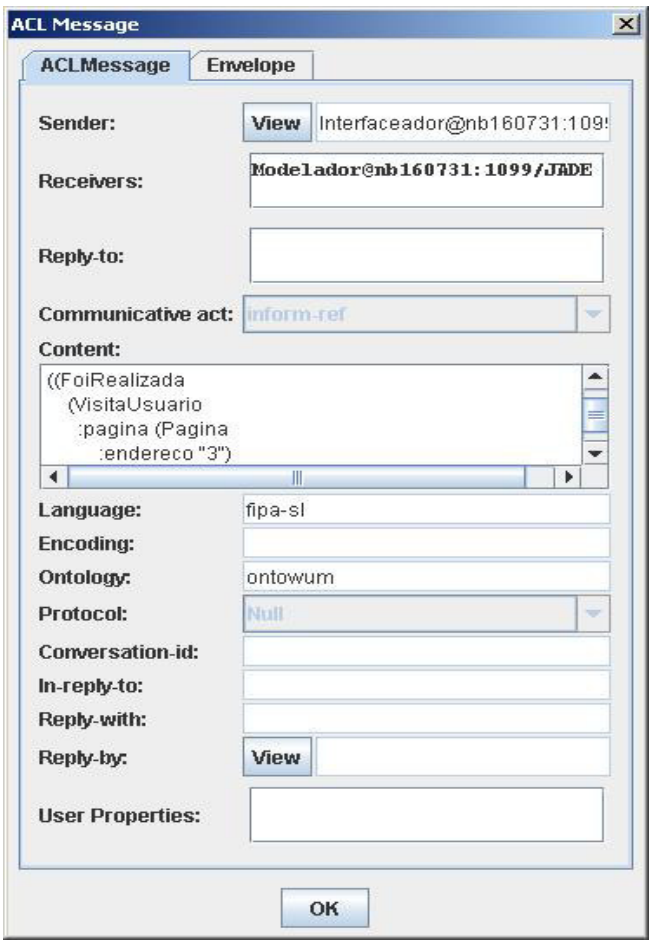


Figura 77 Mensagens do agente *interfaceador* ao agente *modelador*

Em seguida, a Figura 78 mostra a troca de mensagens ocorrida entre os agentes *modelador* e *minerador*.

A primeira mensagem é enviada do *modelador* para o *minerador*, contendo o *modelo da área corrente* (equivalente *ao* modelo de usuários na ONTOWUM*)*, conforme pode ser visto na Figura 79. Ao receber este modelo, o agente *minerador* realiza a *classificação da área corrente*, e retorna ao agente *modelador* uma mensagem contendo o modelo do *grupo de áreas com perfil semelhante* à área corrente (equivalente ao grupo de usuários na ONTOWUM), a partir do qual é realizada a *recomendação da solução de software* (Figura 80).

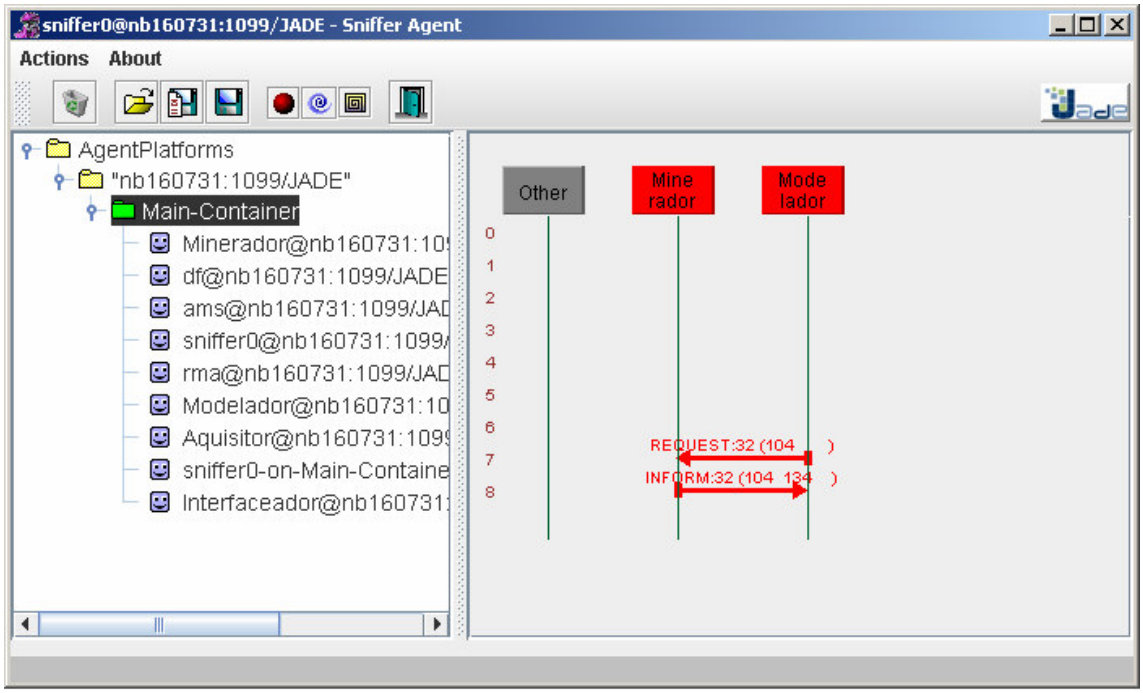


Figura 78 Troca de mensagens entre os agentes *modelador* e *minerador*

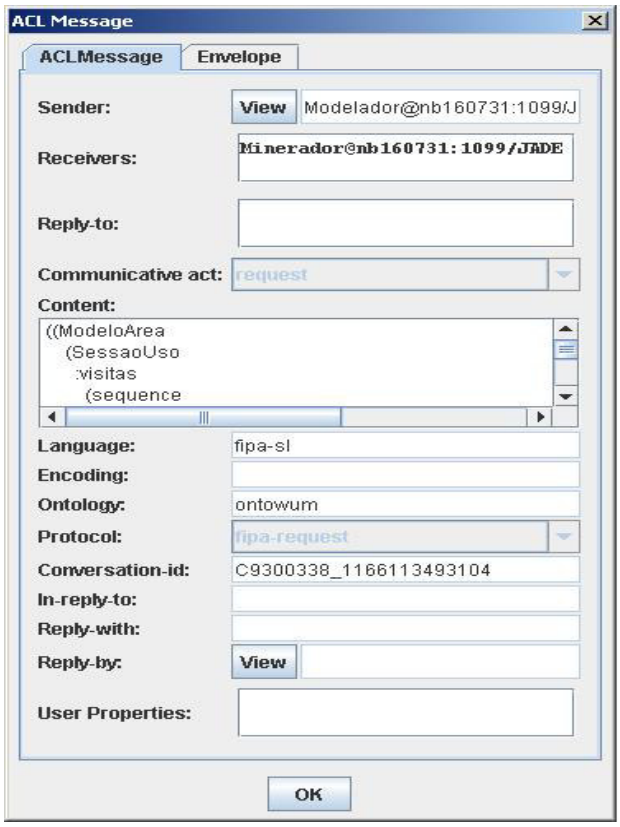


Figura 79 Mensagem do agente *modelador* ao agente *minerador*

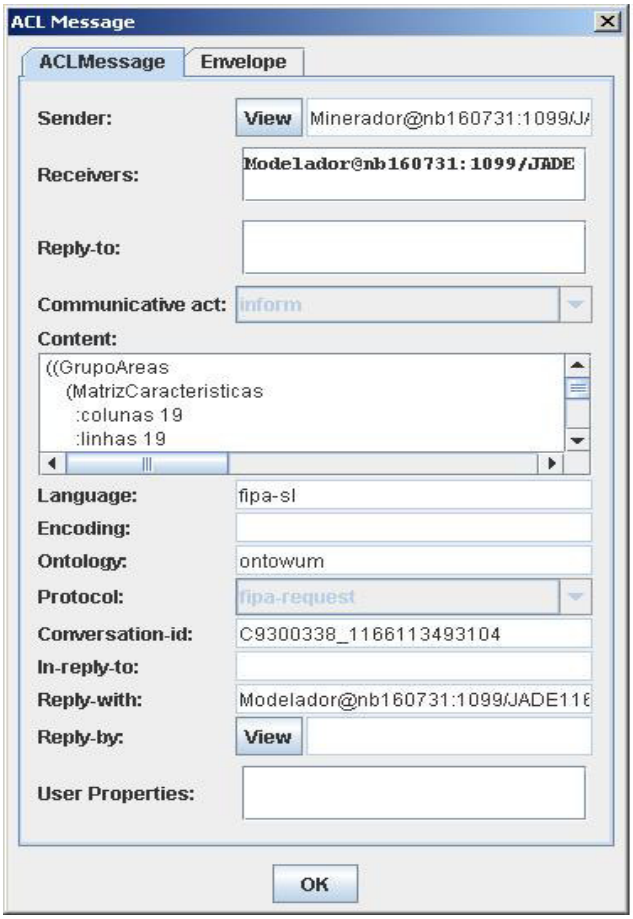


Figura 80 Mensagem do agente *minerador* ao agente *modelador*

A Figura 81 mostra a interação completa ocorrida entre os agentes, na qual pode-se observar a última mensagem enviada do agente *modelador* ao agente *interfaceador*, contendo a recomendação a ser apresentada por este agente ao usuário na interface da aplicação.

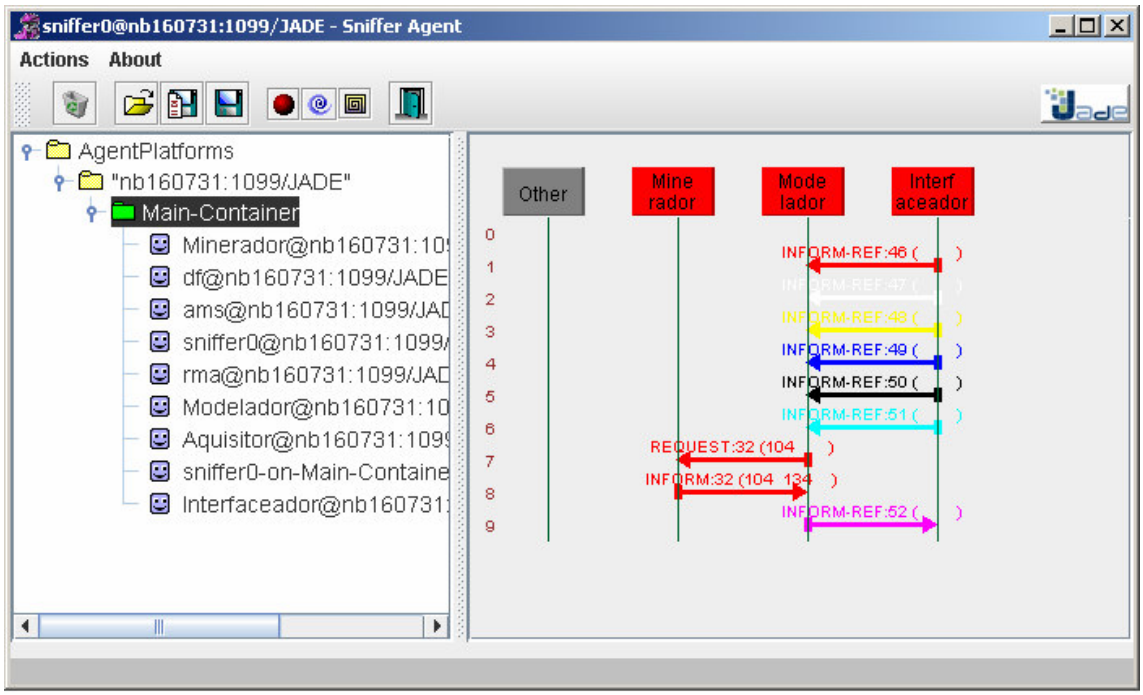


Figura 81 Troca de mensagens entre os agentes na PROPOST para geração da recomendação

.

6.2.4 Exibição da Recomendação

A exibição da *recomendação* pela ferramenta PROPOST ao usuário é mostrada na Figura 82 a seguir, na qual observa-se que as *soluções de software* recomendadas foram as equivalentes aos sistemas de informação *4, 6 e 16*, respectivamente. Na seção 6.2.5 a seguir serão abordados com maiores detalhes os motivos pelos quais a ferramenta processou a geração destas soluções.

Figura 82 Soluções recomendadas pela PROPOST

6.2.5 Conclusão dos Testes

Para melhor visualização dos resultados obtidos na recomendação, os mesmos foram relacionados em uma tabela mostrada na Figura 83. Conforme mencionado anteriormente, as recomendações de soluções de software realizadas pela ferramenta correspondem aos sistemas de informação *4, 6 e 16.*

A referida recomendação foi obtida a partir da identificação pela ferramenta do *grupo de áreas com perfil similar* à *área corrente,* o qual é formado pelas áreas E, I, M e N. E tal similaridade foi definida tomando-se por base a utilização dos *sistemas de informação 3, 9, 13, 15 e 18*, os quais são de uso comum entre a *área corrente* A e as áreas identificadas como *similares* (vizinhas) à mesma. Dessa forma, tendo em vista que essas *áreas similares* também utilizam em geral as soluções de software *4, 6* e *16*, as quais ainda não são utilizadas pela área corrente, a ferramenta decidiu por recomendar também o uso destas soluções de software, considerando que estas poderiam ser de potencial interesse para a mesma.

As constatações acima citadas podem ser mais facilmente visualizadas na Figura 83, proporcionando dessa forma sua conseqüente validação.

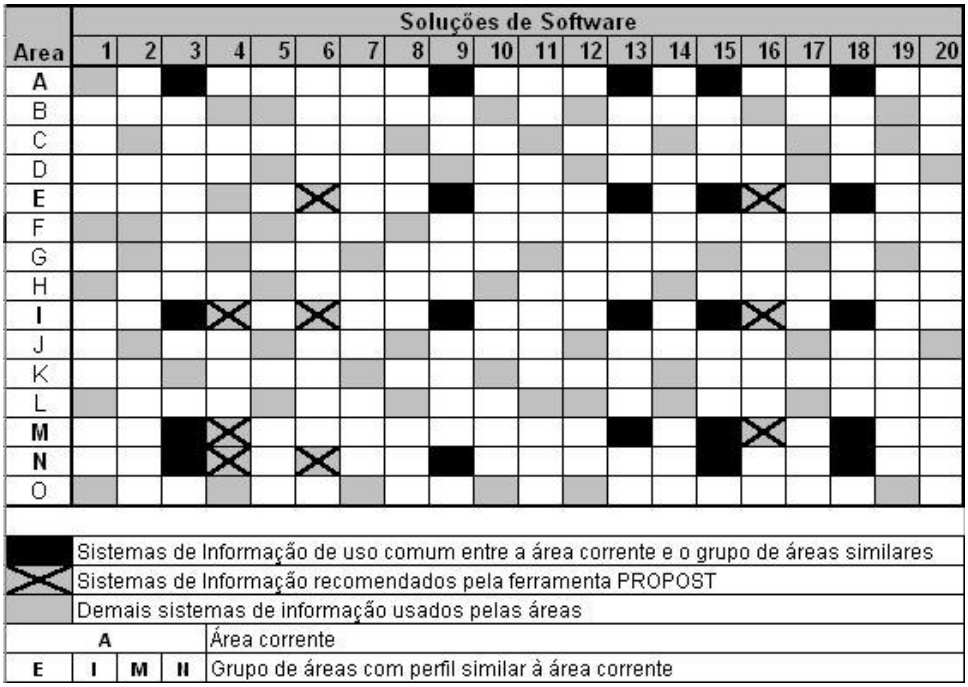


Figura 83 Exibição gráfica das Soluções recomendadas pela PROPOST

Sendo assim, devido às observações acima mencionadas, pode-se considerar que os resultados obtidos com os testes realizados para validação de performance da referida ferramenta foram satisfatórios em relação às expectativas sendo, portanto, tais resultados igualmente indicadores de sucesso na sua implementação.

7 CONCLUSÃO DO TRABALHO

Neste trabalho foi apresentada a PROPOST, uma ferramenta para suporte à Gestão de Portifólio de Projetos, que é um modelo de gestão em ascensão na atualidade. A gestão do portifólio requer um processo dinâmico que envolve, em geral, a seleção, priorização e avaliação constantes do status e valor agregado dos projetos, em relação ao negócio das organizações. Nesse contexto, o suporte oferecido pela ferramenta proposta tem o seu foco no processo da definição dos projetos a serem incorporados no referido portifólio.

O principal objetivo da PROPOST é promover a otimização de recursos através da reutilização de sistemas de informação existentes, bem como evitar duplicidade na definição de projetos para a composição do portifólio de software das empresas. Sendo assim, a concepção desta ferramenta objetivou contribuir para a solução de um problema da atualidade, relacionado à redundância na definição do portifólio de projetos, bem como suporte a outras atividades relacionadas à gestão do portifólio (seleção, priorização e avaliação), necessárias à composição do referido portifólio.

A utilização na prática dos conceitos relacionados à gestão de portifólio ainda é recente no âmbito das empresas e, devido a isso, as ferramentas de software que suportam o processo ainda estão em fase de amadurecimento em relação a determinadas funcionalidades. Porém, esse modelo vem ganhando destaque na atualidade devido à necessidade cada vez maior das empresas otimizarem a utilização de seus recursos em projetos relevantes. Nesse contexto, a reutilização de produtos existentes exerce um papel importante a ser considerado, pois o mesmo evita redundâncias no desenvolvimento de projetos similares, podendo conseqüentemente gerar economia de tempo e custo no processo de definição de novos projetos.

Devido às suas características, as ontologias são consideradas estruturas ideais para representação do conhecimento. As mesmas facilitam a captura do conhecimento armazenado na organização por possuírem alto nível representatividade semântica, bem como formalidade, reusabilidade e adaptabilidade, dentre outros. Isso proporciona tanto o aumento na produtividade quanto a qualidade na produção de software. A ferramenta PROPOST, apresentada neste trabalho, teve seu desenvolvimento dirigido com base em ontologias, sendo dessa forma um caso prático da utilização destas para o desenvolvimento de uma aplicação, cujos benefícios foram devidamente mencionados no decorrer deste trabalho.

O estudo do estado da arte das ferramentas líderes de mercado no suporte à Gestão de Portifólio mostrou que as mesmas possuem boas funcionalidades de suporte à gestão dos projetos de maneira isolada, porém o gerenciamento integrado do portifólio destes projetos ainda necessita de funcionalidades que proporcionem uma visão global mais efetiva do mesmo, para a qual poderiam estar sendo utilizados recursos mais avançados como, por exemplo, as funcionalidades de busca e recomendação apresentadas na PROPOST. As ferramentas líderes de mercado ainda não utilizam tecnologias relativas à *mineração de uso*, a *recuperação e filtragem de informações*, com vistas a um suporte mais efetivo na definição dos projetos. Devido a isso, consideramos que o fato dessas funcionalidades estarem sendo contempladas na PROPOST constitui uma inovação.

Em relação ao suporte oferecido pela ferramenta PROPOST, destacam-se as recomendações de *soluções de software* através de técnicas de *filtragem colaborativa* e *mineração de uso*, e o atendimento às necessidades de informações pontuais através de técnicas de *recuperação de informação*. Em relação à performance da ferramenta, os testes realizados para avaliação da funcionalidade escolhida para implementação (*recomendação de soluções de software*) apresentaram resultados satisfatórios e dentro das expectativas, sendo estes indicativos de que as técnicas abordadas foram utilizadas com sucesso, para o objetivo ao qual se propões o referido trabalho.

7.1 Resultados e Contribuições

Seguem abaixo as principais contribuições decorrentes dos resultados obtidos ao longo deste trabalho (NEWTON, GIRARDI, 2007):

- Modelagem de uma ferramenta de suporte ao processo de Gestão de Portifólio de projetos, a qual possui funcionalidades, dentre as quais duas são inéditas em relação a outras ferramentas similares: a Recomendação Personalizada de *Soluções de software* e a Recuperação de Informações, ambas como suporte aos processos de *Seleção, Priorização e Avaliação* de projetos. Dentre estas funcionalidades, a *recomendação de soluções de software* foi implementada, tendo esta sido avaliada satisfatoriamente através de um estudo de caso
- Adicionalmente, a PROPOST possui um *modelo de priorização de projetos* baseado em *scores* (pontuações) que visa suprir uma limitação comumente observada em modelos similares ao mesmo. Esse fato ocorre devido ao mesmo possuir a flexibilidade da definição de critérios específicos para certos projetos, e isso tem impacto relevante na definição final das pontuações dos mesmos, e suas respectivas prioridades. Geralmente os modelos existentes são essencialmente genéricos, e não possuem essa flexibilidade.
- Demonstração prática de como o desenvolvimento baseado no reuso de conhecimento representado em ontologias pode contribuir para a concepção de ferramentas de qualidade, bem como reduzir o tempo de desenvolvimento e, conseqüentemente, o custo das aplicações.
- Uma avaliação da metodologia MAAEM através da sua aplicação na construção da referida ferramenta. As observações decorrentes desta avaliação seguem abaixo:
    - Foi sugerida a fase de Prototipação da Interface com o Usuário como uma fase a ser inserida na metodologia MAAEM, a qual deverá ocorrer após a modelagem de objetivos. Tal sugestão justifica-se pelos benefícios proporcionados pela prototipação, conforme citados no capítulo referente a este tema.

- O Modelo de Papéis da metodologia MAAEM é construído com o objetivo de dar visão futura para a construção do Modelo de Sociedade Multiagente. Contudo, à proporção que a modelagem ocorre, é normal a realização de eventuais refinamentos no Modelo de Sociedade Multiagente. Muito embora achemos teoricamente correto que as modificações nesse modelo devam implicar em atualizações no Modelo de Sociedade Multiagente, achamos difícil que, na prática isso venha a ocorrer, a menos que as ferramentas gráficas que suportam essa tarefa venham a possuir recursos para a atualização automática do mesmo – fato que não ocorre no presente momento.
- Ao utilizarmos a metodologia MAAEM no desenvolvimento da ferramenta PROPOST ficou evidente a necessidade da retroalimentação de informações entre as diversas fases que compõem essa metodologia. Essa necessidade ocorre devido a estarmos utilizando um modelo de desenvolvimento interativo, no qual, à proporção que são identificadas necessidades que não estavam claras no início da modelagem, há necessidade de refletir as mesmas não somente no modelo que está sendo desenvolvido no momento, mas também nos demais modelos cuja alteração for relevante. Sugerimos que essa necessidade de retroalimentação entre as fases seja mencionada no material referente às descrições das fases da MAAEM.
- Na nossa opinião todo o conteúdo do Modelo de Conhecimento das Atividades dos Agentes, também consta no Modelo de Sociedade Multiagente. Logo, achamos desnecessário o desenvolvimento deste modelo, tendo em vista o mesmo constituir redundância de informações.
- Sugerimos que o Modelo de Papéis, bem como o Modelo da Sociedade Multiagente sejam apresentados exibindo as responsabilidades na mesma ordem em que estas são relacionadas no Modelo de Objetivos. Tal sugestão ocorre devido ao fato das responsabilidades constituírem pontos em comuns entre esses modelos e, uma vez que estas sejam apresentadas na mesma ordem de ocorrência em ambos os modelos, ao nosso ver torna-se mais fácil relacionar os mesmos.

- Análise do estado da arte das ferramentas de suporte à Gestão de Portifólio de Projetos. Essa análise serviu para a consolidação dos conhecimentos sobre o assunto, constituindo dessa forma uma massa crítica para a definição de funcionalidades relevantes ao domínio em questão, a serem contempladas pela ferramenta apresentada.
- Análise do estado da arte do modelo de Gestão de Portifólio de Projetos, o qual está em ascensão na atualidade. Porém, tendo em vista o mesmo ser de uso recente, o mesmo ainda é muito confundido com outros modelos, principalmente com a gestão de múltiplos projetos. Acreditamos que este trabalho contribua para a divulgação e melhor entendimento do referido modelo e suas características.
- Demonstração prática de como tecnologias de vanguarda, tais como Recuperação e Filtragem da Informação, *Mineração de uso*, desenvolvimento baseado em agentes, dentre outras podem contribuir para a resolução de um problema da atualidade referente à redundância na definição do portifólio, bem como suporte a outras atividades relacionadas à gestão do portifólio (seleção, priorização e avaliação), necessárias à composição do referido portifólio. Achamos válida essa abordagem, tendo em vista que a maioria das ferramentas informatizadas de uso comercial ainda não exploram essas tecnologias.

7.2 Perspectivas Futuras

Tendo em vista ser a Gestão de Portifólio um tema por demais extenso e um vasto campo a ser explorado, a ferramenta proposta buscou contemplar as funcionalidades consideradas por nós mais relevantes em relação ao domínio em questão. Mesmo assim, a implementação prática de todas estas funcionalidades seria por demais extensa para o escopo deste trabalho. Dessa forma, realizamos a implementação da funcionalidade relativa à *Recomendação de Soluções de Software*, tendo em vista ser a mesma a mais relevante das funcionalidades apresentadas nesta ferramenta, pois está mais diretamente relacionada ao principal objetivo ao qual se pretende alcançar.

Reconhecemos que tal ferramenta ainda pode ser aperfeiçoada e, sendo assim, sugerimos como trabalhos futuros tanto a implementação das funcionalidades restantes, bem como a eventual inclusão de outras funcionalidades que possam ser consideradas relevantes ao referido contexto. Dentre estas, poderão ser elaboradas novas formas gráficas de elaboração dos resultados, bem como a inclusão de relatórios para exibição dos mesmos.

Sugerimos também o aperfeiçoamento da funcionalidade de recomendação, para que a mesma possa adotar um modelo híbrido, o qual contemple também recomendações a partir da filtragem baseada no conteúdo das descrições dos sistemas usados nas áreas.

As sugestões anteriormente citadas decorrem do fato que a Gestão de Portifólio de Projetos é um modelo em ascensão no âmbito das empresas, sendo considerada uma tendência de mercado com boas perspectivas de exploração e retorno. Outra justificativa seria a continuidade da divulgação dos benefícios decorrentes da utilização de tecnologias de vanguarda, tais como as mencionadas ao longo deste trabalho.

REFERÊNCIAS

ADOMAVICUS, Gediminas. TUZHILIN, A. **Toward next generation of Recommender Systems: a survey of the state-of-the-art and possible extensions.** IEEE Transactions on knowledge and data engineering, vol. 17, no. 6, 2005.

BAEZA, Ricardo Yates and RIBEIRO NETO, Berthier. **Modern Information Retrieva**l. Addison Wesley, Harlow, 2000.

BALABANOVIC, M., & SHOHAM, Y. **Fab: Content-Based, Collaborative Recommendation**, Communications of the ACM, v. 40, n. 3, pp. 66-72, 1997.

BARROCA L., GIMENES I., HUZITA E. **Desenvolvimento Baseado em Componentes: Conceitos e Técnicas**. Editora Ciência Moderna, Maringá, pps 2 e 59, 2003.

BAUER, Michael G.; SPECHT, Gunther. **Enhancing Digital Library Documents by Aposteriori Cross Linking Using XSLT**. Institut fur Informatik, TU Munchen Lecture Notes in Computer Science, 2001.

BELLIFEMINE, F., CAIRE, G., POGGI, A., RIMASSA, G., 2003. **JADE A White Paper**. Exp v. 3 n. 3. http://jade.tilab.com/.

BOOCH, G., RUMBAUGH, J. and JACOBSON, I. **Unified Modeling Language User Guide**. Reading, Addison Wesley, 1999.

BELKIN, N. J. & CROFT, W. B. **Information filtering and information retrieval: Two sides of the same coin ?**, Communications of the ACM, v. 35, n. 12, pp. 29-38, 1992.

BURKE, R. **Hybrid Recommender Systems: survey and experiments. User Modeling and User-Adapted Interaction**, 2002.

BURKE, R, **Knowledge-Based Recommender Systems**, Encyclopedia of Library and Information Systems, A. Kent, ed., vol. 69, Supplement 32, Marcel Dekker, 2000.

CAZELA, Silvio Cesar, REATEGUI, Eliseo Berni. **Sistemas de Recomendação**. Capitulo 1. Dissertação de mestrado. UFRGS, 2004.

CHEN, M., PARK, J.S., YU, P.S. **Efficient data mining for path traversal patterns**, IEEE Transactions on Knowledge and Data Engineering, v.10, n. 2 (Mar), pp. 209-221, 1998.

CHANDRASEKARAN, B. AND JOSEHSON, J. **What are Ontologies, and Why Do We Need Them?**, IEEE Intelligent Systems, Vol 14, nº1, pp. 20 – 26, 1999.

CHOO, W. C. **The knowing organization: how organizations use information to construct meaning, create knowledge, and make decisions**. New York: Oxford University Press, 2000.

COOLEY, R., MOBASHER, B., SRIVASTAVA, J. **Web mining: information and pattern Discovery on the World Wide Web** , Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence, 1997.

COOPER, R. G., Edgett, S. J. and KLEINSCHMIDT, E. J. **Portfolio Management for New Products**, 2dn edn, Perseus Publishing, NY, 2001.

DAHLEN, B. J., KONSTAN, J. A., HERLOCKER, J. L., et al. **Jump-starting Movielens: User benefits os starting a collaborative system dead data**, University of Minnesota, TR 98-017, 1998.

DYE, Lowell D.,PENNYPACKER, James S. **Project Portfolio Management and Managing Multiple Projects-Two Sides of the Same Coin,** Proceedings of the Project Management Institute Annual Seminars & Symposium, Houston, 2003.

FALBO, R. A., GUIZZARDI G., DUARTE, K. C. **An Ontological Approach to Domain Engineering**. Proceedings of the XIV International Conference on Software Engineering and Knowledge Engineering (SEKE 2002), ACM Press, pp. 351-358. Ischia, Italy, 2002.

FERREIRA, Stefferson Lima; GIRARDI, María Del Rosario. **Arquiteturas de Software Baseadas em Agentes: Do Nível Global ao Detalhado**. Revista Electrônica de Iniciação Científica, v. II, n. II, 2002.

FREEMAN P., Editor. **Tutorial: Software Reusability**, IEEE Catalog Number EH0256-8, IEEE Computer Society Press, Washington, D.C., 1987.

FARIA, Carla Gomes de. **Uma Técnica para a Aquisição e Construção de Modelos de Domínio e Modelos de Usuários Baseados em Ontologias para a Engenharia de Domínio Multiagente**. Dissertação de Mestrado, Universidade Federal do Maranhão (CPGEE/UFMA). São Luís-MA, 2004.

FORRESTER RESEARCH. **The Forrester Wave™: Project Portfolio Management**, March 2006 by Margo Visitacion with Craig Symons, Lindsey Hogan, and Andrew Sahalie - Forrester, Forrester Research, Inc, 2006.

FREITAS, Celso C. C.; MOURA, Hermano P. **GMP: Uma Ferramenta para a Gestão de Múltiplos Projetos**. Artigo publicado no Simpósio Brasileiro de Sistemas de Informação - SBSI 2004.

HERLOCKER, J. **Understanding and Improving Automated Collaborative Filtering Systems.** Dissertação de doutorado, University of Minnesota, USA, 2001.

HERLOCKER, J., Konstan, J.A., Terveen, L.G., Riedl, J.T. **Evaluating collaborative filtering recommender systems**. ACM Transactions on Information Systems, 2004.

GESEC – **GRUPO DE PESQUISA EM ENGENHARIA DE SOFTWARE E ENGENHARIA DO CONHECIMENTO**, disponível em http://maae.deinf.ufma.br, acessado em 01-03-2006.

GIRARDI, Rosario. **Engenharia de Software baseada em Agentes**. Anais do IV Congresso Brasileiro de Ciência da Computação (CBCOMP 2004), Ed. UNIVALI, pp. 913-937, Itajai, Santa Catarina, Brasil. 08 a 12 de outubro de 2004.

GIRARDI, Rosário LINDOSO, Alisson. (2005). **An Ontology-based Methodolgy for Multi-agent Domain Engineering**. 3RD WORKSHOP ON MULTI-AGENT SYSTEMS: THEORY AND APPLICATIONS (MASTA 2005) AT 12TH PORTUGUESE CONFERENCE ON ARTIFICIAL INTELLIGENCE (EPIA 2005), Ed. IEEE. Covilhã, Portugal. 05 a 08 de dezembro de 2005.

GIRARDI, Rosário, BALBY, Leandro, OLIVEIRA, Ismênia. **A System of Agent-based Patterns for User Modeling based on Usage Mining**, Interacting with Computers, v. 17, n.5, Ed. Elsevier, pp. 567-591. Setembro de 2005.

GIRARDI, Rosario, LINDOSO, Alisson. **DDEMAS: A Domain Design Technique for Multi-agent Domain Engineering.** Lecture Notes in Computer Science, Perspectives in Conceptual Modeling: ER 2005 Workshops CAOIS, BP-UML, CoMoGIS, eCOMO, and QoIS, Volume 3770/2005, ISSN 0302-9743., Ed. Springer-Verlag GmbH, pp. 141-150. Klagenfurt, Austria. 24 a 28 de outubro de 2005.

GIRARDI, Rosario, BALBY, Leandro. **A Domain Model of Web Recommender Systems based on Usage Mining and Collaborative. Filtering**, - Springer-Verlag London Limited, 2006.

GIRARDI, Rosário. **Recuperação da Informação e Sistemas Multimídia**. Notas de aula. Universidade Federal do Maranhão, 2005.

GOLDBERG, N., NICHOLS, D., OKI, B. M., et al. **Using collaborative filtering to weave information Tapestry**, Communications of the ACM, v. 35, n. 12, pp. 61-70, 2004.

GRUBER, T. R . **Toward Principles for the Design of Ontologies used for Knowledge Sharing**, International Journal of Human-Computer Studies. Nº 43, pp. 907-928, 1995.

HUHNS, N., and STEPHENS, L. M. **Multi-Agent Systems and Societies of Agents**, In: Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence, G. Weiss (ed.), The MIT Press, 1999.

JANSEN PEREIRA, Mauro H. **Uma metodologia e uma ferramenta para o reuso**. Dissertação (Mestrado em Engenharia de Eletricidade – Área de Ciência da Computação), Universidade Federal do Maranhão (CPGEE/UFMA). São Luís-MA, 2006.

JENNINGS, N.,. **On Agent-based Software Engineering**. Artificial Intelligence, 117, 277-296., 2000.

KAPLAN, Robert S., NORTON David P., **A Estratégia em Ação**.Rio de Janeiro,1997.

KENDALL, GERALD I., **Executive Guide to Project Portfolio Management**, TOC International, FL 2004.

KERZNER, Harold. **A Systems Approach to Planning, Scheduling, and Controlling**, Project Management – New York NY, John Willey & Sons, 2001.

KRUEGER, C. W. **Software Reuse**. ACM Computing Surveys, 24(2):, June 1992.

LANCASTER, Frederick Wilfrid and WARNER , Amy J. **Information Retrieval Today**. Information Resources Press, Arlington, Virginia, 1998.

LASSILA, O. AND SWICK, R., **Resource Description Framework (RDF) Model and Syntax Specification**. W3C recommendation, World Wide Web Consortium, 1999.

LAWRENCE, S. "Context in Web Search", IEEE Data Engineering Bulletin, v. 23, n. 3, pp. 25-32, 2000.

LEVINE, H. A., **Project Portfolio Management: A Practical Guide to Selecting Projects, Managing Portfolios and Maximizing Benefits**, Jossey-Bass, CA, 2005

LINDOSO, Alisson Neres.**Uma Metodologia baseada em Ontologias para a Engenharia de Aplicações Multiagente**. Dissertação (Mestrado em Engenharia de Eletricidade – Área de Ciência da Computação), Universidade Federal do Maranhão (CPGEE/UFMA). São Luís-MA, 2006.

LINDOSO, Alisson, GIRARDI, Rosario. **Uma Técnica baseada em Ontologias para o Reuso de Padrões de Software e de Frameworks no Projeto de Aplicações Multiagente.** First Workshop on Software Engineering for Agent-oriented Systems (SEAS 2005), 19º Simpósio Brasileiro de Engenharia de Software (XIX SBES). Uberlândia, Minas Gerais, Brasil. 03 de outubro de 2005.

MARINHO, Leandro Balby. **Um Framework Multiagente para a Personalização da Web baseado na Modelagem de Usuários e na Mineração De Uso**. Dissertação (Mestrado em Engenharia de Eletricidade – Área de Ciência da Computação), Universidade Federal do Maranhão (CPGEE/UFMA). São Luís-MA, 2004.

McGRATH, Michael E; ANTHONY, Michael T; SHAPIRO, Amram R. **Product development: success through product and cycle-time excellence**. Newton: Butterworth-Heinemann, 1998.

MIZZARO, S. **Relevance: the whole history,** Journal of American Society for Information Science, v. 48, n.9, pp. 810-832, 2000.

MIDDLETON, S.E., ALANI, H., SHADBOLT, N.R., et al. **Exploiting Synergy Between Ontologies and Recommender Systems**. In: proceedings of the Eleventh International World Wide Web Conference (WWW2002, Hawaii, USA, 2002.

MOBASHER, B., COOLEY, R., SRIVASTAVA, J., **Creating adaptive Web sites through usage based clustering of URLs**, In: Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99), 143-153, 1999.

MORO, R.D., NARDI J.C., Falbo, R.A. **ControlPro: Uma Ferramenta de Acompanhamento de Projetos Integrada a um Ambiente de Desenvolvimento de Software**. XII Sessão de Ferramentas do Simpósio Brasileiro de Engenharia de Software - SBES 2005.

NEWTON, Eduardo; GIRARDI, Rosario. **PROPOST: A knowledge-based tool for supporting Project Portfolio Management**, Ed. IEEE (to appear). URL: dori2.technion.ac.il/ICSEM07/. Março de 2007.

NONAKA, Ikujiro, TAKEUCHI, Hirotaka. **Criação de Conhecimento na Empresa. Como as empresas japonesas geram a dinâmica da inovação**. Rio de Janeiro, 1997.

NOY, N.; Mcguiness, D. **Ontology Development 101: A guide to creating your first ontology**. KSL Technical Report, Standford University, 2001.

PATTERSON, Marvin L. **Leading product innovation: accelerating growth in a product-based business**, New York: John Wiley & Sons, 1999.

PEZZIN, J., FALBO, R. A. **AgeODE: Uma Infra-estrutura para Apoiar a Construção de Agentes para Atuarem no Ambiente de Desenvolvimento de Software ODE**. Actas de las IV Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento - JIISIC 2004.

PIERRAKOS, D., PALIOURAS, G., PAPATHEODOROU, C., SPYROPOLOUS, C. D., **"Web Usage Mining as a Tool for Personalization: A Survey"**, User Modeling and User-Adapted Interaction 13: 311-372, 2003.

PMBOK 2004, **A Guide to Project Management Body of Knowledge**, Project Management Institute - PMI®, 2004.

PORTER, Michael E. **Competição = On Competition**: **estratégias competitivas essenciais**, Rio de Janeiro: Campus, 1999.

PORTER, M.**, Competitive Strategy.** Editora Simon & Schuster. Brochura, 2003.

PRESSMAN, Roger. **Software Engineering – a Practitioner's Approach**. 5th Edition. New York: McGraw-Hill, 2001.

PROTÉGÉ Project. http://protege.stanford.edu. Acesso em: 08 de janeiro de 2006.

RESNICK, P. & VARIAN, H. **Recommender System**, Communications of the ACM, v. 40 n.3, pp. 56-58, 1997.

RIZZI, Claudia Brandelero, et al.**Fazendo uso da categorização de textos em atividades empresariais**. Instituto de Informática, Universidade Federal do Rio Grande do Sul. Mar, 2003.

SANTOS, A. R. et al., **Gestão do Conhecimento como modelo empresarial**. In: SANTOS, A. R. et al. Gestão do conhecimento: uma experiência para o sucesso empresarial. Curitiba: Champagnat, 2001.

SCHAFER, J. B., KONSTAN, J. A., RIEDL, J. **E-Commerce Recommendation Applications**. In: proceedings of Data Mining and Knowledge Discovery, pp.115-153, 2001.

SHAHABI, C., BANAEI-KASHANI, F., **Efficient and Anonymous Web Usage Mining for Web Personalization**, INFORMS Journal on Computing - Special Issue on Data Mining, Vol.15, No.2, Spring, 2003.

SHARDANAD, U. & MAES, P. **Social information filtering: Algorithms for automating "word of mouth".** In: Proceedings of the Conference on Human Factors in Computing Systems – CHI 95, Denver, pp. 210-127, 1995.

SILVA, A. P. A., **O que falta para o GED decolar definitivamente?** Jornal Mundo da Imagem, no 56, Mar/Abr. São Paulo: Cenadem, 2003.

SILVEIRA -Maria de Lourdes, NETO, RIBEIRO NETO - Berthier, VALE DE FREITAS -Rodrigo. **Vertical searching in juridical digital libraries**. In Proceedings of the 25th European Conference on Information Retrieval Research ECIR, 2003.

STEWART, T. A., **Capital intelectual: a nova vantagem competitiva das empresas**. Rio de Janeiro: Campus, 1998.

SVEIBY, K. E. **A nova riqueza das organizações: gerenciando e avaliando patrimônios de conhecimento**. Rio de Janeiro: Campus, 1998.

TERVEEN, L. & HILL, W. **Beyond Recommender System: Helping People to Find Each Other**. In: Proceedings of HCI in the New Millennium, Jack Caroll, ed., Addison-Wesley, 2003.

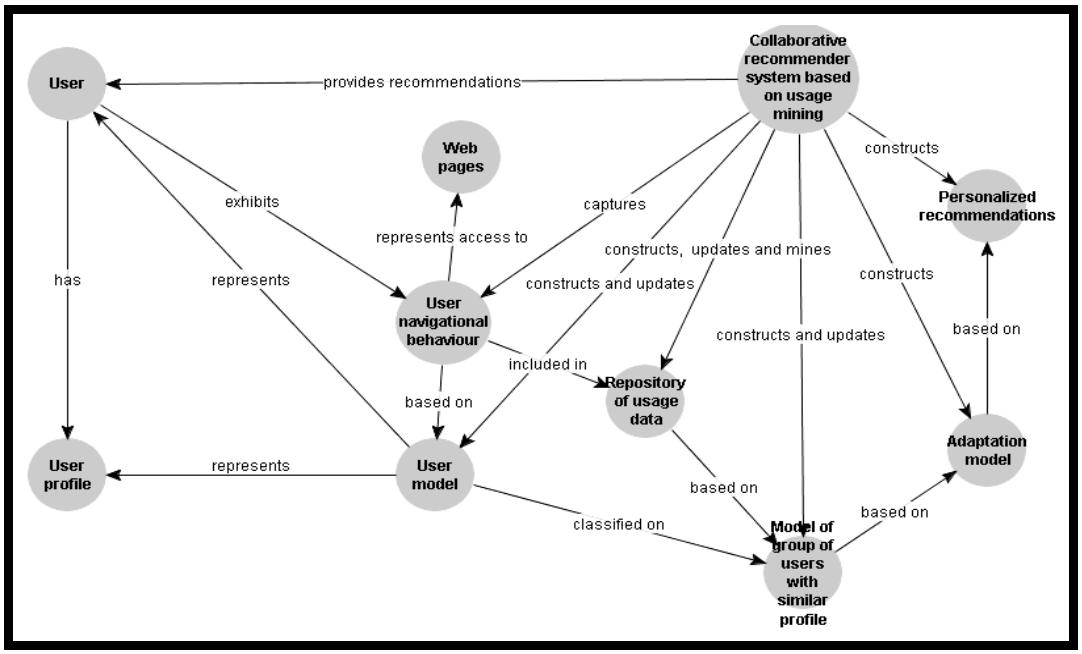
TORRES, Roberto. **Personalização na Internet**. Novatec Editora: São Paulo, 2004

V. Basili, D. Rombach (1988). **The TAME Project – TowardsImprovement-Oriented Software Environments**. IEEE Trans-actions on Software Engineering 14, 6 , 1988

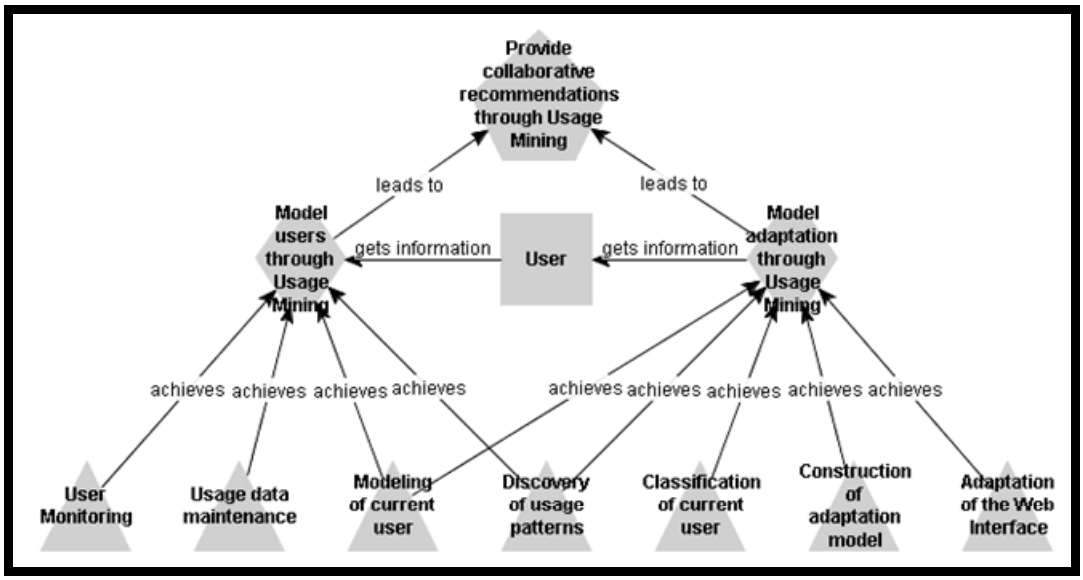
YELIN, K. C. **Managing the Business as a Portfolio of Projects**. Resumo sobre apresentação no ProjectWorld: Boston, 1999.

WHEELWRIGHT, Steven C; CLARK, Kim B. **Creating project plans to focus product development**. Harvard Business Review, Boston; mar./apr. 1999.

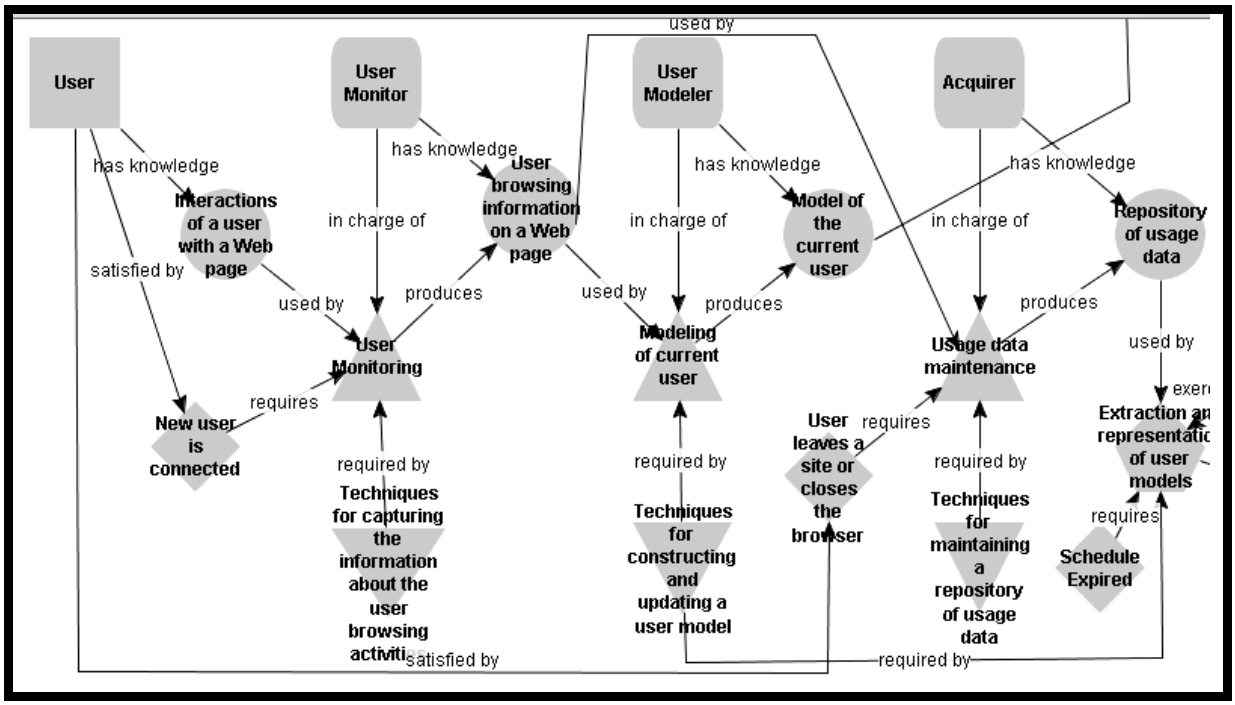
APÊNDICE A – Instâncias da modelagem da ONTOWUM: Modelo de Domínio (*DM*) e *Framework* Multiagente (*DD*)



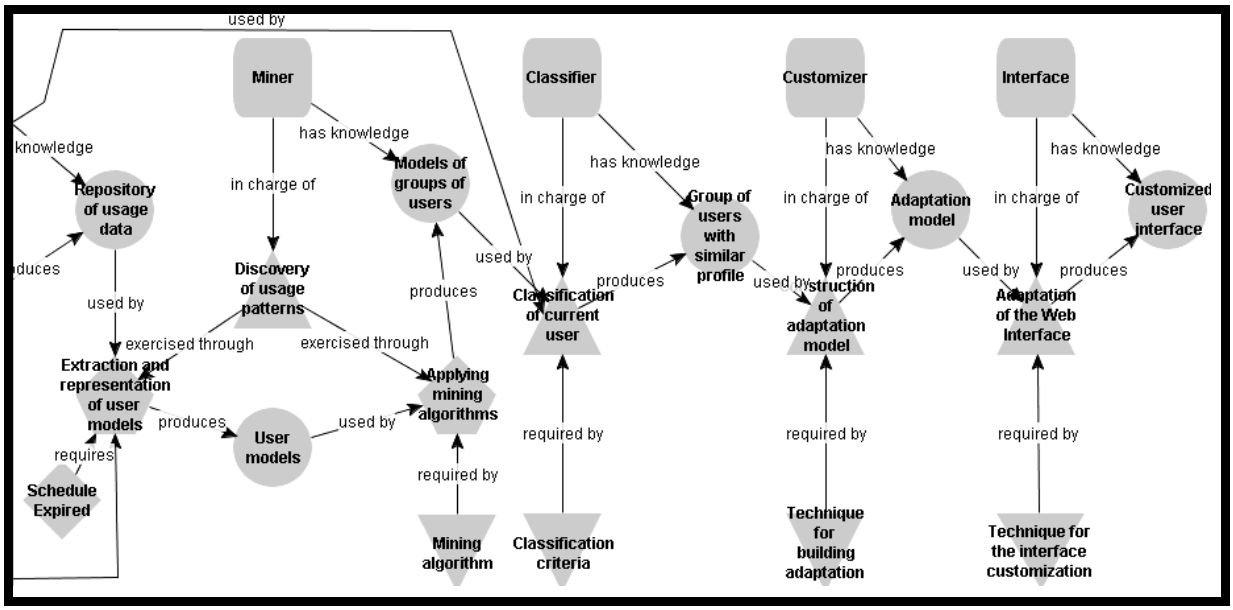
Modelo de Conceitos da ONTOWUM-DM (GIRARDI, BALBY, 2006)



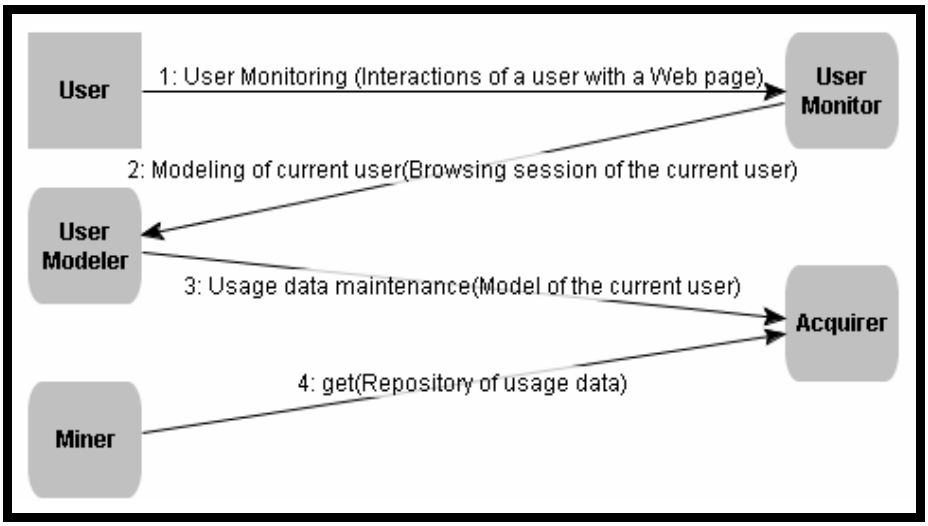
Modelo de Objetivos da ONTOWUM-DM (GIRARDI, BALBY, 2006)



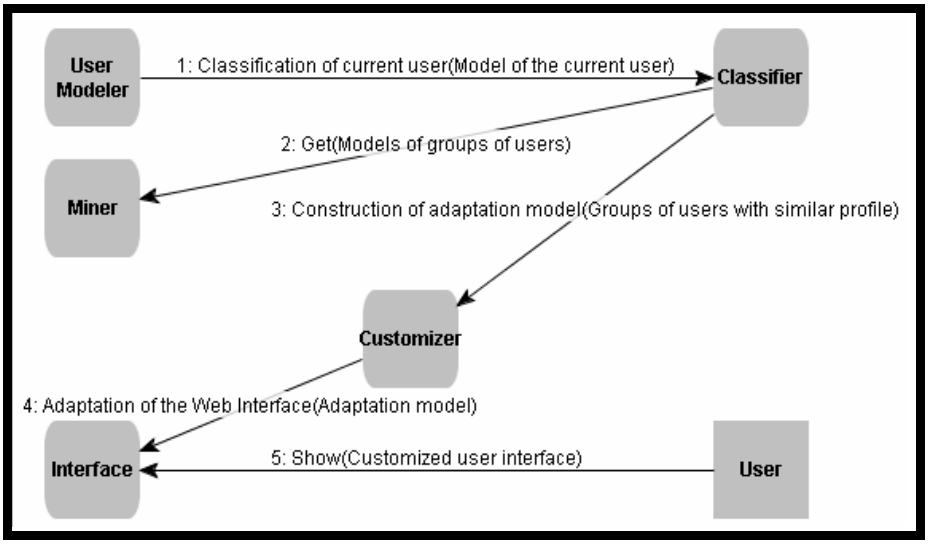
Modelo de Papéis da ONTOWUM-DM mostrando os papéis
Monitor de Usuário, *Modelador de Usuário* e *Aquisitor* (GIRARDI, BALBY, 2006)



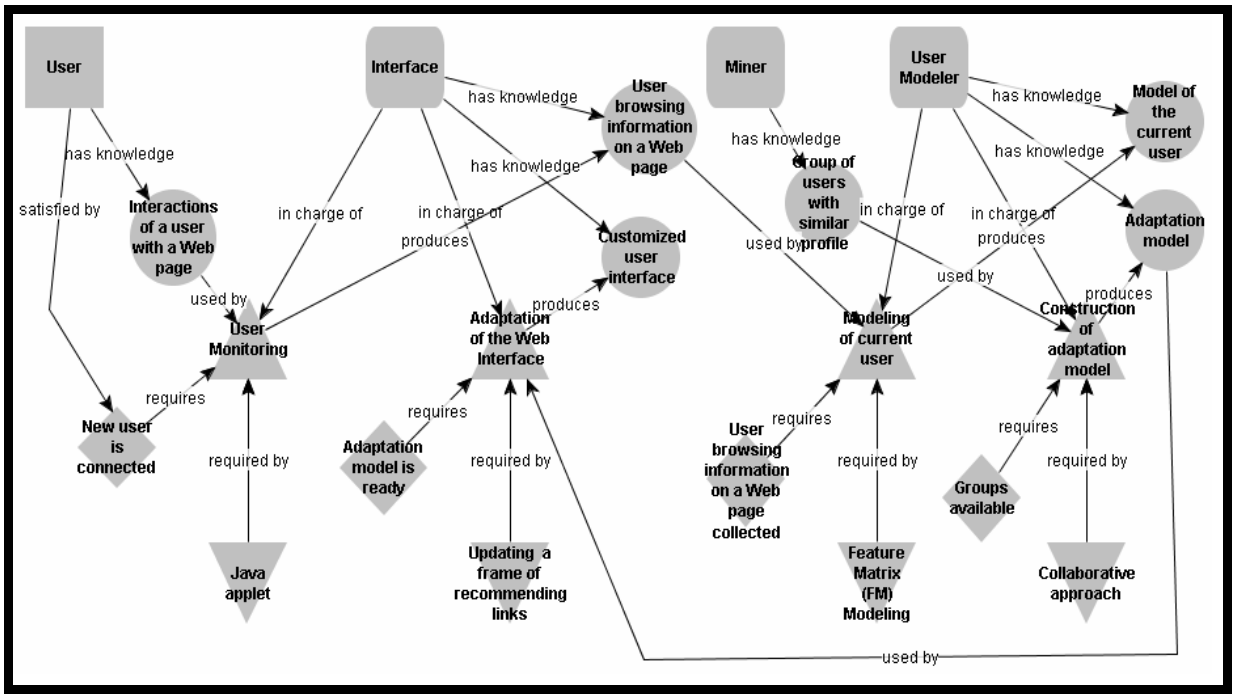
Modelo de Papéis da ONTOWUM-DM mostrando os papéis
Minerador, *Classificador*, *Personalizador* e *Interfaceador* (GIRARDI, BALBY, 2006)



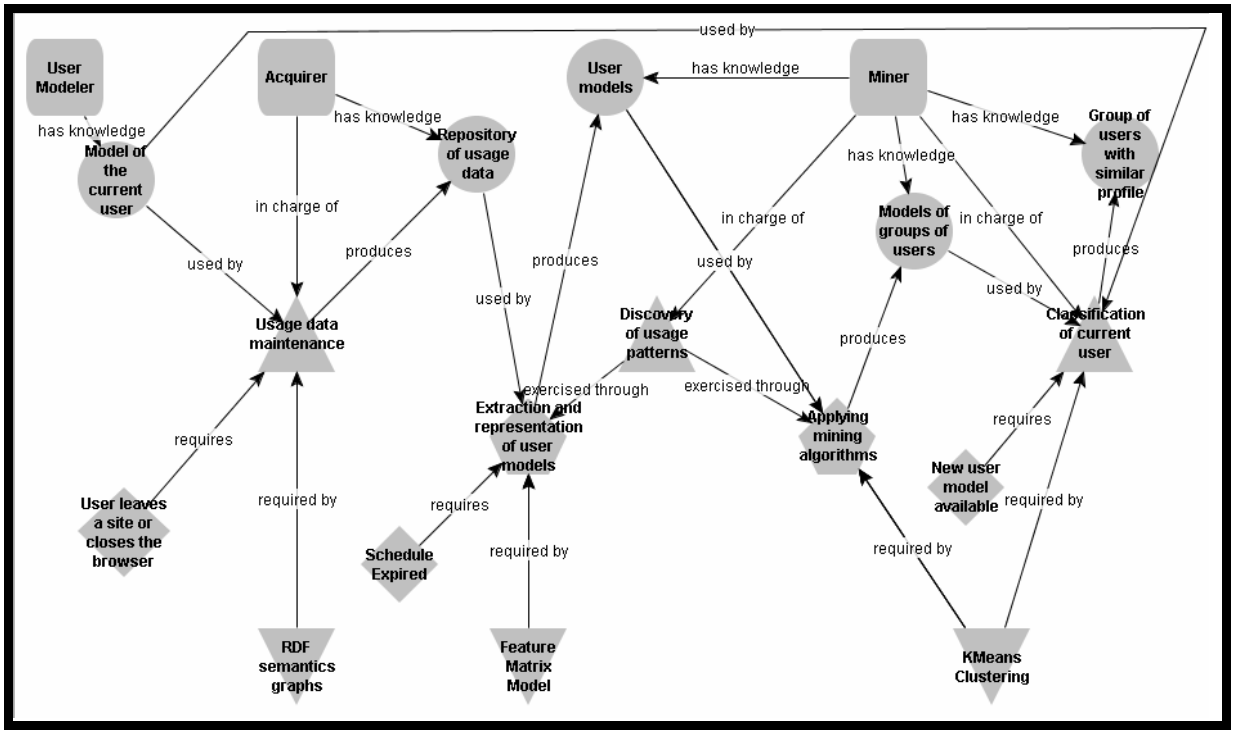
Modelo de Interações entre Papéis da ONTOWUM-DM relativo ao objetivo específico *Modelar usuários através de mineração de uso* (GIRARDI, BALBY, 2006)



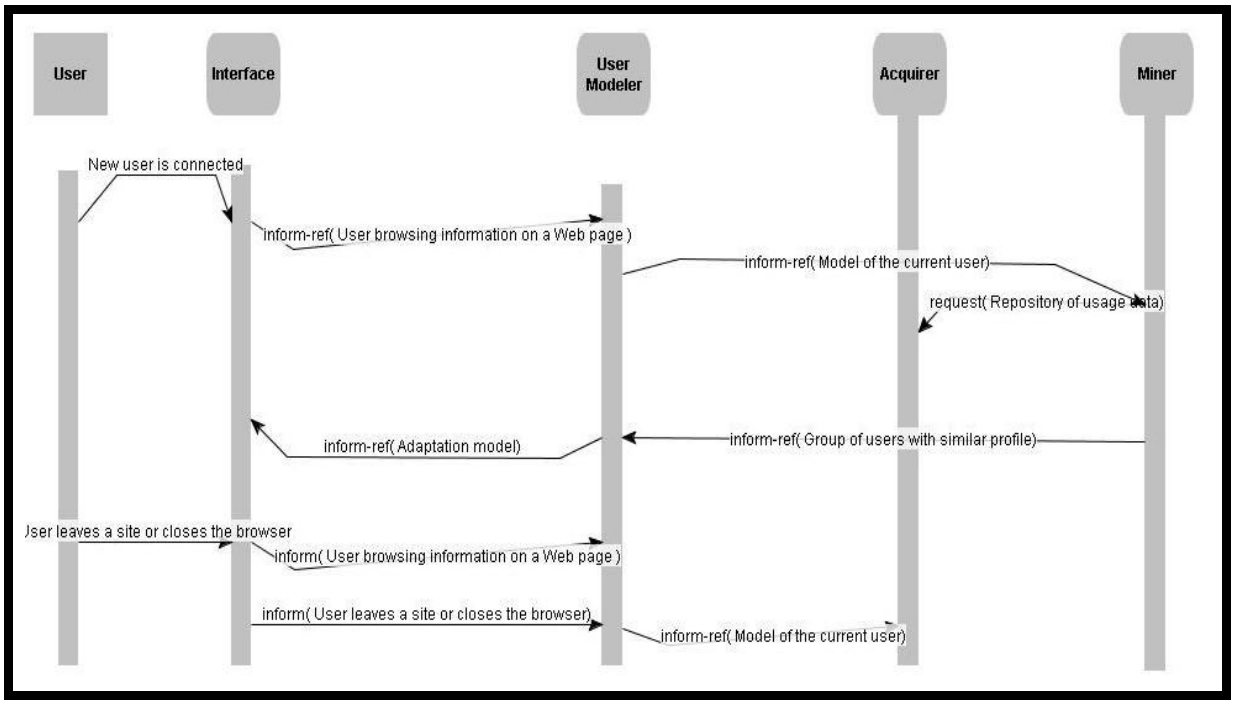
Modelo de Interações entre Papéis da ONTOWUM-DM relativo ao objetivo específico *Modelar adaptação através de mineração de uso* (GIRARDI, BALBY, 2006)



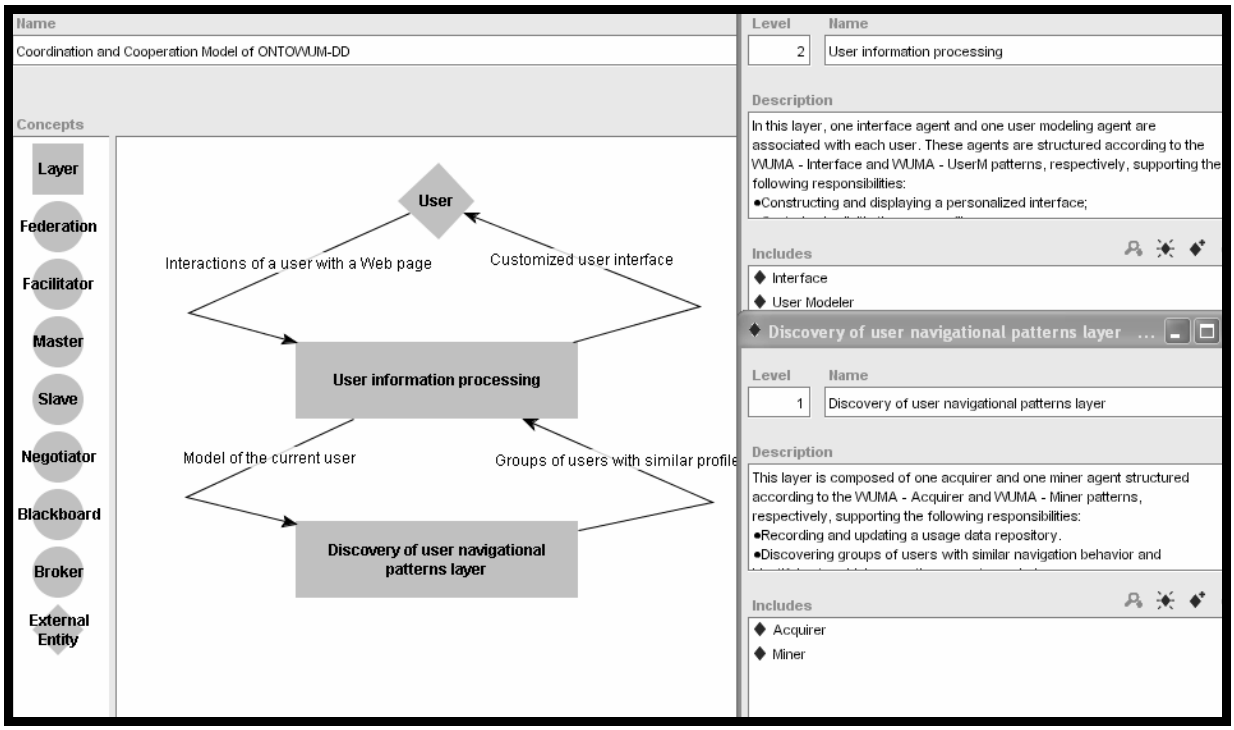
Modelo da Sociedade Multiagente da ONTOWUM-DD mostrando os agentes *Interfaceador* e *Modelador* de *Usuário* (GIRARDI, BALBY, 2006)



Modelo da Sociedade Multiagente da ONTOWUM-DD mostrando os agentes *Aquisitor* e *Minerador* (GIRARDI, BALBY, 2006)

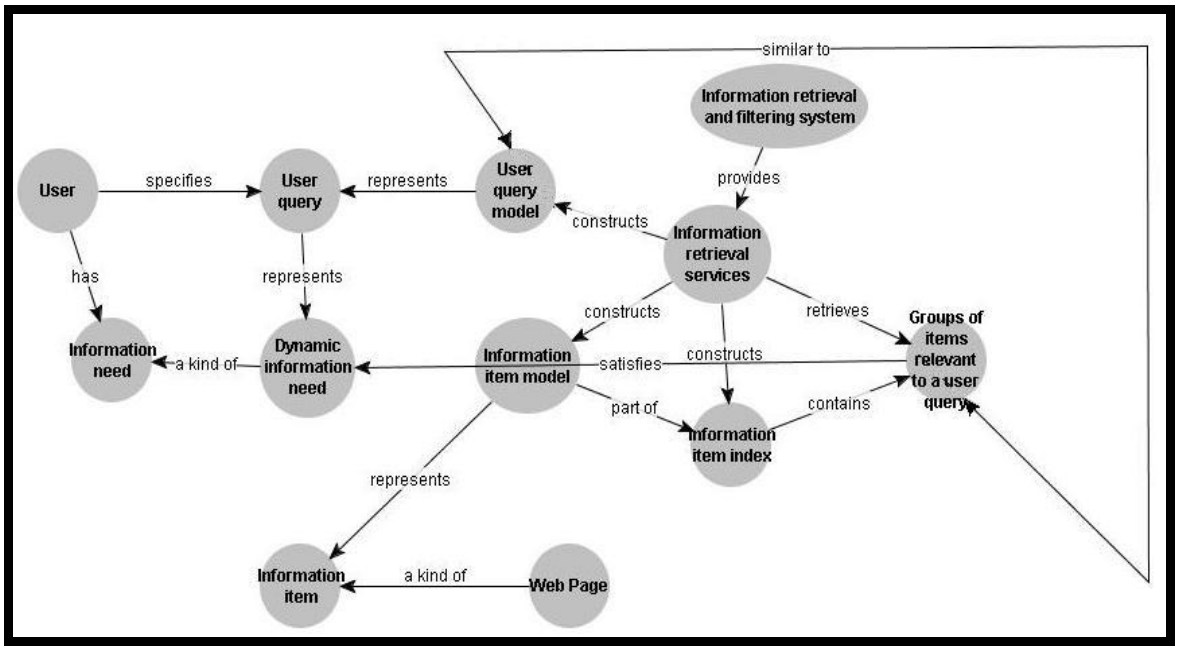


Modelo de Interações entre Agentes da ONTOWUM-DD (GIRARDI, BALBY, 2006)

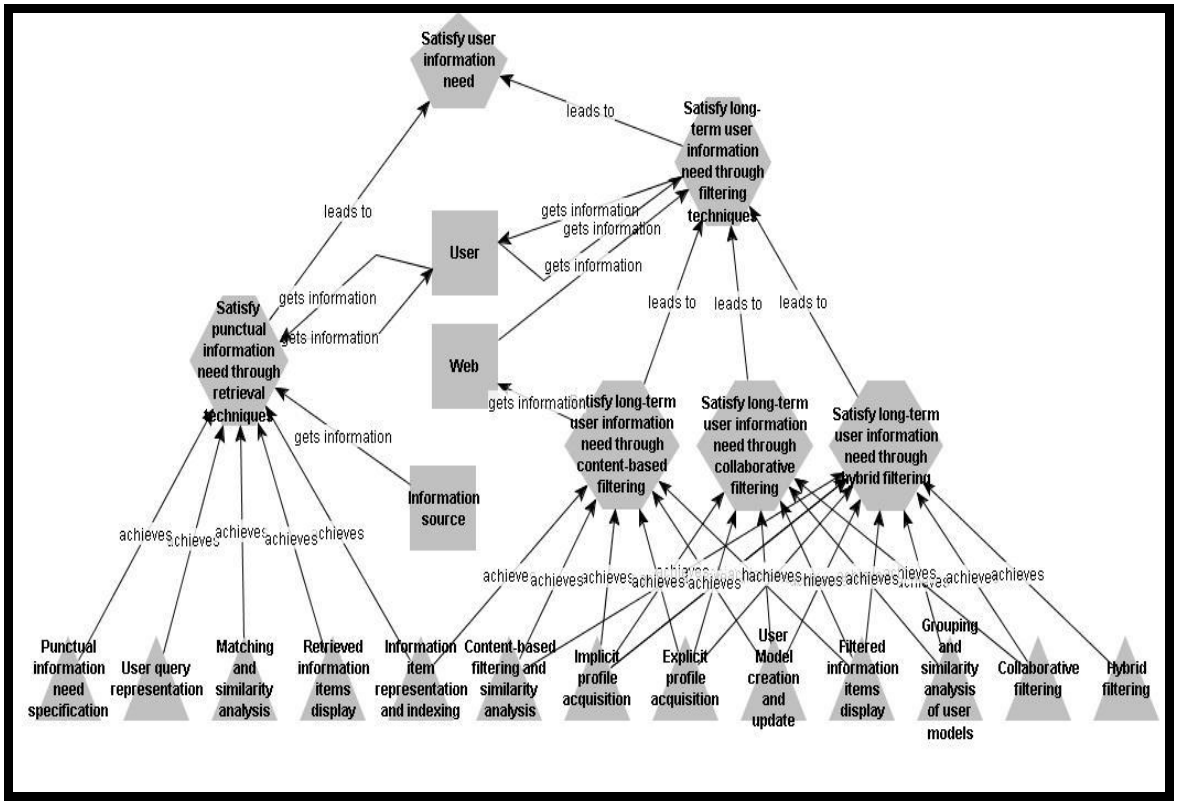


Modelo dos Mecanismos de Cooperação e Coordenação da ONTOWUM-DD (GIRARDI, BALBY, 2006)

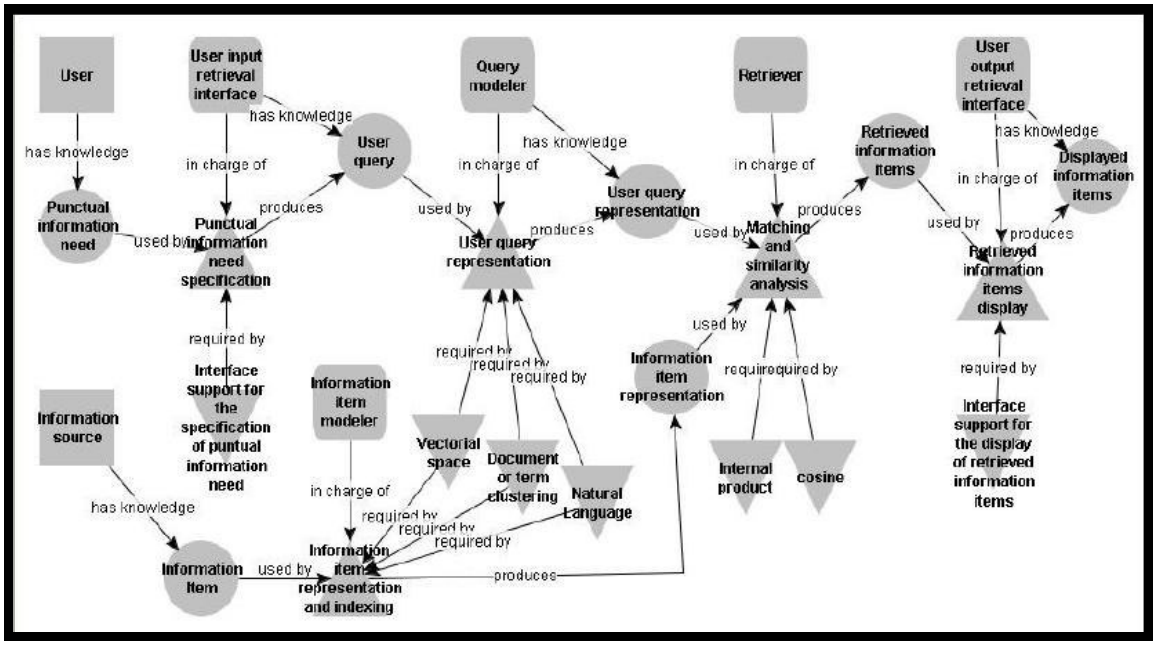
APÊNDICE B – Instâncias da modelagem da ONTOINFO: Modelo de Domínio (DM) e Framework Multiagente (DD)



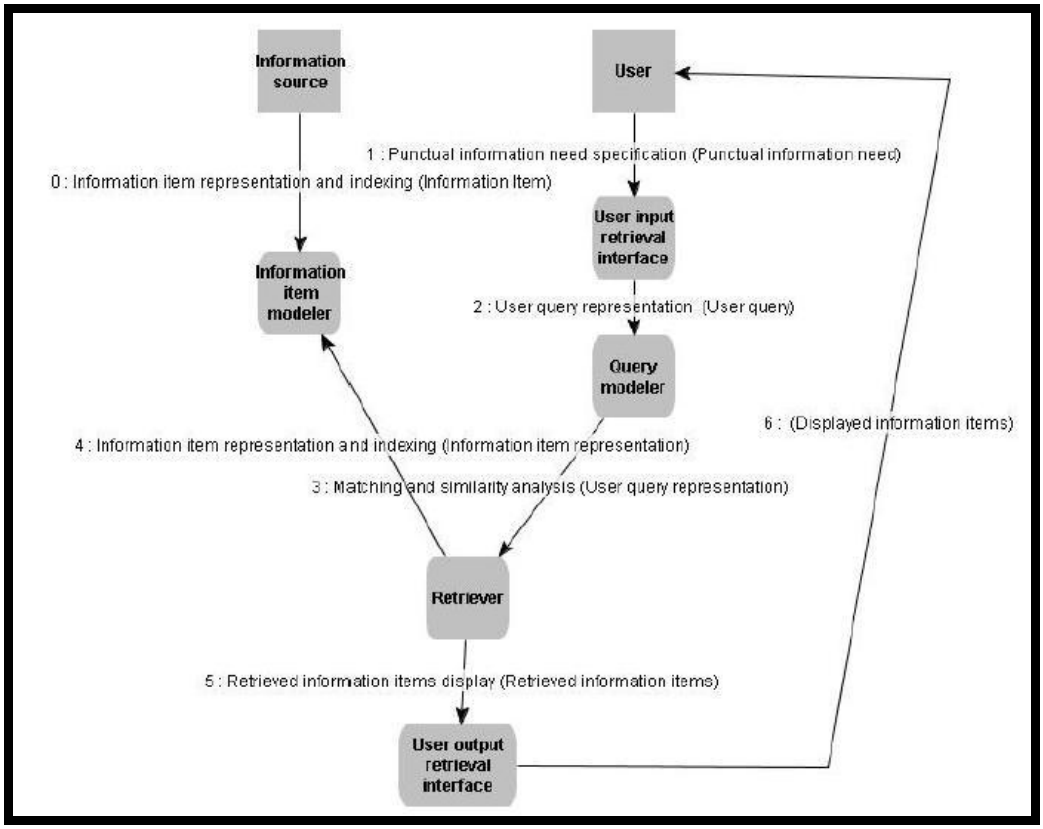
Modelo de Conceitos da ONTOINFO-DM (JANSEN PEREIRA, 2006)



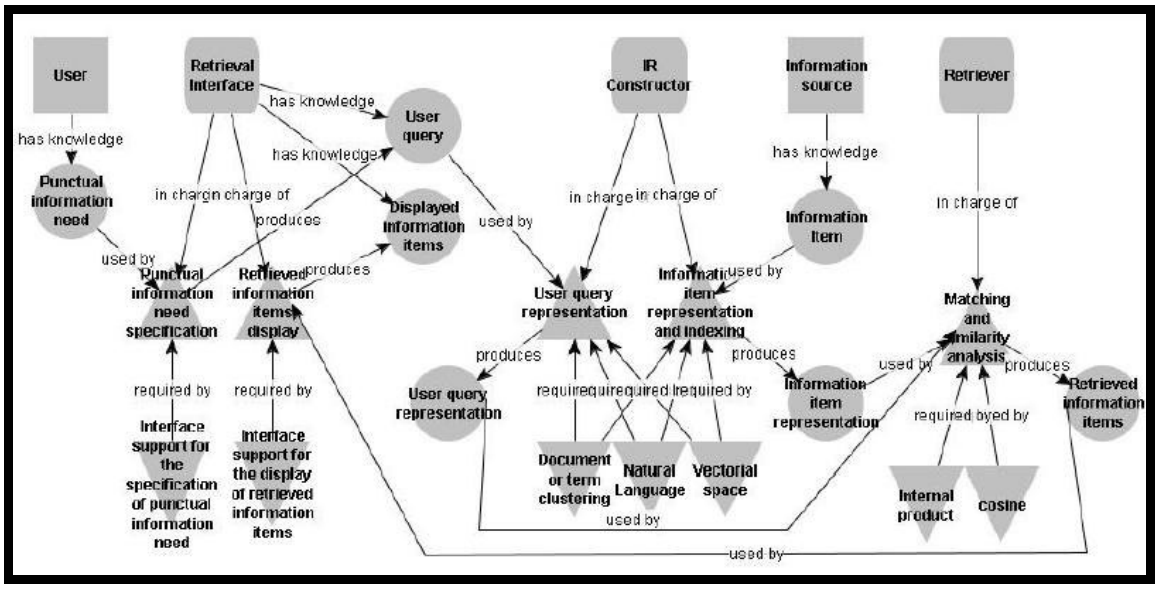
Modelo de Objetivos da ONTOINFO-DM (JANSEN PEREIRA, 2006)



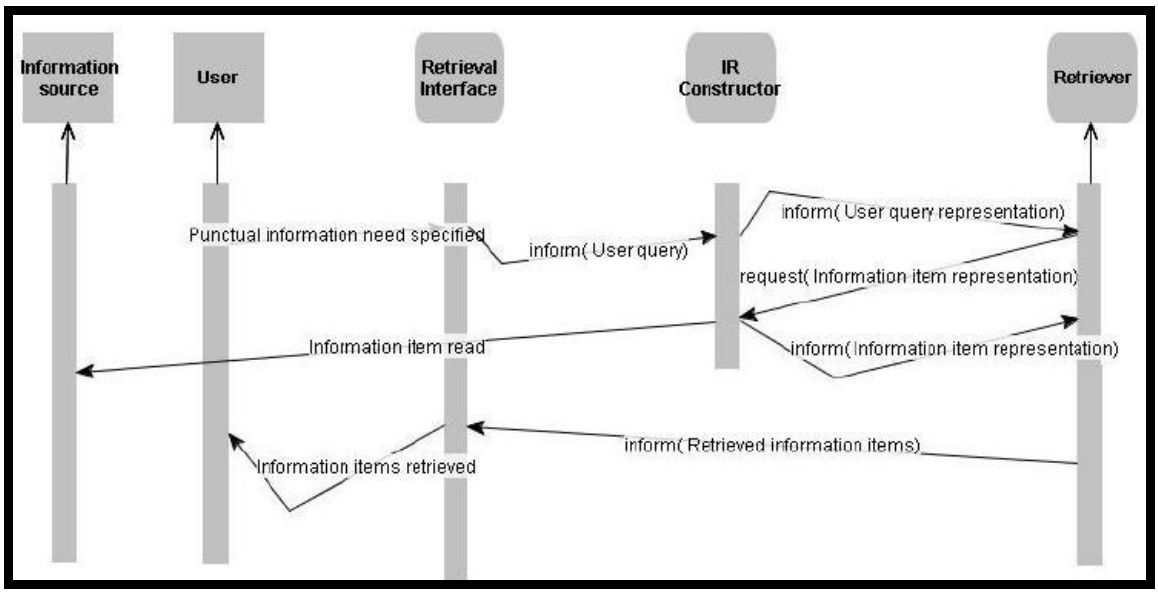
Modelo de Papéis da ONTOINFO-DM relativo à recuperação de informação (JANSEN PEREIRA, 2006)



Modelo de Interações entre Papéis da ONTOINFO-DM relativo ao objetivo específico *satisfazer necessidades pontuais de informação* (JANSEN PEREIRA, 2006)



Modelo da Sociedade Multiagente da ONTOINFO-DD relativo à recuperação de informação (JANSEN PEREIRA, 2006)



Modelo de Interações entre Agentes da ONTOINFO-DD relativo à recuperação de informações (JANSEN PEREIRA, 2006)

APÊNDICE C – Código Fonte da PROPOST

```
package gui;

import jade.wrapper.StaleProxyException;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.StringTokenizer;

import javax.swing.JFrame;

import javax.swing.BorderFactory;
import javax.swing.JTabbedPane;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.border.Border;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JTextArea;
import javax.swing.JComboBox;

public class PROPOST extends JFrame {

	private javax.swing.JPanel jContentPane = null;
	private javax.swing.JMenuBar jJMenuBar = null;
	private javax.swing.JMenu fileMenu = null;
	private javax.swing.JMenu editMenu = null;
	private javax.swing.JMenu helpMenu = null;
	private javax.swing.JMenuItem exitMenuItem = null;
	private javax.swing.JMenuItem aboutMenuItem = null;
	private javax.swing.JMenuItem cutMenuItem = null;
	private javax.swing.JMenuItem copyMenuItem = null;
	private javax.swing.JMenuItem pasteMenuItem = null;
	private javax.swing.JMenuItem saveMenuItem = null;
	private JTabbedPane selecaoTabbedPane = null;
	private JPanel jPanel = null;
	private JButton projetoSelecaoButton = null;
	private JButton projetoPriorizacaoButton = null;
	private JButton projetoAvaliacaoButton = null;
	private JPanel detalheSolucaoPanel = null;
	private JPanel achaSolucaoPanel = null;
	private JPanel recomendacaoPanel = null;
	private JPanel recursosPanel = null;
	private JPanel detalheBotoesPanel = null;
	private JButton detalheAbrirButton = null;
	private JButton detalheNovoButton = null;
	private JButton detalheApagarButton = null;
	private JButton detalheSalvarButton = null;
	private JPanel detalheDescricaoPanel = null;
	private JLabel solutionNameLabel = null;
	private JTextField solutionNameTextField = null;
	private JLabel descricaoLabel = null;
	private JTextArea jTextArea = null;
	private JPanel recomendaPanel = null;
	private JPanel areaPanel = null;
	private JLabel jLabel1 = null;
	private JComboBox areaComboBox = null;
	private JLabel similaridadeLabel = null;
	private JComboBox similaridadeComboBox = null;
	private JButton recomendacaoButton = null;
	/**
	 * This is the default constructor
	 */
	public PROPOST() {
		super();
		initialize();
```

```
}
/**
 * This method initializes this
 *
 * @return void
 */
private void initialize() {
		this.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
		this.setJMenuBar(getJJMenuBar());
		this.setSize(741, 445);
		this.setContentPane(getJContentPane());
		this.setTitle("PROPOST - Project Portfolio Support Tool");

		JadeContainer.startContainer();
}
/**
 * This method initializes jContentPane
 *
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJContentPane() {
		if(jContentPane == null) {
			jContentPane = new javax.swing.JPanel();
			jContentPane.setLayout(new java.awt.BorderLayout());
			jContentPane.add(getSelecaoTabbedPane(), java.awt.BorderLayout.CENTER);
			jContentPane.add(getJPanel(), java.awt.BorderLayout.NORTH);
		}
		return jContentPane;
}
/**
 * This method initializes jMenuBar
 *
 * @return javax.swing.JMenuBar
 */
private javax.swing.JMenuBar getJJMenuBar() {
		if (jJMenuBar == null) {
			jJMenuBar = new javax.swing.JMenuBar();
			jJMenuBar.add(getFileMenu());
			jJMenuBar.add(getEditMenu());
			jJMenuBar.add(getHelpMenu());
		}
		return jJMenuBar;
}
/**
 * This method initializes jMenu
 *
 * @return javax.swing.JMenu
 */
private javax.swing.JMenu getFileMenu() {
		if (fileMenu == null) {
			fileMenu = new javax.swing.JMenu();
			fileMenu.setText("File");
			fileMenu.add(getSaveMenuItem());
			fileMenu.add(getExitMenuItem());
		}
		return fileMenu;
}
/**
 * This method initializes jMenu
 *
 * @return javax.swing.JMenu
 */
private javax.swing.JMenu getEditMenu() {
		if (editMenu == null) {
			editMenu = new javax.swing.JMenu();
			editMenu.setText("Edit");
			editMenu.add(getCutMenuItem());
			editMenu.add(getCopyMenuItem());
			editMenu.add(getPasteMenuItem());
		}
		return editMenu;
}
/**
 * This method initializes jMenu
 *
 * @return javax.swing.JMenu
```

```
	 */
	private javax.swing.JMenu getHelpMenu() {
		if (helpMenu == null) {
			helpMenu = new javax.swing.JMenu();
			helpMenu.setText("Help");
			helpMenu.add(getAboutMenuItem());
		}
		return helpMenu;
	}
	/**
	 * This method initializes jMenuItem
	 *
	 * @return javax.swing.JMenuItem
	 */
	private javax.swing.JMenuItem getExitMenuItem() {
		if (exitMenuItem == null) {
			exitMenuItem = new javax.swing.JMenuItem();
			exitMenuItem.setText("Exit");
			exitMenuItem.addActionListener(new java.awt.event.ActionListener() {
				public void actionPerformed(java.awt.event.ActionEvent e) {
					try {
						JadeContainer.shutDown();
					} catch (StaleProxyException e1) {
						// TODO Auto-generated catch block
						e1.printStackTrace();
					}
					System.exit(0);
				}
			});
		}
		return exitMenuItem;
	}
	/**
	 * This method initializes jMenuItem
	 *
	 * @return javax.swing.JMenuItem
	 */
	private javax.swing.JMenuItem getAboutMenuItem() {
		if (aboutMenuItem == null) {
			aboutMenuItem = new javax.swing.JMenuItem();
			aboutMenuItem.setText("About");
			aboutMenuItem.addActionListener(new java.awt.event.ActionListener() {
				public void actionPerformed(java.awt.event.ActionEvent e) {
					new javax.swing.JDialog(PROPOST.this, "About", true).show();
				}
			});
		}
		return aboutMenuItem;
	}
	/**
	 * This method initializes jMenuItem
	 *
	 * @return javax.swing.JMenuItem
	 */
	private javax.swing.JMenuItem getCutMenuItem() {
		if (cutMenuItem == null) {
			cutMenuItem = new javax.swing.JMenuItem();
			cutMenuItem.setText("Cut");

	cutMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_X,
java.awt.Event.CTRL_MASK, true));
		}
		return cutMenuItem;
	}
	/**
	 * This method initializes jMenuItem
	 *
	 * @return javax.swing.JMenuItem
	 */
	private javax.swing.JMenuItem getCopyMenuItem() {
		if (copyMenuItem == null) {
			copyMenuItem = new javax.swing.JMenuItem();
			copyMenuItem.setText("Copy");

	copyMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_C,
java.awt.Event.CTRL_MASK, true));
```

```
                }
                return copyMenuItem;
        }
        /**
         * This method initializes jMenuItem
         *
         * @return javax.swing.JMenuItem
         */
        private javax.swing.JMenuItem getPasteMenuItem() {
                if (pasteMenuItem == null) {
                        pasteMenuItem = new javax.swing.JMenuItem();
                        pasteMenuItem.setText("Paste");

        pasteMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_V,
java.awt.Event.CTRL_MASK, true));
                }
                return pasteMenuItem;
        }
        /**
         * This method initializes jMenuItem
         *
         * @return javax.swing.JMenuItem
         */
        private javax.swing.JMenuItem getSaveMenuItem() {
                if (saveMenuItem == null) {
                        saveMenuItem = new javax.swing.JMenuItem();
                        saveMenuItem.setText("Save");

        saveMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_S,
java.awt.Event.CTRL_MASK, true));
                }
                return saveMenuItem;
        }
        /**
         * This method initializes jTabbedPane
         *
         * @return javax.swing.JTabbedPane
         */
        private JTabbedPane getSelecaoTabbedPane() {
                if (selecaoTabbedPane == null) {
                        selecaoTabbedPane = new JTabbedPane();
                        selecaoTabbedPane.addTab("Solution Detail", null, getDetalheSolucaoPanel(), null);
                        selecaoTabbedPane.addTab("Find Solution", null, getAchaSolucaoPanel(), null);
                        selecaoTabbedPane.addTab("Ask for Recommendation", null, getRecomendacaoPanel(),
null);
                        selecaoTabbedPane.addTab("Resources", null, getRecursosPanel(), null);


                }
                return selecaoTabbedPane;
        }
        /**
         * This method initializes jPanel
         *
         * @return javax.swing.JPanel
         */
        private JPanel getJPanel() {
                if (jPanel == null) {
                        jPanel = new JPanel();
                        jPanel.add(getProjetoSelecaoButton(), null);
                        jPanel.add(getProjetoPriorizacaoButton(), null);
                        jPanel.add(getProjetoAvaliacaoButton(), null);
                }
                return jPanel;
        }
        /**
         * This method initializes jButton
         *
         * @return javax.swing.JButton
         */
        private JButton getProjetoSelecaoButton() {
                if (projetoSelecaoButton == null) {
                        projetoSelecaoButton = new JButton();
                        projetoSelecaoButton.setSize(300, 300);
                        projetoSelecaoButton.setText("Project Selection");
                }
```

```
		return projetoSelecaoButton;
	}
	/**
	 * This method initializes jButton
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getProjetoPriorizacaoButton() {
		if (projetoPriorizacaoButton == null) {
			projetoPriorizacaoButton = new JButton();
			projetoPriorizacaoButton.setText("Project Priorization");
		}
		return projetoPriorizacaoButton;
	}
	/**
	 * This method initializes jButton1
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getProjetoAvaliacaoButton() {
		if (projetoAvaliacaoButton == null) {
			projetoAvaliacaoButton = new JButton();
			projetoAvaliacaoButton.setText("Project Avaliation");
		}
		return projetoAvaliacaoButton;
	}
	/**
	 * This method initializes jPanel1
	 *
	 * @return javax.swing.JPanel
	 */
	private JPanel getDetalheSolucaoPanel() {
		if (detalheSolucaoPanel == null) {
			detalheSolucaoPanel = new JPanel();
			detalheSolucaoPanel.setLayout(null);
			detalheSolucaoPanel.add(getDetalheBotoesPanel(), null);
			detalheSolucaoPanel.add(getDetalheDescricaoPanel(), null);

		}
		return detalheSolucaoPanel;
	}
	/**
	 * This method initializes jPanel1
	 *
	 * @return javax.swing.JPanel
	 */
	private JPanel getAchaSolucaoPanel() {
		if (achaSolucaoPanel == null) {
			achaSolucaoPanel = new JPanel();
		}
		return achaSolucaoPanel;
	}
	/**
	 * This method initializes jPanel1
	 *
	 * @return javax.swing.JPanel
	 */
	private JPanel getRecomendacaoPanel() {
		if (recomendacaoPanel == null) {
			recomendacaoPanel = new JPanel();
			recomendacaoPanel.setLayout(new BorderLayout());
			recomendacaoPanel.add(getRecomendaPanel(), BorderLayout.CENTER);
		}
		return recomendacaoPanel;
	}
	/**
	 * This method initializes jPanel1
	 *
	 * @return javax.swing.JPanel
	 */
	private JPanel getRecursosPanel() {
		if (recursosPanel == null) {
			recursosPanel = new JPanel();
		}
		return recursosPanel;
	}
```

```
/**
 * This method initializes jPanel1
 *
 * @return javax.swing.JPanel
 */
private JPanel getDetalheBotoesPanel() {
	if (detalheBotoesPanel == null) {
		detalheBotoesPanel = new JPanel();
		detalheBotoesPanel.setLayout(null);
		detalheBotoesPanel.setBorder(BorderFactory.createBevelBorder(1));
		detalheBotoesPanel.setBounds(10, 13, 687, 53);
		detalheBotoesPanel.add(getDetalheNovoButton(), null);
		detalheBotoesPanel.add(getDetalheApagarButton(), null);
		detalheBotoesPanel.add(getDetalheSalvarButton(), null);
		detalheBotoesPanel.add(getDetalheAbrirButton(), null);
	}
	return detalheBotoesPanel;
}
/**
 * This method initializes jButton
 *
 * @return javax.swing.JButton
 */
private JButton getDetalheAbrirButton() {
	if (detalheAbrirButton == null) {
		detalheAbrirButton = new JButton();
		detalheAbrirButton.setText("Abrir");
		detalheAbrirButton.setBounds(10, 10, 75, 34);
	}
	return detalheAbrirButton;
}
/**
 * This method initializes jButton
 *
 * @return javax.swing.JButton
 */
private JButton getDetalheNovoButton() {
	if (detalheNovoButton == null) {
		detalheNovoButton = new JButton();
		detalheNovoButton.setBounds(100, 10, 75, 34);
		detalheNovoButton.setText("Novo");
	}
	return detalheNovoButton;
}
/**
 * This method initializes jButton
 *
 * @return javax.swing.JButton
 */
private JButton getDetalheApagarButton() {
	if (detalheApagarButton == null) {
		detalheApagarButton = new JButton();
		detalheApagarButton.setBounds(190, 10, 75, 34);
		detalheApagarButton.setText("Apagar");
	}
	return detalheApagarButton;
}
/**
 * This method initializes jButton
 *
 * @return javax.swing.JButton
 */
private JButton getDetalheSalvarButton() {
	if (detalheSalvarButton == null) {
		detalheSalvarButton = new JButton();
		detalheSalvarButton.setBounds(275, 10, 74, 34);
		detalheSalvarButton.setText("Salvar");
	}
	return detalheSalvarButton;
}
/**
 * This method initializes jPanel1
 *
 * @return javax.swing.JPanel
 */
private JPanel getDetalheDescricaoPanel() {
```

```
		if (detalheDescricaoPanel == null) {
			solutionNameLabel = new JLabel();
			descricaoLabel = new JLabel();
			detalheDescricaoPanel = new JPanel();
			detalheDescricaoPanel.setBounds(13, 81, 679, 305);
			detalheDescricaoPanel.setLayout(null);
			solutionNameLabel.setBounds(16, 8, 85, 16);
			descricaoLabel.setBounds(14, 53, 72, 19);
			descricaoLabel.setText("Description:");
			detalheDescricaoPanel.add(descricaoLabel, null);
			detalheDescricaoPanel.add(getJTextArea(), null);
			solutionNameLabel.setText("Solution Name:");
			detalheDescricaoPanel.add(solutionNameLabel, null);
			detalheDescricaoPanel.add(getSolutionNameTextField(), null);
		}
		return detalheDescricaoPanel;
}
/**
 * This method initializes jTextField
 *
 * @return javax.swing.JTextField
 */
private JTextField getSolutionNameTextField() {
		if (solutionNameTextField == null) {
			solutionNameTextField = new JTextField();
			solutionNameTextField.setBounds(108, 5, 549, 21);
		}
		return solutionNameTextField;
}
/**
 * This method initializes jTextArea
 *
 * @return javax.swing.JTextArea
 */
private JTextArea getJTextArea() {
		if (jTextArea == null) {
			jTextArea = new JTextArea();
			jTextArea.setBounds(104, 54, 553, 104);
		}
		return jTextArea;
}
/**

 * This method initializes jPanel2
 *
 * @return javax.swing.JPanel
 */
private JPanel getRecomendaPanel() {
		if (recomendaPanel == null) {
			recomendaPanel = new JPanel();
			recomendaPanel.setBorder(BorderFactory.createTitledBorder("Key Word Definition"));
			recomendaPanel.setLayout(new BorderLayout());
			recomendaPanel.add(getAreaPanel(), BorderLayout.CENTER);
		}
		return recomendaPanel;
}
/**
 * This method initializes jPanel2
 *
 * @return javax.swing.JPanel

 */
private JPanel getAreaPanel() {
		if (areaPanel == null) {
			similaridadeLabel = new JLabel();
			jLabel1 = new JLabel();
			areaPanel = new JPanel();
			areaPanel.setLayout(null);
			jLabel1.setText("Related Area:");
			jLabel1.setBounds(16, 13, 76, 16);
			similaridadeLabel.setText("SimilarityLevel:");
			similaridadeLabel.setBounds(16, 55, 86, 16);
			areaPanel.setToolTipText("");
			areaPanel.setLocation(82, 74);
			areaPanel.setSize(526, 220);
			areaPanel.add(jLabel1);
```

```
                areaPanel.add(getAreaComboBox());
                areaPanel.add(similaridadeLabel, null);
                areaPanel.add(getSimilaridadeComboBox(), null);
                areaPanel.add(getRecomendacaoButton(), null);
            }
            return areaPanel;
        }
        /**
         * This method initializes jComboBox
         *
         * @return javax.swing.JComboBox
         */
        private JComboBox getAreaComboBox() {
            if (areaComboBox == null) {
                areaComboBox = new JComboBox();
                areaComboBox.setLocation(130, 12);
                areaComboBox.setSize(576, 25);
                try {
                    DataInputStream in = new DataInputStream(new
FileInputStream("loguso.dat"));
                    StringTokenizer st = new StringTokenizer(in.readLine());
                    while(!st.nextToken().equals("fim")){
                        areaComboBox.addItem("Area " + st.nextToken());
                        st = new StringTokenizer(in.readLine());
                    }
                    in.close();
                } catch (FileNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
            return areaComboBox;
        }
        /**
         * This method initializes jComboBox
         *
         * @return javax.swing.JComboBox
         */
        private JComboBox getSimilaridadeComboBox() {
            if (similaridadeComboBox == null) {
                similaridadeComboBox = new JComboBox();
                similaridadeComboBox.setBounds(130, 50, 115, 25);
                similaridadeComboBox.addItem("High");
                similaridadeComboBox.addItem("Medium");
                similaridadeComboBox.addItem("Low");
            }
            return similaridadeComboBox;
        }
        /**
         * This method initializes jButton
         *
         * @return javax.swing.JButton
         */
        private JButton getRecomendacaoButton() {
            if (recomendacaoButton == null) {
                recomendacaoButton = new JButton();
                recomendacaoButton.setBounds(267, 183, 213, 40);
                recomendacaoButton.setText("Ask for Recommendation");
                recomendacaoButton.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent e) {
                        String s = (String) areaComboBox.getSelectedItem();
                        //JadeContainer.startContainer(s.substring(5));
                        //JadeContainer.geraRecomendacao(s.substring(5));
                        Recomendacao r = new Recomendacao(s.substring(5));
                        r.show();
                    }
                });
            }
            return recomendacaoButton;
        }
            /**
         * Launches this application
         */
```

```
	public static void main(String[] args) {

		PROPOST application = new PROPOST();
		application.show();
	}
}  //  @jve:decl-index=0:visual-constraint="44,2"
```

```
package gui;

import java.util.Vector;

import javax.swing.JFrame;

import javax.swing.BorderFactory;
import javax.swing.JTextPane;
import javax.swing.JTable;
import javax.swing.JScrollPane;
import javax.swing.table.TableColumn;
import javax.swing.JTextArea;

import utilidades.SolucaoUso;
/**
 * @author 01715078
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class Recomendacao extends JFrame {

	private javax.swing.JPanel jContentPane = null;
	private JTable recomendacoesTable = null;
	private JScrollPane jScrollPane = null;
	private JScrollPane mensagensScrollPane = null;  //  @jve:decl-index=0:visual-constraint="438,11"
	private JTextArea mensagensTextArea = null;
	private String area;
	/**
	 * This is the default constructor
	 */
	public Recomendacao(String area) {
		super();
		this.area = area;
		initialize();
	}
	/**
	 * This method initializes this
	 *
	 * @return void
	 */
	private void initialize() {
		this.setSize(525, 306);
		this.setContentPane(getJContentPane());
		this.setTitle("Recomendacoes");


	}
	/**
	 * This method initializes jContentPane
	 *
	 * @return javax.swing.JPanel
	 */
	private javax.swing.JPanel getJContentPane() {
		if(jContentPane == null) {
			jContentPane = new javax.swing.JPanel();
			jContentPane.setLayout(new java.awt.BorderLayout());
			jContentPane.add(getMensagensScrollPane(), java.awt.BorderLayout.NORTH);
			mensagensTextArea.setText("Gerando Recomendações");
			jContentPane.add(getJScrollPane(), java.awt.BorderLayout.CENTER);
			mensagensTextArea.setText("Recomendações Geradas");

		}
		return jContentPane;
	}
```

```
	/**
	 * This method initializes jTable
	 *
	 * @return javax.swing.JTable
	 */
	private JTable getRecomendacoesTable() {
		if (recomendacoesTable == null) {
			String [] colunas = { "Solucao", "Descricao" };
			Vector v = JadeContainer.geraRecomendacao(area);
			Object dados [][] = new Object[v.size()][2];
			for(int i = 0; i < v.size(); i++){
				SolucaoUso s = (SolucaoUso) v.get(i);
				dados[i][0] = new Integer(s.getSolucao());
				dados[i][1] = new Double(s.getValor());
			}

			recomendacoesTable = new JTable(dados, colunas);

		}
		return recomendacoesTable;
	}
	/**
	 * This method initializes jScrollPane
	 *
	 * @return javax.swing.JScrollPane
	 */
	private JScrollPane getJScrollPane() {
		if (jScrollPane == null) {
			jScrollPane = new JScrollPane();
			jScrollPane.setViewportView(getRecomendacoesTable());
			jScrollPane.setBorder(BorderFactory.createTitledBorder("Recomendacoes: "));
		}
		return jScrollPane;
	}
	/**
	 * This method initializes jScrollPane1
	 *
	 * @return javax.swing.JScrollPane
	 */
	private JScrollPane getMensagensScrollPane() {
		if (mensagensScrollPane == null) {
			mensagensScrollPane = new JScrollPane();
			mensagensScrollPane.setViewportView(getMensagensTextArea());
			mensagensScrollPane.setBorder( BorderFactory.createTitledBorder("Mensagens: " ));
		}
		return mensagensScrollPane;
	}

	/**
	 * This method initializes jTextArea
	 *
	 * @return javax.swing.JTextArea
	 */
	private JTextArea getMensagensTextArea() {
		if (mensagensTextArea == null) {
			mensagensTextArea = new JTextArea();
		}
		return mensagensTextArea;
	}
} //  @jve:decl-index=0:visual-constraint="162,27"
```

```
package Minerador;

import jade.util.leap.ArrayList;
import jade.util.leap.List;

import java.util.Iterator;
import java.util.Vector;

import ontology.MatrizCaracteristicas;
import ontology.ModeloUsuarios;
import ontology.GrupoUsuarios;
import CalculoDistancias.MedidasSimilaridade;
```

```
/**
 * Implements the k-means algorithm
*/
public class KMeans {

	/** Number of clusters */
	private int k;


	/** Array of clusters */
	private List grupos;

	private List modelos;


	/** Number of iterations */
	private int nIterations;


	/** Vector of data points */
	private Vector dataPoints;


	/** Name of the input file */
	private String inputFileName;

	public KMeans(int k, List modelos) {

		this.k = k;
		this.grupos = new ArrayList();
		this.nIterations = 0;
		this.modelos=modelos;

	} // end of kMeans()


	public void runKMeans() {

		// Select k points as initial means
		for (int i=0; i < k; i++){

			int numModelo = (int)(Math.random() * this.modelos.size());
			//int numModelo = i;

			GrupoUsuarios grupo = new GrupoUsuarios(i);

	grupo.setMatrizVisitas(((ModeloUsuarios)this.modelos.get(numModelo)).getMatrizVisitas());

	grupo.setMatrizSequencias(((ModeloUsuarios)this.modelos.get(numModelo)).getMatrizSequencia());

	grupo.setMatrizTempos(((ModeloUsuarios)this.modelos.get(numModelo)).getMatrizTempo());

			this.grupos.add(i,grupo);
			//System.out.println("Random: "+numModelo);
		}
		do {
			// Form k clusters
			Iterator i = this.modelos.iterator();
			while (i.hasNext())
				this.assignToCluster((ModeloUsuarios)(i.next()));
			this.nIterations++;

		}
		// Repeat while centroids do not change
		while (atualizaGrupos());

	} // end of runKMeans()


	private void assignToCluster(ModeloUsuarios modelo) {

		int grupoCandidato=0;
```

```
                MedidasSimilaridade medidasSimilaridade = new MedidasSimilaridade();
                double distanciaVisitas =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizVisitas(),((GrupoUsuarios)this.grupos.get(grupoCandid
ato)).getMatrizVisitas());
                double distanciaSequencias =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizSequencia(),((GrupoUsuarios)this.grupos.get(grupoCa
ndidato)).getMatrizSequencias());
                double distanciaTempos =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizTempo(),((GrupoUsuarios)this.grupos.get(grupoCandi
dato)).getMatrizTempos());

                double minDistance =
medidasSimilaridade.calcularMediaPonderadaDissimilaridade(distanciaVisitas,distanciaSequencias,distanciaTempos,0.
5,0.3,0.2);

                for (int i=1; i <this.k; i++) {
                        double distVisitas =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizVisitas(),((GrupoUsuarios)this.grupos.get(i)).getMatriz
Visitas());
                        double distSequencias =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizSequencia(),((GrupoUsuarios)this.grupos.get(i)).getMa
trizSequencias());
                        double distTempos =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizTempo(),((GrupoUsuarios)this.grupos.get(i)).getMatriz
Tempos());
                        double mDistance =
medidasSimilaridade.calcularMediaPonderadaDissimilaridade(distVisitas,distSequencias,distTempos,0.5,0.3,0.2);
                        if (mDistance < minDistance) {
                                minDistance = mDistance;
                                grupoCandidato = i;

                        }
                }
                modelo.setGrupoNum(grupoCandidato);
                ((GrupoUsuarios)this.grupos.get(grupoCandidato)).incCardinalidade();
                } // end of assignToCluster

        public GrupoUsuarios classificaUsuarioCorrente(ModeloUsuarios modelo) {

                int grupoCandidato=0;
                MedidasSimilaridade medidasSimilaridade = new MedidasSimilaridade();
                double distanciaVisitas =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizVisitas(),((GrupoUsuarios)this.grupos.get(grupoCandid
ato)).getMatrizVisitas());
                double distanciaSequencias =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizSequencia(),((GrupoUsuarios)this.grupos.get(grupoCa
ndidato)).getMatrizSequencias());
                double distanciaTempos =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizTempo(),((GrupoUsuarios)this.grupos.get(grupoCandi
dato)).getMatrizTempos());
                double minDistance =
medidasSimilaridade.calcularMediaPonderadaDissimilaridade(distanciaVisitas,distanciaSequencias,distanciaTempos,0.
5,0.3,0.2);

                for (int i=1; i <this.k; i++) {

                        double distVisitas =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizVisitas(),((GrupoUsuarios)this.grupos.get(i)).getMatriz
Visitas());
                        double distSequencias =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizSequencia(),((GrupoUsuarios)this.grupos.get(i)).getMa
trizSequencias());
                        double distTempos =
medidasSimilaridade.medirDistanciaEuclidiana(modelo.getMatrizTempo(),((GrupoUsuarios)this.grupos.get(i)).getMatriz
Tempos());
                        double mDistance =
medidasSimilaridade.calcularMediaPonderadaDissimilaridade(distVisitas,distSequencias,distTempos,0.5,0.3,0.2);
                        if (mDistance < minDistance) {
                                minDistance = mDistance;
                                grupoCandidato = i;
                        }
                }
                ((GrupoUsuarios)this.grupos.get(grupoCandidato)).incCardinalidade();
```

```
		atualizarCentroide(modelo,(GrupoUsuarios)this.grupos.get(grupoCandidato),((GrupoUsuarios)this.grupos.get(
grupoCandidato)).getCardinalidade());
			return (GrupoUsuarios)this.grupos.get(grupoCandidato);

		}


		/**
		 * Updates the means of all k clusters, and returns if they have changed or not
		 *
		 * @return		have the updated means of the clusters changed or not
		 */
		private void calculaCentroide() {
			int[] size = new int[this.k];
			Iterator i = this.modelos.iterator();
			while (i.hasNext()) {

				ModeloUsuarios mu = (ModeloUsuarios)(i.next());
				int grupoAtual = mu.getGrupoNum();
				//System.out.println("Pertenco ao grupo:"+grupoAtual);

				((GrupoUsuarios) this.grupos.get(grupoAtual))
				.setMatrizVisitas(calcularSomaMatrizes(mu.getMatrizVisitas(),((GrupoUsuarios)
this.grupos.get(grupoAtual)).getMatrizVisitas()));

				((GrupoUsuarios) this.grupos.get(grupoAtual))
				.setMatrizSequencias(calcularSomaMatrizes(mu.getMatrizSequencia(),((GrupoUsuarios)
this.grupos.get(grupoAtual)).getMatrizSequencias()));

				((GrupoUsuarios) this.grupos.get(grupoAtual))
				.setMatrizTempos(calcularSomaMatrizes(mu.getMatrizTempo(),((GrupoUsuarios)
this.grupos.get(grupoAtual)).getMatrizTempos()));

				size[grupoAtual]++;
			}
			for (int j=0; j < this.k; j++ )
				if(size[j] != 0) {

					((GrupoUsuarios) this.grupos.get(j))
					.setMatrizVisitas(calcularMediaMatriz(((GrupoUsuarios)
this.grupos.get(j)).getMatrizVisitas(), size[j]));

					((GrupoUsuarios) this.grupos.get(j))
					.setMatrizSequencias(calcularMediaMatriz(((GrupoUsuarios)
this.grupos.get(j)).getMatrizSequencias(), size[j]));

					((GrupoUsuarios) this.grupos.get(j))
					.setMatrizTempos(calcularMediaMatriz(((GrupoUsuarios)
this.grupos.get(j)).getMatrizTempos(), size[j]));
				}
		}
		private void atualizarCentroide(ModeloUsuarios m, GrupoUsuarios g, int cardinalidade) {


		g.setMatrizVisitas(atualizarMediaMatrizes(g.getMatrizVisitas(),m.getMatrizVisitas(),cardinalidade));


		g.setMatrizSequencias(atualizarMediaMatrizes(g.getMatrizSequencias(),m.getMatrizSequencia(),cardinalidad
e));


		g.setMatrizTempos(atualizarMediaMatrizes(g.getMatrizTempos(),m.getMatrizTempo(),cardinalidade));


		}


		private boolean atualizaGrupos() {

			boolean reply = false;

			List mediaVisitas = new ArrayList();
			List mediaSequencia = new ArrayList();
			List mediaTempos = new ArrayList();
			List gruposUsuarios = new ArrayList();
```

```
                for (int i=0; i<this.k; i++) {

                        gruposUsuarios.add(i,(GrupoUsuarios)this.grupos.get(i));

                }
                calculaCentroide();
                for (int j=0; j < this.k; j++ ) {

                                MedidasSimilaridade medidasSimilaridade = new MedidasSimilaridade();

                                double distanciaVisitas =
medidasSimilaridade.medirDistanciaEuclidiana(((GrupoUsuarios)
gruposUsuarios.get(j)).getMatrizVisitas(),((GrupoUsuarios) this.grupos.get(j)).getMatrizVisitas());
                                double distanciaSequencia =
medidasSimilaridade.medirDistanciaEuclidiana(((GrupoUsuarios)
gruposUsuarios.get(j)).getMatrizSequencias(),((GrupoUsuarios) this.grupos.get(j)).getMatrizSequencias());
                                double distanciaTempo =
medidasSimilaridade.medirDistanciaEuclidiana(((GrupoUsuarios)
gruposUsuarios.get(j)).getMatrizTempos(),((GrupoUsuarios) this.grupos.get(j)).getMatrizTempos());

                                double dist =
medidasSimilaridade.calcularMediaPonderadaDissimilaridade(distanciaVisitas,distanciaSequencia,distanciaTempo,0.5,
0.3,0.2);
                                //System.out.println("Dist: "+dist);
                                if (dist !=0 )
                                        reply = true;

                        }

                return reply;

        }


        /**
         * Returns the value of k
         *
         * @return              the value of k
         */
  public void imprimeGrupos () {
        for (int a=0;a<this.grupos.size();a++) {
                GrupoUsuarios g=(GrupoUsuarios)this.grupos.get(a);
                        MatrizCaracteristicas visita = g.getMatrizVisitas();
                        MatrizCaracteristicas sequencia = g.getMatrizSequencias();
                        MatrizCaracteristicas tempo = g.getMatrizTempos();

        }
  }
        public int getK() {

                return this.k;

        } // end of getK()

        public MatrizCaracteristicas calcularSomaMatrizes(MatrizCaracteristicas m1,MatrizCaracteristicas m2) {

                MatrizCaracteristicas matrizMedia;

                int linhas  = m1.getLinhas(),
                        colunas = m2.getColunas();

                matrizMedia = new MatrizCaracteristicas();
                matrizMedia.contruirMatriz(linhas,colunas);

                for(int i = 0; i < linhas; i++)
                        for(int j = 0; j < colunas; j++)
                                matrizMedia.ajustarValor((m1.getValor(i,j)+m2.getValor(i,j)),i,j);

                return matrizMedia;

        }
        public MatrizCaracteristicas calcularMediaMatriz(MatrizCaracteristicas m1, int size) {
                MatrizCaracteristicas matrizMedia;

                int linhas  = m1.getLinhas(),
```

```
                        colunas = m1.getColunas();

                matrizMedia = new MatrizCaracteristicas();
                matrizMedia.contruirMatriz(linhas,colunas);

                for(int i = 0; i < linhas; i++)
                        for(int j = 0; j < colunas; j++) {
                                matrizMedia.ajustarValor((m1.getValor(i,j)/size),i,j);
                        }
                return matrizMedia;

        }
        public MatrizCaracteristicas atualizarMediaMatrizes(MatrizCaracteristicas m1,MatrizCaracteristicas m2, int
cardinalidade) {

                MatrizCaracteristicas matrizMedia;

                int linhas  = m1.getLinhas(),
                        colunas = m2.getColunas();

                matrizMedia = new MatrizCaracteristicas();
                matrizMedia.contruirMatriz(linhas,colunas);

                for(int i = 0; i < linhas; i++)
                        for(int j = 0; j < colunas; j++)

        matrizMedia.ajustarValor((cardinalidade*(m1.getValor(i,j))+m2.getValor(i,j))/cardinalidade+1,i,j);

                return matrizMedia;

        }


} // end of class
```

```
package Minerador;

import jade.content.ContentElement;
import jade.content.lang.Codec;
import jade.content.lang.Codec.CodecException;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.Ontology;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.proto.AchieveREResponder;
import jade.proto.FIPAProtocolNames;
import jade.util.leap.List;
import ontology.GrupoUsuarios;
import ontology.MatrizCaracteristicas;
import ontology.ModeloUsuarios;
import ontology.OntowumOntology;

public class ClassificaUsuarioCorrente extends AchieveREResponder {

        private Minerador agent;
        private Codec codec = new SLCodec();
        private Ontology ont = OntowumOntology.getInstance();

        public ClassificaUsuarioCorrente(Minerador a, MessageTemplate mt) {

                super(a, mt);
                this.agent=a;

        }
        protected ACLMessage prepareResponse(ACLMessage request) {

                agent.getContentManager().registerLanguage(codec);
                agent.getContentManager().registerOntology(ont);
                ACLMessage reply = request.createReply();
                reply.setPerformative(ACLMessage.INFORM);

                try {
```

```
                                ContentElement ce = agent.getContentManager().extractContent(request);
                                if (ce instanceof ModeloUsuarios) {

                                        ModeloUsuarios modelo_corrente = (ModeloUsuarios) ce;
                                        GrupoUsuarios grupo_corrente =
agent.kMeans.classificaUsuarioCorrente(modelo_corrente);

                                        System.out.println("Matriz de Visitas Grupos:\n");
                                        for(int i = 0; i < visita.getLinhas(); i++) {
                                                for(int j = 0; j < visita.getColunas(); j++) {
                                                        System.out.print(new Float(visita.getValor(i,j)));
                                                        if(j < (visita.getColunas() - 1))
                                                                System.out.print("   ");
                                                }
                                                System.out.println("\n");
                                        }
                                        System.out.println();*/

                                        try {
                                                agent.getContentManager().fillContent(reply, grupo_corrente);
                                        }catch (CodecException c) {
                                                c.printStackTrace();
                                        }
                                }
                        }
                        catch (Exception e) {
                                e.printStackTrace();
                        }
                        return reply;
                }
}
```

```
package modelador;

import java.io.IOException;
import java.util.List;
import java.util.Vector;
import java.util.Collections;

import interfaceador.Interfaceador;
import jade.content.ContentElement;
import jade.content.Predicate;
import jade.content.abs.AbsPredicate;
import jade.content.lang.Codec;
import jade.content.lang.Codec.CodecException;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.Ontology;
import jade.content.onto.OntologyException;
import jade.core.AID;
import jade.core.behaviours.CyclicBehaviour;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.proto.FIPAProtocolNames;
import jade.wrapper.AgentContainer;
import ontology.FoiRealizada;
import ontology.GrupoUsuarios;
import ontology.MatrizCaracteristicas;
import ontology.OntowumOntology;
import ontology.Pagina;
import ontology.VisitaUsuario;
import fm.ModeloCaracteristicas;
public class CriaModeloUsuarios extends CyclicBehaviour {

private Modelador agent;
private Codec codec = new SLCodec();
private Ontology ont = OntowumOntology.getInstance();;
   private AgentContainer serverContainer;


        /* (non-Javadoc)
         * @see jade.core.behaviours.Behaviour#action()
         */
        public CriaModeloUsuarios(Modelador a) {
                super(a);
```

```
		this.agent=a;
	}
	public void action() {
		// TODO Auto-generated method stub
		agent.getContentManager().registerLanguage(codec);
		agent.getContentManager().registerOntology(ont);
MessageTemplate mt = MessageTemplate.and(
		MessageTemplate.MatchLanguage(codec.getName()),
		MessageTemplate.MatchOntology(ont.getName()));
//mt.MatchSender(new AID("Interfaceador@nb160731:1099/JADE"));
		ACLMessage msg = agent.receive(mt);


		if (msg!=null) {
		try {
			ContentElement ce = null;
			if (msg.getPerformative() == ACLMessage.INFORM_REF) {
			ce = agent.getContentManager().extractContent(msg);
			if (ce instanceof FoiRealizada) {
				FoiRealizada visitaRealizada = (FoiRealizada) ce;
				VisitaUsuario visita = visitaRealizada.getVisita();
				agent.sessao.addVisitas(visita);
				System.out.println("Numero da visita: "+agent.sessao.getNumeroVisitas());
				System.out.println("End. da visita: "+visita.getPagina().getEndereco());
				agent.sessao.setOrdemSegmento(agent.ordemSegmento);

				if (agent.sessao.getNumeroVisitas()>=agent.ordemSegmento) {

					ModeloCaracteristicas modeloCaracteristicas = new
ModeloCaracteristicas(agent.sessao,agent.matrizSegmentos);
					ontology.MatrizCaracteristicas matrizVisitas =
modeloCaracteristicas.construirMatrizVisitas();
					ontology.MatrizCaracteristicas matrizSequencias =
modeloCaracteristicas.construirMatrizSequencias();
					ontology.MatrizCaracteristicas matrizTempos =
modeloCaracteristicas.construirMatrizTempos();
					agent.modeloUsuarios.setMatrizVisitas(matrizVisitas);
					agent.modeloUsuarios.setMatrizSequencia(matrizSequencias);
					agent.modeloUsuarios.setMatrizTempo(matrizTempos);
					agent.modeloUsuarios.setSessaoUso(agent.sessao);
					ACLMessage request = new ACLMessage(ACLMessage.REQUEST);
					request.setProtocol(FIPAProtocolNames.FIPA_REQUEST);

					int k = 6;
					if (agent.sessao.getNumeroVisitas()==k) {
						int i = agent.sessao.getNumeroVisitas(); // pega numero de
visitas
						String s = agent.sessao.getVisita(i-
1).getPagina().getEndereco(); //pega o endereco da ultima visita
						int ultimaVisita = Integer.parseInt(s);
						matrizVisitas.ajustarValor(
matrizVisitas.getValor(ultimaVisita-1, ultimaVisita-1)+1, ultimaVisita-1, ultimaVisita-1);
						agent.addBehaviour(new
SolicitaClassificacao(agent,request));
						k=k+4;
					}
					}
			}
		}
		} else {
			//System.out.println("mandando mensagem");
			this.agent.addBehaviour( new FinalizaSecaoUsuario(this.agent) );
		}
	}
			catch (CodecException ce) {
			ce.printStackTrace();
			}
			catch (OntologyException oe) {
				oe.printStackTrace();
			}
		}
		else
			block();
	}
```

```
}
```

```
package modelador;

import jade.content.ContentElement;
import jade.content.Predicate;
import jade.content.lang.Codec;
import jade.content.lang.Codec.CodecException;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.Ontology;
import jade.content.onto.OntologyException;
import jade.core.AID;
import jade.lang.acl.ACLMessage;
import jade.proto.SimpleAchieveREInitiator;
import ontology.GrupoUsuarios;
import ontology.MatrizCaracteristicas;
import ontology.ModeloUsuarios;
import ontology.OntowumOntology;

public class SolicitaClassificacao extends SimpleAchieveREInitiator {

	private Modelador agent;
	private Codec codec = new SLCodec();
	private Ontology ont = OntowumOntology.getInstance();

	public SolicitaClassificacao(Modelador a, ACLMessage msg) {
		super(a, msg);
		this.agent=a;
	}
	protected ACLMessage prepareRequest(ACLMessage request) {

		AID saAID = new AID();
		saAID.setName("Minerador@nb160731:1099/JADE");
		//saAID.addAddresses("");
		request.setLanguage(codec.getName());
		request.setOntology(ont.getName());
		request.addReceiver(saAID);
		try {
			agent.getContentManager().fillContent(request, agent.modeloUsuarios);
		}
		catch (CodecException ce) {
			ce.printStackTrace();
		}
		catch (OntologyException oe) {
			oe.printStackTrace();
		}
		return request;
	}


	protected void handleInform(ACLMessage msg) {
		try {

			ContentElement ce = agent.getContentManager().extractContent(msg);
			if(ce instanceof Predicate) {
			Predicate p = (Predicate) ce;
			if (p instanceof GrupoUsuarios) {
				GrupoUsuarios grupo_corrente = (GrupoUsuarios)p;
				MatrizCaracteristicas visita = grupo_corrente.getMatrizVisitas();
				MatrizCaracteristicas sequencia = grupo_corrente.getMatrizSequencias();
				MatrizCaracteristicas tempo = grupo_corrente.getMatrizTempos();

				System.out.println("Este grupo possui " + grupo_corrente.getCardinalidade() + "
membros.");

		catch (Exception e) {
			e.printStackTrace();
		}

	}
}
```

```
package fm;

import ontology.SessaoUso;
import ontology.VisitaUsuario;
import ontology.SegmentoUso;


public class ModeloCaracteristicas {
	private SessaoUso sessaoUso;
	private MatrizSegmentos matrizSegmentos;

	public ModeloCaracteristicas(SessaoUso sessaoUso, MatrizSegmentos matrizSegmentos){
		this.sessaoUso = sessaoUso;
		this.matrizSegmentos = matrizSegmentos;
	}

	public ontology.MatrizCaracteristicas construirMatrizVisitas() {
		ontology.MatrizCaracteristicas matrizVisitas = new ontology.MatrizCaracteristicas();

	matrizVisitas.contruirMatriz(this.matrizSegmentos.obterLinhas(),this.matrizSegmentos.obterColunas());
		for(int i = 0; i < (this.sessaoUso.getNumeroSegmentos()); i++) {
			SegmentoUso segmentoUso = this.sessaoUso.getSegmento(i);
			ontology.Posicao posicao =
this.matrizSegmentos.obterPosicaoSegmento(segmentoUso);
			matrizVisitas.incremetarValor(posicao);
		}
		return matrizVisitas;
	}

	public ontology.MatrizCaracteristicas construirMatrizSequencias() {
		ontology.MatrizCaracteristicas matrizSequencias = new ontology.MatrizCaracteristicas();


	matrizSequencias.contruirMatriz(this.matrizSegmentos.obterLinhas(),this.matrizSegmentos.obterColunas());
		for(int i = 0; i < (this.sessaoUso.getNumeroSegmentos()); i++) {
			SegmentoUso segmentoUso = this.sessaoUso.getSegmento(i);
			ontology.Posicao posicao =
this.matrizSegmentos.obterPosicaoSegmento(segmentoUso);
			/*
			 * Lucas 01
			 * matrizSequencias.incremetarValor(i + 1,posicao);
			 */
			matrizSequencias.ajustarValor(1,posicao);
		}



		ontology.MatrizCaracteristicas matrizVisitas = this.construirMatrizVisitas();
		for(int i = 0; i < (this.matrizSegmentos.obterLinhas()); i++)
			for(int j = 0; j < (this.matrizSegmentos.obterColunas()); j++)
				if(matrizVisitas.getValor(i,j) != 0) {
					float valor = (float) matrizSequencias.getValor(i,j) /
matrizVisitas.getValor(i,j);
					System.out.println(i+","+j);
					/*
					 * Lucas 02
					 * matrizSequencias.ajustarValor(valor,i,j);
					 */
					matrizSequencias.ajustarValor(1,i,j);
				}
		return matrizSequencias;
	}

	public ontology.MatrizCaracteristicas construirMatrizTempos() {
		ontology.MatrizCaracteristicas matrizTempos = new ontology.MatrizCaracteristicas();

	matrizTempos.contruirMatriz(this.matrizSegmentos.obterLinhas(),this.matrizSegmentos.obterColunas());

		for(int i = 0; i < (this.sessaoUso.getNumeroSegmentos()); i++) {
			SegmentoUso segmentoUso = this.sessaoUso.getSegmento(i);
			float tempo = 0;
			for(int j = 0; j < (segmentoUso.getNumeroVisitas()); j++) {
				VisitaUsuario visitaUsuario = segmentoUso.getVisita(j);
//				if((i > 0) &&
!(visitaUsuario.obterPagina().obterEndereco().equalsIgnoreCase(this.sessaoUso.obterSegmento(i - 1).obterVisita(j -
1).obterPagina().obterEndereco())))
```

```
                                                tempo += visitaUsuario.getTempo();
                                        }
                                        ontology.Posicao posicao =
this.matrizSegmentos.obterPosicaoSegmento(segmentoUso);
                                        /*
                                         * Lucas 02
                                         * matrizTempos.incremetarValor(tempo,posicao);
                                         */
                                        matrizTempos.ajustarValor(1,posicao);
                                }
                                return matrizTempos;
                        }
}
```

```
package ontology;

import jade.content.onto.*;
import jade.content.schema.*;
import jade.util.leap.HashMap;
import jade.content.lang.Codec;
import jade.core.CaseInsensitiveString;

public class OntowumOntology extends jade.content.onto.Ontology implements ProtegeTools.ProtegeOntology {
  /**
   * These hashmap store a mapping from jade names to either protege names of SlotHolder
   * containing the protege names.  And vice versa
   */
  private HashMap jadeToProtege;

 //NAME
 public static final String ONTOLOGY_NAME = "ontowum";
 // The singleton instance of this ontology
 private static ProtegeIntrospector introspect = new ProtegeIntrospector();
 private static Ontology theInstance = new OntowumOntology();
 public static Ontology getInstance() {
   return theInstance;
 }

  // ProtegeOntology methods
  public SlotHolder getSlotNameFromJADEName(SlotHolder jadeSlot) {
    return (SlotHolder) jadeToProtege.get(jadeSlot);
  }


  // storing the information
  private void storeSlotName(String jadeName, String javaClassName, String slotName){
      jadeToProtege.put(new SlotHolder(javaClassName, jadeName), new SlotHolder(javaClassName, slotName));
  }


  // VOCABULARY
   public static final String MODELOUSUARIOS_MATRIZTEMPO="matrizTempo";
   public static final String MODELOUSUARIOS_MATRIZVISITAS="matrizVisitas";
   public static final String MODELOUSUARIOS_MATRIZSEQUENCIA="matrizSequencia";
   public static final String MODELOUSUARIOS_SESSAOUSO="sessaoUso";
   public static final String MODELOUSUARIOS="ModeloUsuarios";
   public static final String FOIREALIZADA_VISITA="visita";
   public static final String FOIREALIZADA="FoiRealizada";
   public static final String SESSAOUSO_ORDEMSEGMENTO="ordemSegmento";
   public static final String SESSAOUSO_VISITAS="visitas";
   public static final String SESSAOUSO="SessaoUso";
   public static final String SEGMENTOUSO_VISITAS="visitas";
   public static final String SEGMENTOUSO="segmentoUso";
   public static final String ESPACOPAGINAS_PAGINAS="paginas";
   public static final String ESPACOPAGINAS="EspacoPaginas";
   public static final String PAGINA_ENDERECO="endereco";
   public static final String PAGINA="Pagina";
   public static final String POSICAO_LINHA="linha";
   public static final String POSICAO_COLUNA="coluna";
   public static final String POSICAO="Posicao";
   public static final String MATRIZCARACTERISTICAS_MATRIZES="matrizes";
   public static final String MATRIZCARACTERISTICAS_LINHAS="linhas";
   public static final String MATRIZCARACTERISTICAS_COLUNAS="colunas";
   public static final String MATRIZCARACTERISTICAS="MatrizCaracteristicas";
```

```
  public static final String VISITAUSUARIO_TEMPO="tempo";
  public static final String VISITAUSUARIO_PAGINA="pagina";
  public static final String VISITAUSUARIO="VisitaUsuario";
  public static final String GRUPOUSUARIOS="GrupoUsuarios";
  public static final String GRUPOUSUARIOS_MATRIZVISITAS="matrizVisitas";
  public static final String GRUPOUSUARIOS_MATRIZSEQUENCIA="matrizSequencia";
  public static final String GRUPOUSUARIOS_MATRIZTEMPO="matrizTempo";
  public static final String GRUPOUSUARIOS_GRUPONUM="grupoNum";
  public static final String GRUPOUSUARIOS_CARDINALIDADE="cardinalidade";

 /**
  * Constructor
 */
 private OntowumOntology(){
  super(ONTOLOGY_NAME, BasicOntology.getInstance());
  introspect.setOntology(this);
  jadeToProtege = new HashMap();
  try {

  // adding Concept(s)
  ConceptSchema visitaUsuarioSchema = new ConceptSchema(VISITAUSUARIO);
  add(visitaUsuarioSchema, ontology.VisitaUsuario.class);
  ConceptSchema matrizCaracteristicasSchema = new ConceptSchema(MATRIZCARACTERISTICAS);
  add(matrizCaracteristicasSchema, ontology.MatrizCaracteristicas.class);
  ConceptSchema posicaoSchema = new ConceptSchema(POSICAO);
  add(posicaoSchema, ontology.Posicao.class);
  ConceptSchema paginaSchema = new ConceptSchema(PAGINA);
  add(paginaSchema, ontology.Pagina.class);
  ConceptSchema espacoPaginasSchema = new ConceptSchema(ESPACOPAGINAS);
  add(espacoPaginasSchema, ontology.EspacoPaginas.class);
  ConceptSchema sessaoUsoSchema = new ConceptSchema(SESSAOUSO);
  add(sessaoUsoSchema, ontology.SessaoUso.class);
  ConceptSchema segmentoUsoSchema = new ConceptSchema(SEGMENTOUSO);
  add(sessaoUsoSchema, ontology.SessaoUso.class);

  // adding AgentAction(s)

  // adding AID(s)

  // adding Predicate(s)
  PredicateSchema foiRealizadaSchema = new PredicateSchema(FOIREALIZADA);
  add(foiRealizadaSchema, ontology.FoiRealizada.class);
  PredicateSchema modeloUsuariosSchema = new PredicateSchema(MODELOUSUARIOS);
  add(modeloUsuariosSchema, ontology.ModeloUsuarios.class);
  PredicateSchema grupoUsuariosSchema = new PredicateSchema(GRUPOUSUARIOS);
  add(grupoUsuariosSchema, ontology.GrupoUsuarios.class);

  // adding fields
  visitaUsuarioSchema.add(VISITAUSUARIO_PAGINA, paginaSchema, ObjectSchema.MANDATORY);
  visitaUsuarioSchema.add(VISITAUSUARIO_TEMPO, (TermSchema)getSchema(BasicOntology.FLOAT),
ObjectSchema.MANDATORY);
  matrizCaracteristicasSchema.add(MATRIZCARACTERISTICAS_COLUNAS,
(TermSchema)getSchema(BasicOntology.INTEGER), ObjectSchema.OPTIONAL);
  matrizCaracteristicasSchema.add(MATRIZCARACTERISTICAS_LINHAS,
(TermSchema)getSchema(BasicOntology.INTEGER), ObjectSchema.OPTIONAL);
  matrizCaracteristicasSchema.add(MATRIZCARACTERISTICAS_MATRIZES,
(AggregateSchema)getSchema(BasicOntology.SEQUENCE), 1, ObjectSchema.UNLIMITED);
  posicaoSchema.add(POSICAO_COLUNA, (TermSchema)getSchema(BasicOntology.INTEGER),
ObjectSchema.OPTIONAL);
  posicaoSchema.add(POSICAO_LINHA, (TermSchema)getSchema(BasicOntology.INTEGER),
ObjectSchema.OPTIONAL);
  paginaSchema.add(PAGINA_ENDERECO, (TermSchema)getSchema(BasicOntology.STRING),
ObjectSchema.MANDATORY);
  espacoPaginasSchema.add(ESPACOPAGINAS_PAGINAS, paginaSchema, 1, ObjectSchema.UNLIMITED);
  sessaoUsoSchema.add(SESSAOUSO_VISITAS, visitaUsuarioSchema, 1, ObjectSchema.UNLIMITED);
  sessaoUsoSchema.add(SESSAOUSO_ORDEMSEGMENTO, (TermSchema)getSchema(BasicOntology.INTEGER),
ObjectSchema.MANDATORY);
  segmentoUsoSchema.add(SEGMENTOUSO_VISITAS, visitaUsuarioSchema, 1, ObjectSchema.UNLIMITED);
  foiRealizadaSchema.add(FOIREALIZADA_VISITA, visitaUsuarioSchema, ObjectSchema.OPTIONAL);
  modeloUsuariosSchema.add(MODELOUSUARIOS_SESSAOUSO, sessaoUsoSchema, ObjectSchema.OPTIONAL);
  modeloUsuariosSchema.add(MODELOUSUARIOS_MATRIZTEMPO, matrizCaracteristicasSchema,
ObjectSchema.MANDATORY);
  modeloUsuariosSchema.add(MODELOUSUARIOS_MATRIZVISITAS, matrizCaracteristicasSchema,
ObjectSchema.MANDATORY);
  modeloUsuariosSchema.add(MODELOUSUARIOS_MATRIZSEQUENCIA, matrizCaracteristicasSchema,
ObjectSchema.MANDATORY);
```

```
   grupoUsuariosSchema.add(GRUPOUSUARIOS_MATRIZVISITAS, matrizCaracteristicasSchema,
ObjectSchema.MANDATORY);
   grupoUsuariosSchema.add(GRUPOUSUARIOS_MATRIZTEMPO, matrizCaracteristicasSchema,
ObjectSchema.MANDATORY);
   grupoUsuariosSchema.add(GRUPOUSUARIOS_MATRIZSEQUENCIA, matrizCaracteristicasSchema,
ObjectSchema.MANDATORY);
   grupoUsuariosSchema.add(GRUPOUSUARIOS_GRUPONUM,
(TermSchema)getSchema(BasicOntology.INTEGER), ObjectSchema.MANDATORY);
   grupoUsuariosSchema.add(GRUPOUSUARIOS_CARDINALIDADE,
(TermSchema)getSchema(BasicOntology.INTEGER), ObjectSchema.MANDATORY);

   // adding name mappings
   storeSlotName("pagina", "ontology.VisitaUsuario", "pagina");
   storeSlotName("tempo", "ontology.VisitaUsuario", "tempo");
   storeSlotName("colunas", "ontology.MatrizCaracteristicas", "colunas");
   storeSlotName("linhas", "ontology.MatrizCaracteristicas", "linhas");
   storeSlotName("matrizes", "ontology.MatrizCaracteristicas", "matrizes");
   storeSlotName("coluna", "ontology.Posicao", "coluna");
   storeSlotName("linha", "ontology.Posicao", "linha");
   storeSlotName("endereco", "ontology.Pagina", "endereco");
   storeSlotName("paginas", "ontology.EspacoPaginas", "paginas");
   storeSlotName("visitas", "ontology.SessaoUso", "visitas");
   storeSlotName("ordemSegmento", "ontology.SessaoUso", "ordemSegmento");
   storeSlotName("segmentoUso", "ontology.SegmentoUso", "visitas");
   storeSlotName("visita", "ontology.FoiRealizada", "visita");
   storeSlotName("sessaoUso", "ontology.ModeloUsuarios", "sessaoUso");
   storeSlotName("matrizTempo", "ontology.ModeloUsuarios", "matrizTempo");
   storeSlotName("matrizVisitas", "ontology.ModeloUsuarios", "matrizVisitas");
   storeSlotName("matrizSequencia", "ontology.ModeloUsuarios", "matrizSequencia");
   storeSlotName("matrizTempo", "ontology.GrupoUsuarios", "matrizTempo");
   storeSlotName("matrizVisitas", "ontology.GrupoUsuarios", "matrizVisitas");
   storeSlotName("matrizSequencia", "ontology.GrupoUsuarios", "matrizSequencia");
   storeSlotName("grupoNum", "ontology.GrupoUsuarios", "grupoNum");
   storeSlotName("cardinalidade", "ontology.GrupoUsuarios", "cardinalidade");

   // adding inheritance

  }catch (java.lang.Exception e) {e.printStackTrace();}
 }
 }
```

```
package Minerador;

import jade.content.lang.Codec;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.Ontology;
import jade.core.behaviours.TickerBehaviour;
import jade.util.leap.List;
import jade.wrapper.AgentContainer;
import ontology.OntowumOntology;


public class RemontaModelos extends TickerBehaviour{
                        private Minerador agent;
            private Codec codec = new SLCodec();
            private Ontology ont = OntowumOntology.getInstance();;
            /* (non-Javadoc)
             * @see jade.core.behaviours.Behaviour#action()
             *
             */
   public RemontaModelos(Minerador a, long duracao) {
                        super(a,duracao);
                        this.agent=a;
            }

            protected void onTick() {
                        LogHandler fileHandler = new
LogHandler("http://maae.deinf.ufma.br/ONTOWUM/modeloUsuarios#","Log.rdf");
                        List listaModelos = fileHandler.remontarModelosUsuarios();
                        agent.kMeans = new KMeans(15,listaModelos);
                        agent.kMeans.runKMeans();
                        //agent.kMeans.imprimeGrupos();
            }
            }
```