Universidade Federal de Santa Catarina – UFSC

SISTEMA DE APOIO À APLICAÇÃO DA METODOLOGIA COMMONKADS EM PROJETOS DE ENGENHARIA DO CONHECIMENTO

Cleosvaldo G. Vieira Junior

Florianópolis
2005/2

Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Curso Bacharelado em Sistemas de Informação

SISTEMA DE APOIO À APLICAÇÃO DA METODOLOGIA COMMONKADS EM PROJETOS DE ENGENHARIA DO CONHECIMENTO

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Cleosvaldo G. Vieira Junior

Prof. Dr. Roberto Carlos Pacheco
Orientador

Florianópolis
2005/2

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

RESUMO

A aplicação da Gestão do Conhecimento nas organizações tem crescido constantemente. Cada vez mais empresas procuram gerir o conhecimento de forma estratégica para tornarem-se mais competitivas.

A Gestão do Conhecimento possui íntima relação com a Engenharia do Conhecimento, visto que toda a análise, modelagem e estudos necessários para gerar sistemas para Gestão do Conhecimento estão fundamentados nas teorias de conceitos definidos na Engenharia do Conhecimento. Diferentes metodologias foram criadas para auxiliar o Engenheiro do Conhecimento. Atualmente a metodologia mais completa e utilizada é o CommonKADS.

A metodologia CommonKADS traz um conjunto de modelos que abrange todas as etapas para o projeto de um sistema de conhecimento, abrangendo desde a análise e verificação de viabilidade de uma solução até o gerenciamento do projeto da solução escolhida.

Apesar de ser uma metodologia madura e que existe há cerca de 20 anos, não se encontram disponíveis muitas ferramentas capazes de auxiliar o Engenheiro do Conhecimento no que diz respeito à gestão da documentação gerada a partir da aplicação do CommonKADS.

O presente trabalho propõe um sistema de informação com a finalidade de armazenar e gerenciar as análises e resultados gerados a partir da aplicação dos modelos da metodologia CommonKADS. Como resultado, o sistema permite a fácil recuperação e reutilização das informações registradas, o que diminui significativamente o trabalho que o Engenheiro do Conhecimento tem ao documentar sua análise.

Ao final deste trabalho são apresentados os trabalhos futuros que podem ser realizados a partir deste, dentre os quais é possível destacar a geração automática de código CML (Knowledge Modelling Language) a partir das informações registradas.

Palavras-chave: CommonKADS, Engenharia do Conhecimento, Gestão do Conhecimento.

1 INTRODUÇÃO

1.1 Contexto

A Engenharia do Conhecimento deu seus primeiros passos na década de 60 com a criação de sistemas especialistas setorizados. Estes sistemas resumiam-se na transferência do conhecimento humano para uma base de conhecimento que era implementada de forma a tornar tal conhecimento acessível. O foco principal estava nos profissionais especialistas que através de entrevistas explicitavam como resolviam tarefas específicas. A partir dai estas informações eram armazenadas e criavam-se regras para manipulá-las.

Com a evolução das pesquisas, a partir da década de 80, deu-se início a segunda geração de sistemas especialistas, onde a proposta de transferência de conhecimento cedeu espaço para a proposta de modelagem conhecimento (Benjamins apud Alkaim, 2003, p.107), que trouxe consigo maior suporte para a área de aquisição de conhecimento.

A percepção de que o conhecimento não deve ser perdido e pode ser utilizado de forma estratégica, desde que gerido corretamente, fez com que a produção de sistemas para gestão de conhecimento aumentasse e encontrasse na Engenharia do Conhecimento os fundamentos e métodos necessários para conceber sistemas com a capacidade de atender as diversas situações e necessidades das grandes corporações.

1.1.1 Engenharia e Gestão do Conhecimento

Inicialmente os sistemas especialistas eram setorizados e isolados. Com o surgimento dos sistemas de conhecimento intensivo aumentou-se potencialmente a possibilidade de criar sistemas capazes de fornecer uma análise global do conhecimento das organizações, agilizar processos, e fornecer formas de gerir todo conhecimento da organização de forma estratégica.

A Engenharia do Conhecimento guarda íntima relação com a Gestão do Conhecimento, considerando as organizações como inimagináveis sem a utilização de avançados sistemas de informação, baseados nos avanços da Engenharia de Software, aplicados a projetos em que conhecimento e raciocínio são centrais (Schreiber et. al.,2002).

Segundo Schereiber et. al., 2002, a Engenharia do Conhecimento oferece muitos conceitos e métodos úteis à Gestão do Conhecimento:

- Análise da organização orientada a conhecimento ajuda a rapidamente se mapear áreas proveitosas para ações de gestão do conhecimento. Os métodos da Engenharia do Conhecimento podem ser úteis para rapidamente procurar ou auditar conhecimento.
- Análise da tarefa e do agente que a executa tem sido muito útil para identificar gargalos de conhecimento em áreas específicas. Não é incomum que isso mostre que os resultados sejam diferentes do que era esperado na organização. Técnicas como essas são relevantes ao reprojeto de processos do negócio e a melhoramentos do trabalho intensivo em conhecimento. CommonKADS fornece uma transição gradual entre a análise do negócio e da informação, o que é chave, também, para melhor compreender a integração da Tecnologia da Informação na organização.
- A Engenharia do Conhecimento coloca uma forte ênfase na modelagem de atividades intensivas em conhecimento. Técnicas gráficas têm se mostrado muito úteis em clarificar a maioria dos aspectos tácitos do conhecimento de forma (não-técnica e não-sistêmica) a capacitar e estimular comunicação proveitosa com uma variedade de pessoas (gestores, especialistas, usuários finais e consumidores) que geralmente não têm um background em tecnologia da informação.
- A experiência acumulada da Engenharia do Conhecimento mostra que existem muitas estruturas e mecanismos recorrentes em trabalho intensivo em conhecimento. Isso tem levado, por exemplo, a bibliotecas de modelos de tarefa que são aplicáveis em diferentes domínios. Essa abordagem oferece muitos insights proveitosos na construção de arquiteturas de informação reutilizáveis e

componentes de software que são cada vez mais necessários em organizações modernas baseadas em TI.

A Engenharia do Conhecimento estabelece metodologia, métodos e ferramentas para o ciclo de identificação de necessidades, concepção e desenvolvimento de sistemas para codificação, armazenagem e apoio à gestão do conhecimento organizacional.

1.1.2 Metodologias da Engenharia do Conhecimento

O surgimento da segunda geração de sistemas especialistas veio a partir da noção de níveis de conhecimento, introduzida por Allen Newell nos anos 80. Nesta abordagem, o desenvolvimento de um sistema de conhecimento parte da construção de modelos que se relacionam segundo um comportamento de solução para o problema. Dessa forma, o sistema de conhecimento independe da linguagem de programação, pois ele é visto como um agente que atua como se possuísse conhecimento e utilizasse-o de modo racional para atingir seus objetivos (Abel, 2003).

Ao longo da história de desenvolvimento de sistemas de conhecimento, novas abordagens e ferramentas de análise, aquisição de conhecimento e projeto de sistemas inteligentes fora propostas. Para a modelagem de sistemas alguns frameworks[1] de modelagem foram criados e, a partir desses, foram estabelecidas metodologias de desenvolvimento:

- **SPEDE**: uma combinação de princípios, técnicas e ferramentas retiradas da Engenharia do Conhecimento para o uso na Gestão do Conhecimento.
- **MOKA** - *Methodology and tools Oriented to Knowledge-Based Engineering Applications*. Mais utilizado atualmente em indústrias aeronáuticas e automotivas.
- **VITAL:** metodologia baseada em ciclos de vida que visa o apoio ao desenvolvimento de sistemas de conhecimento em larga escala.

[1] Existem diversas definições de framework, mas em linhas gerais pode ser definido como um conjunto de especificações que serve de base para a construção de aplicações (Zamai 2005, apud Cavaness, 2003)

- **CommonKADS** – framework para projetos de sistemas de conhecimento. É considerado o mais completo atualmente por fornecer suporte a vários aspectos do desenvolvimento de projetos de Engenharia do Conhecimento. O CommonKADS traz a união de vários conceitos de outras metodologias e por isso tornou-se o mais difundido.

1.1.3 A Metodologia CommonKADS

O CommonKADS é uma metodologia que integra características de metodologias orientadas a modelos e abrange diversos aspectos do projeto de desenvolvimento de um sistema de conhecimento, incluindo: análise organizacional; gerenciamento de projetos; aquisição, representação e modelagem do conhecimento; integração e implementação de sistemas.

Esta metodologia possui um conjunto de 6 modelos que especificam todos os aspectos ligados à aplicação a ser desenvolvida, incluindo a organização, os recursos humanos, os aspectos de implementação e a interação entre eles, estes modelos são: Organizacional, Tarefas, Agentes, Conhecimento, Comunicação e Projeto. O CommonKADS dá suporte à realização de três atividades principais: atividades de modelagem, atividades de gestão de projetos e reusabilidade.

1.1.4 Ferramenta de apoio ao Engenheiro de Conhecimento

A aplicação da Metodologia CommonKADS resulta em uma série de planilhas e documentos associados, de acordo com a evolução do trabalho do engenheiro de conhecimento. O CommonKADS faz uso da UML[2] (Booch et al., 2000), para auxiliar na descrição simbólica da implementação dos seus modelos, e da CML2 (Knowledge Modelling Language) para uma especificação de forma estruturada e seguindo características de programação.

1.2 Objetivos

1.2.1 Objetivo Geral

O presente trabalho tem por objetivo conceber, especificar e desenvolver um sistema de informação para o apoio à aplicação da Metodologia CommonKADS em projetos de Engenharia do Conhecimento.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral do trabalho, serão necessários, ainda, os seguintes objetivos específicos:

- Realizar um levantamento das partes componentes da Metodologia CommonKADS, destacando cada elemento informacional que especifica seus componentes.

[2] UML: Unified Modeling Language (Booch et al., 2000)

- Realizar um levantamento na literatura sobre exemplos de aplicação da Metodologia CommonKADS, de forma a estabelecer conjuntos amostrais para aplicação do sistema que será desenvolvido.
- Especificar (com utilização da Linguagem UML) o modelo do Sistema de Informação proposto;
- Desenvolver o sistema, conforme os requisitos especificados nos levantamentos da metodologia CommonKADS e de seus exemplos;
- Aplicar ao Sistema desenvolvido os exemplos discutindo a efetividade do sistema e a praticidade que o mesmo proporciona para o engenheiro do conhecimento.

1.2.3 Justificativa e Motivações

O conhecimento é um dos maiores valores existentes para qualquer pessoa, organização ou instituição. Saber lidar com ele de forma metódica, para obter soluções e resultados, tem se tornado algo cada vez mais necessário e presente. Com isso, a Engenharia e a Gestão do Conhecimento representam um papel fundamental no desenvolvimento do que podemos chamar de “Era do Conhecimento”.

Apesar da Engenharia do Conhecimento existir desde a década de 60, a metodologia CommonKADS, que surgiu na década de 80, e atualmente é a mais difundida, possui poucas ferramentas que auxiliem efetivamente na sua aplicação.

A metodologia CommonKADS possui vários componentes que são utilizados conforme o contexto. Da aplicação dos modelos da metodologia tem-se como resultado planilhas e documentos. Gerenciar as informações destas planilhas e documentos de forma a conectar as etapas do CommonKADS ainda é uma carência do CommonKADS.

A modesta contribuição deste trabalho pode ser o primeiro passo para que ocorram outras iniciativas que fortaleçam o crescimento desta metodologia da Engenharia do Conhecimento.

1.3 Estrutura do Trabalho

O objetivo desta seção é apresentar a estrutura do trabalho trazendo uma breve descrição do que será apresentado, em cada uma das seções, e sua finalidade.

O Capítulo 1 apresenta a contextualização do tema fazendo uma breve explanação sobre a Engenharia e Gestão do Conhecimento, mostrando sua importância no contexto organizacional. É apresentada a metodologia CommonKADS bem como os objetivos, motivações e justificativas para a execução deste trabalho, a estrutura do trabalho e suas delimitações também são descritas neste capítulo.

O Capítulo 2 apresenta um levantamento das características da metodologia CommonKADS, seus componentes e sua utilização. O objetivo deste capítulo é aproximar o leitor dos conceitos e modelos do CommonKADS de forma que se possa entender a finalidade do sistema de informação proposto neste trabalho.

O Capítulo 3 traz a definição do sistema de informações proposto, seus requisitos, especificações e aspectos técnicos de desenvolvimento. O objetivo deste capítulo é explicitar o processo de elaboração do sistema de informação proposto.

O Capítulo 4 relata casos, encontrados na literatura, onde o CommonKADS foi utilizado e a aplicação destes casos no sistema desenvolvido, como forma de testar seu funcionamento e adequação. Este capítulo mostrará qual será a eficiência do sistema diante de uma situação de uso real.

O Capítulo 5 apresenta as conclusões sobre o trabalho elaborado e sugestões para trabalhos futuros.

No Capítulo 6 são apresentadas as referências bibliográficas e a seguir os anexos.

1.4 Delimitações do Trabalho

A implementação de projetos de Engenharia do Conhecimento é em si tema abrangente e escopo de pesquisa e formação de áreas de concentração de programas de pós-graduação. No presente projeto, no entanto, está-se delimitando o foco no desenvolvimento de uma ferramenta que apóie o engenheiro na documentação e gestão das informações geradas durante o processo de aplicação do CommonKADS. Com isso, não faz parte deste trabalho estabelecer estudos de aplicação ou aprofundamento da metodologia CommonKADS propriamente dita. O que se estará desenvolvendo é, em realidade, um sistema de informação que permita ao Engenheiro de Conhecimento que conheça a metodologia, registrar os artefatos que a mesma gera ao longo de sua aplicação.

Também não estão incluídos no escopo deste trabalho a geração automática de código CML, a partir dos dados registrados, bem como a implementação de recursos de segurança, tais como: controle de usuários e restrições de acesso a informações.

2 A Metodologia CommonKADS

Segundo Schreiber et al., 2002, o CommonKADS originou-se da necessidade de construir sistemas de conhecimento de qualidade em larga escala, de forma estruturada, controlável e repetível.

Trata-se de uma metodologia que integra características de outras metodologias orientadas a modelos e abrange diversos aspectos do projeto de desenvolvimento de um sistema de conhecimento, incluindo: análise organizacional; gerenciamento de projetos; aquisição, representação e modelagem do conhecimento; integração e implementação de sistemas (Freitas, 2003).

O CommonKADS possui um conjunto de 6 modelos que especificam todos os aspectos ligados à aplicação a ser desenvolvida, incluindo a organização, os recursos humanos, os aspectos de implementação e a interação entre eles. Além disso, oferece suporte à realização de atividades de modelagem, atividades de gestão de projetos e reusabilidade (Schreiber et al., 2002)

A experiência acumulada ao longo dos anos tornou o conjunto de modelos do CommonKADS a expressão prática dos princípios de base da análise de conhecimento. Como conseqüência disso o CommonKADS atualmente é a metodologia mais difundida e testada em projetos reais (Freitas, 2003).

Segundo Olsson (apud Freitas, 2003) o CommonKADS é uma metodologia para desenvolvimento de sistemas baseados em conhecimento, resultante do projeto ESPRIT-II (P5248) e KADS II (Knowledge Analysis and Documentation System, posteriormente Knowledge Analysis and Design Support) iniciado em 1990 e terminado em 1994.

O KADS-II surgiu como sucessor ao projeto KADS (P1098) que terminou em 1989. Ele foi adotado por muitas companhias e organizações de pesquisa principalmente da Europa e EUA. CommonKADS está fortemente posicionado em seu padrão de desenvolvimento de Sistemas Baseados em Conhecimento na Europa. A Figura 2, mostrada a seguir, resume a evolução do CommonKADS:

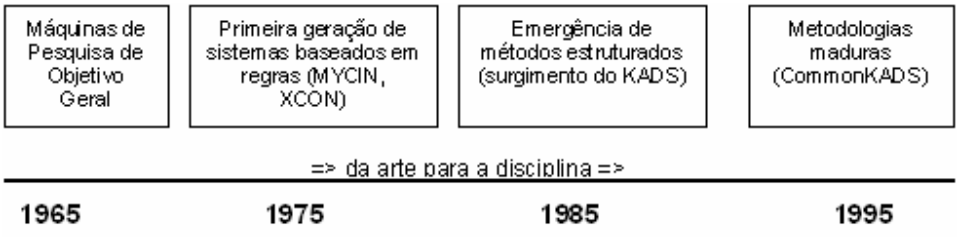

Figura 1: Evolução do CommonKADS (Fonte: [Alkaim, 2003])

A abrangência da metodologia CommonKADS é significativamente grande. Ela fornece recursos para todos os profissionais do conhecimento: analista, gerente, desenvolvedor e gerente de projetos. Outras características importantes do CommonKADS:

- Traz métodos para a análise organizacional (incluindo problemas / identificando oportunidades), teste de viabilidade e aderência de soluções;
- Modelos para identificação de tarefas, processos, agentes e a comunicação dentro da organização.
- Técnicas de aquisição de conhecimento;
- Possui *templates* de modelos de conhecimento;
- Notação UML para definição de todas as etapas de análise e projeto;

Para prover todos os recursos, o CommonKADS é dotado de modelos agrupados por assuntos, conforme será apresentado a seguir.

2.1 Componentes da Metodologia CommonKADS

Segundo Schreiber et al., 2002, para estabelecer um projeto de engenharia do conhecimento, o engenheiro deve definir o contexto do projeto (i.e., a organização onde ocorrerá o projeto, os processos com tarefas de conhecimento intensivo e os responsáveis

pela realização dessas tarefas). Uma vez estabelecidos esses elementos, o engenheiro deve formalizar os componentes que conceituam o projeto de engenharia do conhecimento (i.e., o conhecimento e a comunicação entre os atores responsáveis), para, finalmente, estabelecer o tipo de projeto de conhecimento que realizará. Para tal, a metodologia CommonKADS está composta pelos seguintes modelos:

- **Modelo da Organização** - Apóia a análise das maiores características da organização, a fim de descobrir problemas e oportunidades para sistemas de conhecimento, estabelecer sua viabilidade e acessar o impacto das ações de conhecimento pretendidas na organização.
- **Modelo da Tarefa -** analisa o layout das principais tarefas do domínio, suas entradas, saídas, pré-condições e critérios de performance, bem como recursos e competências necessários. Com a aplicação deste modelo tem-se a identificação de quais tarefas possuem conhecimento intensivo.
- **Modelo do Agente -** descreve as características dos agentes, em particular suas competências, autoridades e restrições para agir. Além disso, relaciona os links de comunicação entre agentes necessários para executar uma tarefa.
- **Modelo do Conhecimento –** descreve o conhecimento envolvido no domínio do projeto. Com este modelo é possível detalhar como o conhecimento está relacionado em cada tarefa, quais agentes o possuem e como seus componentes relacionam-se entre si.
- **Modelo de Comunicação -** Dado que muitos agentes podem estar envolvidos em uma tarefa, é importante modelar a transação de comunicação entre os agentes envolvidos. Isso é feito pelo modelo de comunicação, de forma independente de implementação ou de conceito, como ocorre no modelo de conhecimento.

- **Modelo do Projeto** – Os modelos do CommonKADS compõem a especificação necessária para a criação de um sistema de conhecimento. O modelo do projeto conterá, então, a conversão das informações contidas nos modelos em

especificações técnicas do sistema quanto a arquitetura, plataforma de implementação, módulos de softwares, construtores de representação, e mecanismos computacionais necessários para implementar as funções verificadas nos modelos de conhecimento e comunicação (Alkaim,2003).

2.1.1 Atividades de Modelagem

A aplicação dos modelos do CommonKADS na organização tem o objetivo de responder três questionamentos principais, através dos quais serão definidos os caminhos a serem seguidos e as soluções a serem escolhidas. Schreiber et al., 2002, apresentam estes questionamentos da seguinte forma:

a) *Por quê?* – Porque um sistema de conhecimento é uma solução em potencial? Para quais problemas? Que benefícios, custos e impactos organizacionais ele terá? O entendimento do ambiente e do contexto organizacional é o ponto mais importante deste questionamento.

b) *Qual?* – consiste em entender qual é a natureza e estrutura do conhecimento envolvido, bem como a natureza e estrutura de comunicação correspondente. Obter a descrição conceitual do conhecimento utilizado na realização de uma tarefa é um dos pontos chaves deste questionamento.

c) *Como?* – Como o conhecimento deve ser implementado no sistema computacional? Como deve ser a infra-estrutura tecnológica necessária para a construção e execução do sistema? Os aspectos técnicos da implementação são o principal foco neste questionamento.

Através da Figura 3 é possível perceber que cada um dos modelos está focado num determinado aspecto do sistema e relacionam-se entre si.

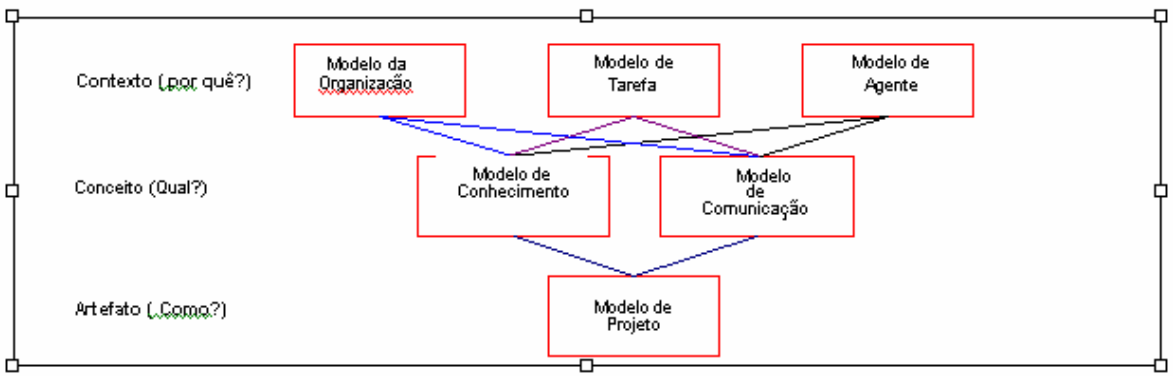

Figura 2 Modelos da Metodologia CommonKADS (Fonte: [SCHREIBER et al., 2002])

A aplicação de cada um desses modelos dependerá da necessidade e da experiência que o engenheiro de conhecimento possui. Os modelos são independentes, não sendo necessário que todos sejam aplicados.

2.1.2 Modelo da Organização

O propósito do Modelo da Organização é traçar o perfil da organização, seus problemas e oportunidades, seus processos e o conhecimento que está envolvido em cada tarefa (com o qual se pretende criar um Sistema de Conhecimento).

A aplicação do modelo dá-se através de cinco tabelas, onde serão descritas informações como, por exemplo, a estrutura organizacional e as funções que são executadas em cada uma das unidades da organização. A partir da análise das informações obtidas em cada tabela é possível identificar as deficiências em cada processo de negócio, bem como identificar quais serão os impactos e melhorias que um Sistema de Conhecimento poderá trazer (Scheiber et al.,2002).

Cada uma das tabelas do Modelo da Organização possui uma abordagem da organização, dessa forma, ao final, tem-se as seguintes perspectivas: estrutura organizacional, atividades, processos de negócios, pessoas e recursos.

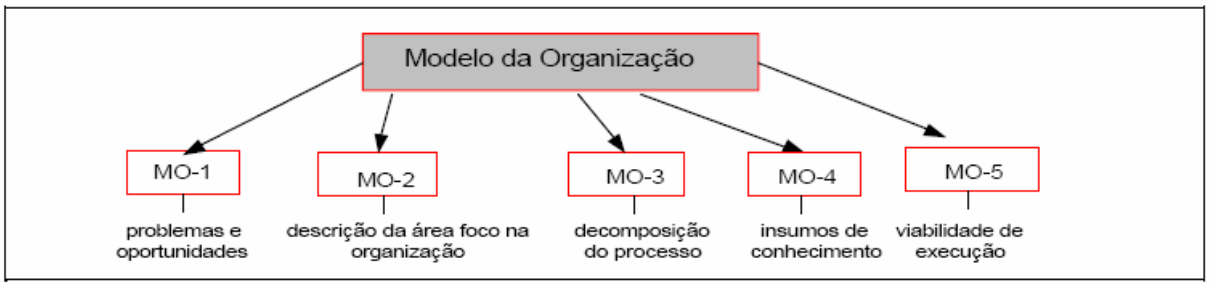

Figura 3: Visão geral do Modelo da Organização (Fonte: [SCHREIBER et al., 2002])

A primeira parte do modelo (MO-1) é voltada para os problemas e oportunidades. Nela o analista, através de entrevistas, brainstorms e conversas com gerentes, criará uma pequena lista de problemas e oportunidades.

Em seguida ele descreverá o contexto da organização a fim de se posicionar melhor os problemas e oportunidades dentro da realidade da organização, para isso leva-se em consideração: a missão, a visão, os objetivos da organização, estratégias, cadeia de valores e fatores externos. Dessa forma, problemas, oportunidades e soluções baseadas em conhecimento devem ser julgados dentro de uma perspectiva empresarial ampla e real, sendo necessária uma compreensão realista e explícita desse contexto.

Após todo o levantamento dos problemas e oportunidade e como eles se enquadram dentro da organização, o analista criará uma lista de possíveis soluções para os problemas percebidos. As possíveis soluções para os problemas e oportunidades podem ser obtidas por meio de entrevistas e discussões, bem como do contexto organizacional com pessoas-chave da organização (Scheiber *et al.*, 2002)

A seguir é apresentada tabela MO-1 segundo a estrutura proposta por Schreiber et al., 2002:

Modelo da Organização	***Problemas e oportunidades – Tabela MO-1***
PROBLEMAS E OPORTUNIDADES	Criar uma pequena lista dos problemas e oportunidades, baseando-se em entrevistas, *brainstorm*s, conversas com gerentes, etc.
CONTEXTO ORGANIZACIONAL	Identificar de uma maneira concisa "características-chave" do contexto organizacional, assim como colocar as oportunidades e problemas listados numa perspectiva própria. Características importantes que devem ser consideradas: 1. Missão, visão e objetivos da organização 2. Fatores externos à organização que podem estar ligados com estratégias da organização. 3. Estratégias da organização 4. Cadeias de valores que conduzem à empresa.
SOLUÇÕES	Listar as possíveis soluções para os problemas percebidos, como sugerido nas entrevistas e discussões, e sobre as características do contexto organizacional.

Tabela 1: MO-1 Identificação dos problemas e oportunidades da organização (Fonte: (Schreiber et al., 2002))

"A segunda parte do modelo (MO-2) concentra-se nos aspectos que influenciam ou são afetados pelas soluções apontadas. Assim, aspectos como processos de negócio, *staff* envolvida, recursos utilizados, cultura organizacional etc., são componentes do modelo que podem mudar como resultado da introdução dos sistemas de conhecimento." [FREITAS, 2003]

Será no MO-2 que o analista fará o levantamento da estrutura organizacional, seus departamentos, grupos, unidades, seções, etc. Esta estrutura normalmente é representada de forma gráfica. O levantamento de processos, pessoas envolvidas, recursos utilizados nos processos de negócios, bem como o conhecimento envolvido são itens que serão listados neste momento para serem detalhados nos modelos seguintes.

A última parte do MO-2 refere-se a aspectos de cultura e poder na organização. Este é um ponto muito importante, pois trata das regras escritas (e não escritas) da empresa, o estilo de trabalho, a comunicação (que quase nunca obedece a um organograma). Os aspectos de cultura e poder são importantes porque podem ter influência direta na viabilidade de uma solução.

O MO-2 pode ser estruturado em uma tabela, conforme é apresentado na tabela 2:

MODELO DA ORGANIZAÇÃO	Aspectos variáveis – Tabela MO-2
ESTRUTURA	Criar uma representação gráfica da organização (ou parte dela) mostrando departamentos, grupos, unidades, seções, etc.
PROCESSOS	Esboço do layout (com a ajuda de UML) da manipulação dos processos de negócios. Um processo é uma parte relevante na cadeia de valores que está recebendo foco no projeto. Um processo é decomposto em tarefas que serão detalhas na tabela OM-3
PESSOAS	Indicar que grupo de funcionários está envolvido, como atores ou patrocinadores, incluindo tomadores de decisões, fornecedores, usuários e beneficiários (clientes) do conhecimento. Estas "pessoas" não precisam necessariamente ser pessoas que existem na empresa, mas podem ser papéis funcionais definidos na organização (por um diretor, ou consultor, por exemplo)
RECURSOS	Descrever os recursos que são utilizados nos processos de negócios. Os tipos mais abrangentes de recursos: 1. Sistemas de Informação e outros recursos computacionais 2. Equipamentos e materiais 3. Tecnologias, patentes e direitos.
CONHECIMENTO	Conhecimento representa um recurso especial explorado no processo de negócios. Ele tem uma importância chave neste contexto, por isso precisa ser detalhado separadamente. A descrição deste componente do modelo da organização será feita na tabela OM-4
CULTURA & PODER	Prestar muita atenção nas regras que não estão escritas, incluindo estilo de trabalho e comunicação (que quase nunca obedece a um organograma), relações pessoais e experiências interpessoais (sem conhecimento), e redes de formais e informais.

Tabela 2: MO-2 – Descrição de aspectos organizacionais que receberão um impacto ou serão afetadas pela solução escolhida. (Fonte: (Schreiber et al., 2002))

A terceira parte do modelo organizacional (MO-3) realiza uma análise dos processos de negócios. Trata-se de um detalhamento dos processos levantados no MO-2 onde serão apresentadas quais são as tarefas que compõem cada processo.

Neste momento, cada tarefa será avaliada, e deverão estar explícitos os agentes que executam cada tarefa, qual o tipo de conhecimento que é utilizado, se este conhecimento é intensivo e qual é significância desta tarefa dentro do domínio do problema.

A análise detalhada de um processo pode resultar na percepção de processos falhos o que mostraria a necessidade de realizar uma reengenharia de processos (Schreiber et al., 2002).

A maneira mais fácil de estruturar a análise feita segundo o MO-3 é através de uma tabela, conforme a estrutura mostrada na tabela 3:

MODELO DA ORGANIZAÇÃO		**Desdobramento de processos – Tabela MO-3**				
N°	TAREFA	EXECUTADA POR:	ONDE?	CONHECIMENTO UTILIZADO.	INTENSIVO?	SIGNIFICÂNCIA?
N° seqüencial (identificador)	Nome da tarefa (alguma parte do processo descrito na MO-2)	Pode ser algum agente, um humano (veja "pessoas" em MO-2) ou um software (veja "recursos" em MO-2)	Algum local da organização	Lista de recursos de conhecimento usados nesta tarefa	Indicador lógico (Sim/ Não) indicando se: a tarefa é considerada de conhecimento intensivo?	Indicação de quão significante a tarefa pode ser considerada (por exemplo, numa escala de 5 pontos, considerando: freqüência, custos, recursos e missão crítica.)

Tabela 3: MO-3 – Descrição dos processos em termos de que tarefas o compõem. (Fonte: SCHREIBER et al., 2002)

A quarta parte do modelo da organização (MO-4) traz a relação de conhecimentos utilizados nas tarefas, e explicita alguns detalhes sobre eles. Este modelo vem trazer um *overview* do que será detalhado no modelo de conhecimento.

Neste momento, será descrito apenas quem possui o conhecimento, onde ele é utilizado, se sua utilização ocorre de forma correta, no lugar correto e no tempo e qualidade corretos. Esta pequena descrição permitirá olhar de forma crítica o conhecimento da organização e avaliá-lo de forma a obter a melhor solução para os problemas levantados.

MODELO DA ORGANIZAÇÃO		Conhecimentos utilizados – Tabela MO-4				
CONHECIMENTO UTILIZADO.	QUEM POSSUI?	USADO EM	DE FORMA CORRETA?	NO LUGAR CERTO?	NO TEMPO CERTO?	QUALIDADE CERTA?
Nome do conhecimento (conforme MO-3)	Agente que possui o conhecimento (conforme MO-3)	Algum lugar na estrutura da organização (veja MO-2)	(Sim ou não; comentários)	(Sim ou não; comentários)	(Sim ou não; comentários)	(Sim ou não; comentários)

Tabela 4: MO-4 – Descrição dos componentes de conhecimento do modelo da organização. (Fonte: (SCHREIBER et al., 2002))

O último passo do modelo organizacional (OM-5) consiste em documentar e avaliar as implicações das informações, colhidas em cada modelo, frente às soluções propostas para o problema. O objetivo aqui é verificar os benefícios e a viabilidade do desenvolvimento do sistema de conhecimento. [FREITAS, 2005]

A tabela 5 apresenta o checklist do OM-5:

MODELO DA ORGANIZAÇÃO	Checklist: Documento de Decisão de Viabilidade - Tabela MO-5
VIABILIDADE DE NEGÓCIOS	Para os dados problemas/oportunidades e soluções, os seguintes questionamentos devem ser feitos: 1. Quais são os benefícios esperados, para a organização, da solução considerada? 2. Quão grande é esta expectativa de adição de valores? 3. Quais são os custos esperados para a solução considerada? 4. Quanto é possível comparar esta solução com outras soluções? 5. Será precisa fazer mudanças organizacionais? 6. Quais são os riscos de negócios e econômicos e as incertezas envolvidas na direção da solução considerada.
VIABILIDADE TÉCNICA	Para os dados problemas/oportunidades e soluções, os seguintes questionamentos devem ser feitos: 1. Quão complexa, em termos de armazenamento de conhecimento e processo de raciocínio, é a tarefa realizada pela solução de sistema de conhecimento considerada? 2. Considerando tempo, qualidade, recursos necessários, ou outros. Existem aspectos críticos envolvidos? Se sim, como são resolvidos? 3. Está claro quais são as medidas de sucesso e como testar a validade, qualidade e performance satisfatória? 4. Quão complexa é a interface com o usuário? Os métodos e técnicas estão disponíveis e são adequados?

	5. Quão complexa é a interação com outros Sistemas de Informação e outros possíveis recursos (interoperabilidade, integração de sistemas)? Os métodos e técnicas estão disponíveis e são adequados?
VIABILIDADE DE PROJETO	Para os dados problemas/oportunidades e soluções, os seguintes questionamentos devem ser feitos: 1. Existe comprometimento dos atores e patrocinadores (gerentes, especialistas, usuários, clientes, membros da equipe de projeto) para ajudar nas etapas do projeto? 2. Os recursos em termos de tempo, orçamento, equipamentos e pessoal estarão disponíveis? 3. O conhecimento necessário e outras competências estão disponíveis? 4. As expectativas voltadas para o projeto e seus resultados são realistas? 5. O projeto da organização e suas comunicações interna e externa são adequadas? 6. Este projeto favorece riscos e incertezas?
AÇÕES PROPOSTAS	Para os dados problemas/oportunidades e soluções, os seguintes questionamentos devem ser feitos: 1. Foco: Qual é o foco recomendado na área de problema/oportunidade identificada? 2. Solução alvo: Qual é a direção recomendada da solução para a área foco? 3. Quais são os resultados, custos e benefícios esperados? 4. Quais são as ações de projeto necessárias para alcançá-los? 5. Riscos: Se circunstâncias internas ou externas à organização mudarem, sob quais condições a solução saberá reconsiderar as decisões propostas?

Tabela 5: MO-5 –Checklist para o documento de decisão de viabilidade. (Fonte: (SCHREIBER et al., 2002))

Este último modelo organizacional é o documento que definirá se o projeto de um sistema de conhecimento é viável e deve prosseguir, ou se ele não é aplicável à situação em questão. Neste processo de avaliação, três tipos de viabilidade são verificados:

- *Viabilidade de Negócios:* procura identificar se a solução proposta enquadra-se dentro das expectativas e da realidade da organização, e qual será o seu impacto.
- *Viabilidade Técnica:* procura avaliar quão complexa é a solução em termos de armazenamento de conhecimento, facilidade de uso, tempo e qualidade dos recursos necessários, etc.
- *Viabilidade do Projeto:* procura avaliar se o projeto será bem aceito pelas pessoas que lidarão direta e indiretamente com ele, se os recursos de tempo,

orçamento e equipamentos são suficientes e estarão disponíveis e principalmente se o projeto oferece riscos e incertezas.

Após a definição da viabilidade serão definidas as ações que serão executadas a seguir.

2.1.3 Modelo da Tarefa

Segundo Schreiber et al., 2002, uma tarefa pode ser definida como um conjunto de atividades que agregam valor à organização e que manipulam entradas e saídas de forma estruturada e controlada; ela consome recursos; provê conhecimentos e competências; está encarregada de fornecer critérios de qualidade e performance e é executada por agentes.

O objetivo do modelo de tarefa é detalhar o fluxo de tarefas em processos, facilitando assim a análise de cada atividade e a forma como elas se relacionam. Neste modelo as tarefas podem ser decompostas em sub-tarefas mais específicas.

O detalhamento de cada tarefa pode ser obtido aplicando os dois modelos de tarefa que ajudarão no refinamento e na definição do conhecimento envolvido na execução de cada tarefa.

O primeiro modelo da tarefa (MT-1) faz um aprofundamento nas tarefas do processo alvo, identificando as dependências, relações, agentes, recursos, etc.

Neste modelo, cada tarefa especificada no modelo organizacional será refinada através de especificações como dependência de fluxo de execução, objetos manipulados, o tempo de duração da tarefa e quais são os agentes responsáveis pela execução da tarefa, bem como quais são conhecimentos e competências necessários para a execução da tarefa. Este modelo ainda leva em consideração os recursos necessários para a tarefa e a performance de sua execução.

A tabela 6 do MT-1 mostra em detalhes os itens a serem observados no refinamento de cada tarefa.

MODELO DA TAREFA	Análise da tarefa – Tabela MT-1
TAREFA	Identificador e nome da tarefa.
ORGANIZAÇÃO	Indica o processo de negócio do qual a tarefa faz parte, e onde na organização (estrutura, pessoas) ele é executado.
OBJETIVOS E VALORES	Descreve o objetivo da tarefa e o valor que sua execução adiciona ao processo ao qual a tarefa faz parte.
DEPENDÊNCIA E FLUXO	Tarefas de entrada: tarefas entregando entradas para esta tarefa. Tarefas de saída: tarefas que usam alguma(s) saída(s) desta tarefa. Pode-se usar um diagrama de fluxo de dados ou um diagrama de atividade para fazer esta descrição.
OBJETOS MANIPULADOS	*Objetos de entrada:* os objetos, incluindo itens de informação e conhecimento, que são entradas para tarefas. *Objetos de saída*: os objetos, incluindo itens de informação e conhecimento, que são entregues pelas tarefas como saídas. *Objetos internos:* objetos importantes, incluindo itens de informação e conhecimento, que são usados internamente numa tarefa mas não são entradas ou saídas para outras tarefas. Pode-se incluir um diagrama de classes para descrever os objetos de informação que são manipulados pela tarefa.
TEMPO E CONTROLE	Descrever freqüência e duração de cada tarefa. Descrever o controle do relacionamento com outras tarefas. Para isso pode-se usar um diagrama de estados ou de atividades. Descrever regras de controle: 1. *pré-condições* que precisam ser atendidas antes que a tarefa seja executada 2. *pós-condições* que precisam ser atendidas como resultado da execução da tarefa.
AGENTES	O grupo de membros e/ou sistemas de informação (conforme MO-2 e MO-3) que são responsáveis pela execução da tarefa.
CONHECIMENTO E COMPETÊNCIA	Competências necessárias para o sucesso da realização da tarefa. Para os itens de conhecimento envolvidos, existe uma tabela separada: a MT-2. Listar outras experiências relevantes e competências aqui. Indicar que elementos da tarefa são conhecimento intensivo. Note que tarefas podem entregar competências para a organização, e isto pode ser importante indicá-las aqui.
RECURSOS	Descrever e preferencialmente qualificar os vários recursos consumidos pela tarefa (tempo por pessoa, sistemas e equipamento, materiais, recursos financeiros). Esta descrição é normalmente um refinamento da descrição de recursos da tabela MO-2.
QUALIDADE E PERFORMANCE	Listar e qualificar as medidas de performance que serão usadas pela organização para determinar o sucesso da execução da tarefa.

Tabela 6: MT-1 –Refinamento das tarefas do processo alvo da solução. (Fonte: (SCHREIBER et al., 2002))

O segundo modelo da tarefa (MT-2) apresenta um detalhamento dos conhecimentos e competências utilizados para realização da tarefa. Trata-se também de um refinamento das informações inseridas no modelo da organização (MO-4) a respeito de conhecimentos utilizados. Neste modelo também será feita a análise dos “gargalos de conhecimento”.

MODELO DA TAREFA	**Item de Conhecimento –Tabela MT-2**	
NOME:	Item de conhecimento.	
POSSUÍDO POR:	Agente	
USADO EM:	Identificador e nome da tarefa	
DOMÍNIO:	Área do Domínio onde o conhecimento está inserido (campo especialista, disciplina, ramo da ciência ou da engenharia, comunidade profissional).	
Natureza do Conhecimento		**Gargalo / ser melhorado?**
Formal, rigoroso		
Empírico, quantitativo		
Heurístico, regras práticas		
Altamente especializado, domínio específico		
Baseado em experiência		
Baseado em ação		
Incompleto		
Incerto, pode estar incorreto		
Muda rapidamente		
Difícil de verificar		
Tácito, difícil de transferir		
Formas de Conhecimento		
Mente		
Papel		
Eletrônico		
Experiência pela prática		
Outros		
Disponibilidade do Conhecimento		
Limitações no tempo		
Limitações no espaço		
Limitações no acesso		
Limitações na qualidade		

Limitações na forma		

Tabela 7: MT-2 –Especificação do Conhecimento disponibilizado para a tarefa, e possíveis gargalos e áreas para melhoramento.(Fonte: (SCHREIBER et al., 2002))

2.1.4 Modelo do Agente

O objetivo do modelo de agente é compreender os papéis e competências que os diversos atores na organização desempenham para executar uma tarefa compartilhada. O modelo de agente representa todos os agentes participantes em um processo de resolução de problema, por meio da descrição de suas características (competências), sua autonomia para agir e suas restrições quanto à tarefa. Além disso, esse modelo permite definir as formas de comunicação entre os agentes participantes na realização da tarefa (Freitas, 2003).

No modelo de agentes (MA-1) serão levantadas informações como: a posição do agente dentro da hierarquia da organização, quais as tarefas (conforme MT-1) com as quais ele está envolvido, quais são os agentes com quem ele se comunica (conforme MT-2), quais são os conhecimentos, competências e responsabilidades que o agente possui ou que seriam necessárias.

O documento do modelo de agentes pode ser estruturado como é apresentado na tabela 8, proposta por Scheiber et al., 2002:

Modelo de Agentes	Agentes – Tabela MA-1
NOME	Nome do agente
ORGANIZAÇÃO	Indicar como o agente está posicionado na organização, deve-se usar as descrições do modelo da organização, incluindo o tipo (humano, sistema de informação), posição na organização, estrutura…
ENVOLVIDO EM	Lista de tarefas (conforme MT-1)
COMUNICA-SE COM	Lista de nomes de agentes
CONHECIMENTO	Lista de itens de conhecimento que o agente possui (conforme MT-2)
OUTRAS COMPETÊNCIAS	Lista de outras competências necessárias ou presentes no agente.
RESPONSABILIDADES E RESTRIÇÕES	Lista de responsabilidades que o agente tem na execução da tarefa, e das restrições a este respeito. Restrições podem referir-se a limitações de autoridade, mas também a leis internas e externas ou normas profissionais, etc.

Tabela 8: MT-2 –Especificação dos agentes envolvidos na tarefa.(Fonte: (SCHREIBER et al., 2002))

2.1.5 Consolidação dos modelos da organização, tarefa e agentes

As informações coletadas através de cada um dos modelos devem ser consolidadas num único documento, de maneira a permitir visualizar de forma mais crítica o ambiente organizacional e vislumbrar onde é possível efetuar mudanças e melhoramentos, bem como avaliar seus impactos. Segundo Schreiber et al., 2002, os principais itens para tomada de decisão sobre mudanças são:

- Mudanças organizacionais são necessárias? Se sim, quais são?
- Quais as medidas devem ser tomadas com relação a alguma tarefa especifica e quais os trabalhadores que diretamente ligados a ela? Em particular, quais melhorias são possíveis com relação ao uso e disponibilidade de conhecimento?
- As possíveis mudanças serão suportadas pelas pessoas envolvidas? Irão facilitar ao invés de complicar?
- Qual será a melhor direção a conduzir o projeto do sistema de conhecimento?

A tabela 9 apresenta a estrutura do checklist proposto por Schreiber et al., 2002, para a unificação dos modelos da organização, tarefa e agentes.

Modelos da Organização, Tarefa e Agentes	Checklist para o Documento de decisão de impactos e melhorias. – Tabela OTA-1
IMPACTOS E MUDANÇAS NA ORGANIZAÇÃO	Descrever que impactos e mudanças a solução de sistema de conhecimento traz com respeito a organização, pela comparação das diferenças entre o modelo da organização (MO-2) na situação atual, e como ela será vista no futuro. Isso tem que ser feito para todos os componentes (que variam) de uma maneira geral. 1. Estrutura 2. Processos 3. Recursos 4. Pessoas 5. Conhecimento 6. Cultura e poder
IMPACTOS E MUDANÇAS EM TAREFAS/AGENTES ESPECÍFICOS.	Descrever que impactos e mudanças à solução de sistema de conhecimento trás a respeito de agentes e tarefas individuais, pela comparação das diferenças entre os modelos de tarefas e agentes das tabelas, na situação atual, e como elas serão no futuro. È importante ver não somente o grupo de membros diretamente envolvidos na tarefa mas também outros atores e patrocinadores (tomadores de decisões, usuários e clientes). 1. Mudanças no layout da tarefa (fluxo, dependências, objetos manipulados, tempo e controle) 2. Mudanças em recursos necessários. 3. Critérios de performance e qualidade 4. Mudanças em grupos de funcionários e agentes envolvidos. 5. Mudanças de forma individual em posições, responsabilidades, autoridade, restrições na execução de tarefas. 6. Mudanças necessárias em conhecimentos e competências. 7. Mudanças na comunicação.
ATITUDE E COMPROMETIMENTO	Considerar como os atores e patrocinadores(de forma individual) envolvidos aceitarão as mudanças sugeridas, e se eles serão uma base para finalizar com sucesso estas mudanças.
AÇÕES PROPOSTAS	Esta é parte do documento de decisão de impactos e melhorias que está diretamente sujeito ao comprometimento gerencial e tomada de decisão. Isto pesa e integra os próximos resultados de análises até os passos concretos recomendados por ação: 1. *Melhorias*: Quais são as mudanças recomendadas, com respeito à organização, tarefas individuais, grupos de membros e sistemas? 2. *Medidas de acompanhamento:* Quais medidas de suporte podem ser utilizadas para facilitar estas mudanças (exemplo: treinamento) 3. Qual ação de promoção do projeto é recomendada com respeito a empregar a solução de sistema de conhecimento? 4. *Resultados esperados, custos e benefícios:* reconsiderar itens do documento de decisão de viabilidade antecipado. 5. Se circunstâncias dentro ou fora da organização mudarem, sob quais condições a solução saberá reconsiderar as decisões propostas?

Tabela 9: MT-2 –Checklist para o documento de decisão de impactos e melhorias.(Fonte: (SCHREIBER et al., 2002))

2.1.6 Modelo de Conhecimento

O modelo de conhecimento é o principal e mais complexo do conjunto de modelos da metodologia CommonKADS, é ele quem detalha o conhecimento do domínio e descreve a capacidade de um sistema de conhecimento em resolver problemas utilizando o conhecimento. O objetivo do modelo de conhecimento é explicar em detalhes os tipos e estruturas de conhecimento usadas na execução das tarefas (Schreiber et al.,2002).

O modelo de conhecimento é dividido em três níveis e, em cada nível, há um tipo particular de conhecimento. Os níveis do modelo de conhecimento são o conhecimento do domínio, conhecimento de inferência e conhecimento da tarefa.

Segundo Freitas, 2003, para construir um modelo de conhecimento é preciso passar pelas seguintes etapas:

- *Identificação do conhecimento*: Nesse estágio, são identificadas as fontes de informação e é construído um glossário preliminar de termos. A familiarização com o domínio e a identificação dos componentes potenciais do modelo constituem o suporte para a identificação do conhecimento;

- *Especificação do conhecimento:* Esse estágio tem como objetivo a especificação completa do modelo de conhecimento. As atividades desse estágio são: escolher a estrutura de inferência (*template*) com base em uma biblioteca disponível; construir o esquema inicial do domínio e completar a especificação do modelo de conhecimento;

- *Refinamento do conhecimento*: Esse estágio volta-se à validação do conhecimento, tanto quanto possível, e a inclusão de instâncias do conhecimento na base de conhecimento. A validação do modelo é realizada por meio do teste do modelo e da base de conhecimento, utilizando cenários (ou contextos) obtidos

na fase de identificação do conhecimento. As atividades relacionadas a esse estágio são: validar o modelo de conhecimento por intermédio de uma simulação e completar a base de conhecimento, incluindo instâncias do conhecimento necessárias à execução da tarefa em questão.

Segundo Schreiber et al.,2002, o modelo de conhecimento de uma aplicação fornece uma especificação dos dados e estrutura de conhecimento necessária para a aplicação. Este modelo é desenvolvido como parte do processo de analise.

O modelo de conhecimento está dividido em 3 categorias, conforme apresenta Schreiber et al.,2002:

- *Conhecimento do Domínio:* refere-se aos conhecimentos e tipos de informações específicos do domínio, dos quais se pretende utilizar na aplicação. Um exemplo prático de conhecimento de domínio é o de uma aplicação para diagnósticos médicos. Nesta aplicação deveria conter as definições das doenças mais relevantes, sintomas e testes, bem como a relação entre estes tipos.
- *Conhecimento de Inferência:* descreve os passos básicos de inferências que se deseja realizar usando o conhecimento do domínio. Inferências podem ser melhor entendidas se comparadas com blocos que compõem uma maquina de raciocínios. Dois exemplos de inferência, na aplicação de diagnósticos médicos, poderiam ser uma inferência “hipótese” que associa sintomas com uma possível doença, e uma inferência “verificação” que identifica testes que podem ser usados para determinar se certa doença é realmente o fator que causa os sintomas observados.
- *Conhecimento da tarefa:* descreve que objetivos uma aplicação deve alcançar, e como estes objetivos podem ser realizados através da decomposição de sub-tarefas e (finalmente) inferências. O aspecto “como” inclui a descrição dos comportamentos dinâmicos da tarefa, ou seja, seus controles internos. Um exemplo na aplicação de diagnósticos médicos seria ter “diagnóstico” como sendo uma tarefa de alto nível, e definindo que ela poder ser realizada através

de seqüências repetidas de invocações das inferências "hipótese" e "verificação".

O modelo de comunicação pode ser elaborado através de diagramas UML ou pela linguagem CML[3] que é a linguagem para modelagem de conhecimento utilizada no CommonKADS.

2.1.7 Modelo de Comunicação

O modelo de comunicação indica todas as transações (comunicações) ocorridas entre agentes, e mostra a comunicação requerida entre estes agentes durante a execução de um processo, podendo especificar, ainda, a troca de mensagens e quem toma a iniciativa em uma transação (Schreiber et al., 2000). Este modelo tem o objetivo de mostrar como o conhecimento produzido será transferido entre os agentes.

A representação do modelo de comunicação pode ser feita através da UML e tabelas, conforme são apresentadas a seguir:

Modelo de Comunicação	**Descrição de Transação - Tabela MC-1**
TRANSAÇÃO IDENTIFICADOR / NOME	Uma transação é definida para cada objeto de informação que é objeto de saída de alguma tarefa no modelo de tarefas ou no modelo de conhecimento (por exemplo, uma função de transferência), e que deve ser comunicada a outro agente para usá-la em suas tarefas. O nome deve refletir, de forma entendível ao usuário, o que é feito com o objeto de informação pela transação. Além disso o nome dá uma breve explanação do objetivo da transação.
OBJETO DE INFORMAÇÃO	Indica o principal objeto de informação, e quais são as duas tarefas entre as quais ele será transmitido.
AGENTES ENVOLVIDOS	Indica quais serão os agentes que irão enviar e receber o objeto de informação.
PLANO DE COMUNICAÇÃO	Indica o plano de comunicação ao qual a transação pertence.

[3] CML - Knowledge Modelling Language

RESTRIÇÕES	Especifica os requisitos e (pré)condições que precisam ser atendidas para que a transação seja executada. Às vezes, isto é útil também para estados de pós-condição que são assumidos válidos após a transação.
ESPECIFICAÇÃO DAS INFORMAÇÕES TRANSFERIDAS	Transações podem ter uma estrutura interna, em que ela consiste de várias mensagens de tipos diferentes, e/ou carregam objetos de informação como suporte adicional, assim como explanações ou itens de ajuda. Isto será detalhado na tabela CM-2. Até este ponto, somente uma referência básica precisa ser dada para depois fazer a especificação das informações transferidas.

Tabela 10: MC-1 – Tabela para Especificação das transações feitas no dialogo entre dois agentes no Modelo de Comunicação (Fonte: (SCHREIBER et al., 2002)).

Modelo de Comunicação	**Especificação das informações transferidas - Tabela MC-2**
TRANSAÇÃO	Informe o identificador e o nome da transação da qual esta especificação de informação transferida faz parte.
AGENTES ENVOLVIDOS	1. **"Sender":** agente que envia o(s) item(s) de informação 2. **"Receiver":** agente que receberá o(s) item(s) de informação
ITENS DE INFORMAÇÃO	Listar todos os itens de informação que serão transmitidos nesta transação. Isto inclui os principais objetos de informação cuja transferência é objetivo desta transação. Porém, ela pode conter outros itens de informação de suporte, que, por exemplo, fornecer ajuda ou explanação. Para cada item de informação, faça a seguinte descrição: 1. **Papel**: se ele é o objeto principal, ou um item de suporte. 2. **Forma:** a forma sintática em que é transmitido para outro agente, por exemplo, dados em string, um texto gravado, um certo tipo de diagrama. 3. **Meio:** o meio através do qual ele será carregado na interação a agente - agente, por exemplo, uma janela "pop-up", um menu de navegação e seleção, interface de linha de comando, intervenção humana, etc.
ESPECIFICAÇÕES DAS MENSAGENS	Descreva todas as mensagens que compõem a transação. Para cada mensagem individual descreva: 1. **Tipo de Comunicação:** o tipo de comunicação, descrevendo sua intenção. 2. **Conteúdo:** uma declaração ou proposição contida na mensagem. 3. **Referência:** em certos casos, isto pode ser útil para adicionar uma referência para, por exemplo, qual domínio de conhecimento ou capacidade de agente é requerida para estar disponível para enviar ou processar

	a mensagem.
CONTROLE SOBRE AS MENSAGENS	Dar, se necessário, uma especificação de controle sobre as mensagens dentro da transação. Isto pode ser feito em formato de pseudo-código ou num diagrama de estados, parecido com como o controle sobre as transações dentro do plano de comunicação é especificado.

Tabela 11: Tabela 9: MT-2 – Tabela para a especificação de mensagens e itens de informação que fazem parte de uma transação do Modelo de Comunicação. (Fonte: (SCHREIBER et al., 2002))

2.2 Ferramenta para Documentação: CommonKADS Editor

Atualmente, segundo pesquisa realizada nesse trabalho, a ferramenta mais completa que se encontra disponível, com objetivo específico de armazenar toda a documentação do CommonKADS, é o *CommonKADS Editor*[4].

O CommonKADS Editor foi criado no ano de 2003 e tem por objetivo principal o suporte ao armazenamento das informações de projetos de Engenharia e Gestão do Conhecimento com o uso da metodologia CommonKADS.

[4] Para referência sobre download do CommonKADS Editor consulte as referências bibliográficas deste trabalho.

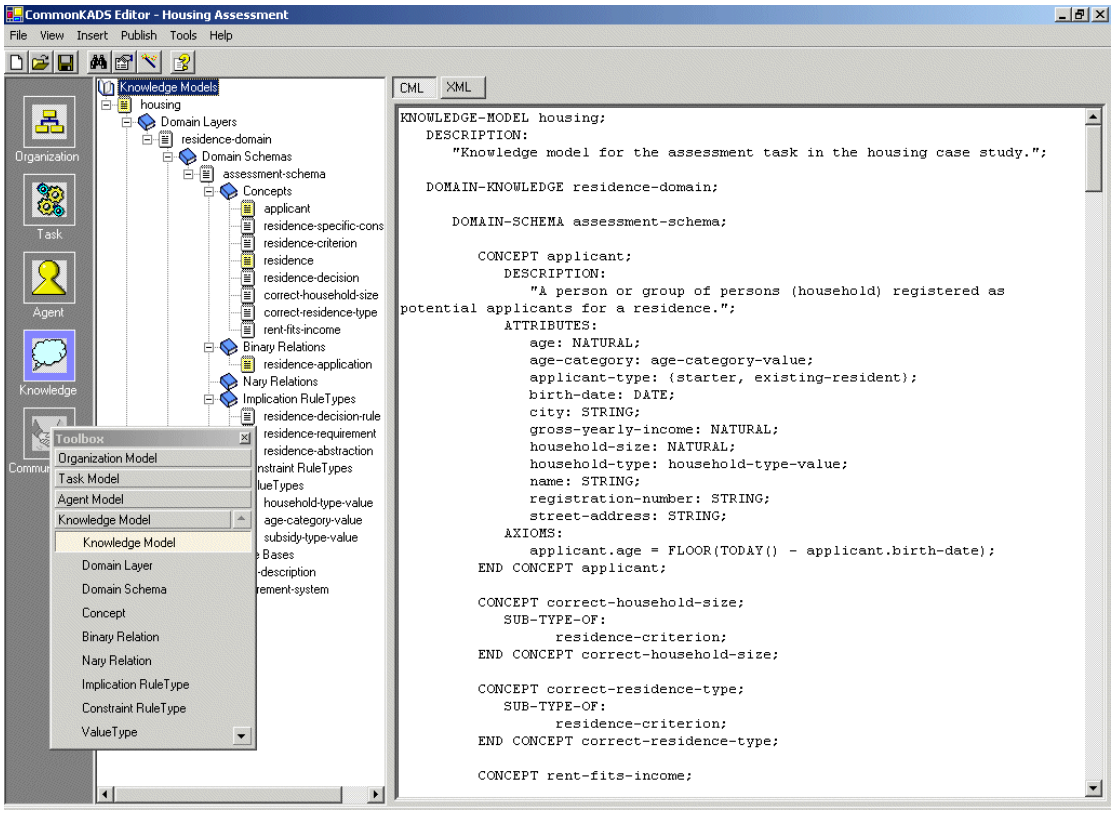

Figura 4: Interface do CommonKADS Editor

A ferramenta oferece suporte para os modelos do CommonKADS sejam usados segundo dois pontos de vista: o cenário atual, em que se encontra a organização, e um cenário futuro, considerando-se as possíveis mudanças que se deseja e que poderão ocorrer dentro desta organização

Todos os *templates* disponíveis na metodologia CommonKADS também estão disponíveis para uso dentro do CommonKADS Editor, sendo que a adição de informações dentro dos modelos resulta na geração automática de código na linguagem CML.

De todos os modelos do CommonKADS, o CommonKADS Editor só não oferece suporte para o Modelo de Projeto. Porém o nível de detalhamento oferecido para o armazenamento das informações dos outros modelos vai além do que é apresentado nos exemplos de tabelas disponíveis na literatura.

O fato de o CommonKADS Editor tratar cada projeto, nele criado, de forma isolada, armazenando-o num arquivo (de extensão .kad), ao invés de um banco de dados único,

limita-o com relação à possibilidade de reaproveitar experiências e conhecimentos obtidos em projetos anteriores.

3 Sistema Proposto

3.1 Introdução

O processo de desenvolvimento de softwares tem como etapa inicial a fase de análise de requisitos e projeto. Esta etapa é indispensável, pois está diretamente ligada relacionada à qualidade final produto que está sendo criado. A análise e projeto compõem duas das etapas do ciclo de vida de desenvolvimento de software.

Este capítulo apresentará a ferramenta proposta neste trabalho mostrando seus casos de uso e sua estrutura.

3.2 Posicionamento: Definição da Aplicação

Para se estabelecer as delimitações de um projeto é necessário que se tenha todo o conhecimento sobre o domínio do problema em questão e quais são os requisitos definidos para aplicação. Sendo assim os tópicos, a seguir, apresentarão as informações coletadas sobre o domínio do problema.

3.2.1 Descrição do Domínio do Problema

Conforme descrito no Capítulo 2, a metodologia CommonKADS possui um conjunto de modelos que abrange todas as etapas para o projeto de um sistema de Conhecimento. Parte destes modelos são tabelas que auxiliam o Engenheiro do Conhecimento na elaboração da análise e projeto.

Dessa forma, a aplicação da metodologia CommonKADS em projetos de Engenharia do Conhecimento resulta em uma série de planilhas com análises descritivas. Normalmente estas planilhas são criadas em algum tipo de editor, como o MS Excel, o que gera alguns problemas e desconfortos:

- Aumento da dificuldade em visualizar de forma sistêmica as informações como um todo.
- Retrabalho na digitação de informações, já que algumas planilhas são refinamento de informações e utilizam dados que constam em outras planilhas.
- No caso de projetos com grande número de processos, tarefas, agentes e conhecimentos envolvidos, o número e tamanho das planilhas resultantes também será grande, com isso, a dificuldade em encontrar informações também aumentará.
- As informações coletadas através da aplicação do CommonKADS também compõe um tipo de conhecimento que, estando somente em planilhas, não poderá ser reaproveitado. Dessa forma, experiências em projetos anteriores ficam perdidas.

Diante destes problemas, faz-se necessária uma ferramenta que possua a capacidade de armazenar as informações coletadas, durante a aplicação do CommonKADS, de forma a poder recuperá-las e analisá-las por vários prismas, a fim de obter-se maior agilidade na elaboração dos documentos e definição de soluções que possam ser viáveis e, cujo impacto, possa ser previsto de forma completa.

Com a finalidade de facilitar o entendimento do domínio do problema e evitar erro de interpretação devido a palavras desconhecidas ou ambíguas é extremamente importante a criação de um glossário com a definição dos termos e elementos que compõem o domínio do problema:

- **Organização:** instituição na qual está sendo realizada a análise e projeto do sistema de conhecimento.
- **Processos:** conjunto de tarefas que são realizadas dentro de uma organização.
- **Tarefas:** trabalho realizado por um agente, num determinado intervalo de tempo, visando um objetivo.
- **Agente:** pode ser um ser humano (uma pessoa) ou um software (recurso).

- **Pessoa:** entidade que possui algum tipo de vinculo com a organização: um tomador de decisões, funcionário, um fornecedor, um cliente ou qualquer outra pessoa que esteja de alguma forma ligada à organização e ao projeto de Engenharia do Conhecimento em andamento.
- **Recursos:** elementos utilizados nos processos de negócios: softwares e outros recursos computacionais equipamentos, materiais, tecnologias, patentes e direitos.

3.2.2 Sistema de apoio à aplicação da Metodologia CommonKADS

A proposta deste trabalho é a criação de uma aplicação que tenha a capacidade de armazenar todas as informações coletadas durante a aplicação da metodologia CommonKADS em projetos de Engenharia do Conhecimento.

De acordo com a literatura apresentada no Capítulo 2, e o detalhamento do domínio do problema, um sistema de apoio à aplicação da metodologia CommonKADS deve ser capaz de armazenar vários projetos de Engenharia do Conhecimento fornecendo meios para que o usuário possa cadastrar informações sobre a organização e suas características, seus processos, tarefas, agentes, conhecimentos e recursos envolvidos em cada tarefa, bem como criar uma ligação entre todos estes itens, de forma que possa existir uma ordem lógica na inclusão de informações. Isso fará com que o refinamento das informações ocorra de maneira natural, como prevê a metodologia CommonKADS pela proposição da ordem em que as planilhas do modelo devem ser preenchidas.

Sendo assim, o sistema deve permitir que o usuário, ao iniciar a entrada de informações num modelo possa ter acesso às informações que foram cadastradas no modelo anterior, de forma a reaproveitar e refinar as informações que já foram definidas e, caso as informações não tenham sido cadastradas no modelo anterior, o sistema deve permitir a inclusão dessa informação no modelo atual, de forma a possibilitar que o Engenheiro de Conhecimento inicie o projeto por qualquer modelo.

O sistema deve atender todos os requisitos levantados durante a fase de análise, conforme será apresentado adiante.

3.3 Especificação de requisitos

A especificação de requisitos dá-se através da verificação de quais são as necessidades funcionais do sistema, para que este solucione o problema em questão, além dos requisitos de utilização do mesmo, pelo usuário. Segundo Larman, 2002, uma especificação de requisitos correta e abrangente é a essência de um projeto de sucesso. E esta especificação deve ser composta de cinco artefatos: visão geral do sistema, clientes, objetivos, funções do sistema e atributos do sistema.

1) **Visão geral do Sistema:** trata-se de uma ferramenta de apoio à documentação de análises feitas em projetos de Engenharia e Gestão do Conhecimento.
2) **Clientes:** são os usuários que irão operar o sistema. No sistema proposto os usuários são Engenheiros do Conhecimento que efetuarão a análise e documentação do projeto de sistema de conhecimento.
3) **Objetivos:** O sistema proposto deverá permitir que o Engenheiro do Conhecimento possa armazenar e consultar as informações coletadas durante as etapas de análise realizadas com base na aplicação da metodologia CommonKADS num projeto de Engenharia do Conhecimento.
4) **Funções do Sistema:** Para atender os objetivos do sistema as seguintes funções estarão disponíveis:
    - Cadastro de projetos de sistemas de conhecimento;
    - Cadastro de informações do Modelo Organizacional (tabela MO-1): cadastro de informações sobre a Organização, seus problemas e oportunidades;

- Cadastro de Aspectos Variáveis da Organização (tabela MO-2): estrutura da organização, cadastro de processos, pessoas e recursos;
- Cadastro de desdobramento dos processos (tabela MO-3): cadastros de tarefas e conhecimentos;
- Cadastro de conhecimentos utilizados (tabela MO-4): relacionamento entre conhecimentos e agentes;
- Cadastro de Documento de Decisão de Viabilidade (tabela MO-5): cadastros de viabilidades: de negócios, técnica, de projeto; e ações propostas;
- Cadastro de análises de tarefas (tabela MT-1): cadastro de tarefas refinadas e relacionamento das mesmas com agentes, conhecimentos e recursos;
- Cadastro de Itens de Conhecimento (tabela MT-2): cadastro do conhecimento refinado e relacionamento do mesmo com agentes e tarefas;
- Cadastro de Agentes (tabela MA-1): refinamento das informações dos agentes relacionando o mesmo com conhecimentos e outros agentes com quem ele comunica-se.
- Cadastro de Documento de decisão de impacto e melhoria (tabela OTA): cadastro dos impactos e ações de melhoria.
- Visualização de problemas e soluções encontradas. Registradas de forma que o Engenheiro do conhecimento tenha condições de decidir qual é a solução que apresenta maior viabilidade.

5) Atributos do Sistema:

Além das funcionalidades ligadas à gestão das informações geradas durante a aplicação da metodologia CommonKADS, o sistema proposto deve ter atributos de usabilidade que facilitem o trabalho do engenheiro do conhecimento. Prevêem-se os seguintes atributos:

- Interface gráfica com o usuário;
- Boa usabilidade e layout que seja familiar aos modelos propostos no CommonKADS.
- Funcionamento independente de plataforma de software e hardware.
- Utilizável com qualquer sistema gerenciador de banco de dados.

3.3.1 Definição dos Usuários do Sistema

De acordo com as delimitações do escopo do sistema, estabelecidos no início deste trabalho, o sistema possuirá um único perfil de usuário que será o Engenheiro do Conhecimento. Este usuário terá acesso a todas as opções do sistema. Sendo assim não será implementado nenhum tipo de controle relacionado a usuários, como login, cadastro de usuários, controle de acessos, etc.

Dessa forma, em todos os diagramas em que haja iteração do usuário com o sistema, ele será representado da seguinte forma:

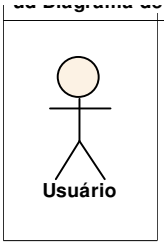

Figura 5: Usuário do Sistema

3.3.2 Áreas do Sistema e Casos de Uso

Os casos de uso são uma maneira, em alto nível, de se descrever quais são as necessidades do usuário com relação ao sistema, e qual será sua forma de iteração com mesmo. Será no documento de casos de uso que serão descritos os fluxos de operações principais (de sucesso) do caso de uso e as extensões (fluxos alternativos) em caso de exceções.

Visando agrupar os casos de uso de maneira lógica, foram definidas áreas de acesso do usuário ao sistema. Os casos de uso contidos em cada área corresponderão às atividades executadas em cada um dos modelos do CommonKADS,

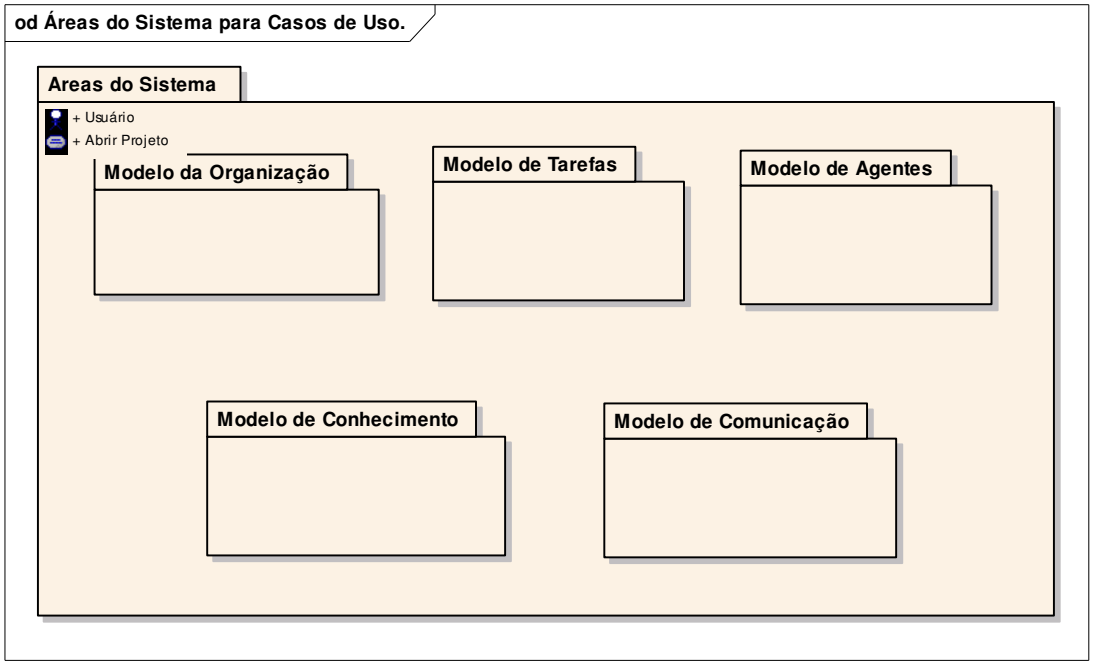

Figura 6: Áreas de acesso do usuário

A seguir serão descritos cada um dos casos de uso essenciais do sistema pertencente a cada uma das áreas definidas acima.

Tabela 12: Caso de Uso: Abrir Projeto

Caso de Uso: Abrir Projeto	
Breve Descrição:	O usuário ao abrir o sistema deverá abrir um projeto no qual trabalhará adicionando as informações da análise.
Ator:	Usuário do sistema.
Pré-condições:	O projeto deverá estar cadastrado.
Fluxo Principal:	1. O usuário informa qual é o projeto no qual deseja trabalhar 2. O sistema abrirá o projeto apresentando quais as informações relativas à metodologia CommonKADS que já foram cadastradas.
Fluxos Alternativos e Exceções:	1a. O projeto ainda não está cadastrado no banco de dados 1. O usuário informa qual o nome do projeto 2. O sistema cria o novo projeto e abrirá o mesmo para o usuário.
Pós-condições:	O sistema estará pronto para receber informações pertinentes ao projeto aberto no momento.
Requisitos não funcionais:	As outras funções do sistema não poderão ser acessadas se não houver um projeto aberto
Detalhes de interface:	Ao abrir o sistema, deverá ser exibida uma tela com a lista de projetos cadastrados para que o usuário possa selecionar e abrir um projeto, ou cadastrar um novo projeto.

3.3.2.1 Modelo da Organização

O Modelo da Organização é a área onde o usuário fará o cadastro de todas as informações do Modelo da Organização proposto pelo CommonKADS. Nesta área as cinco principais tarefas a serem realizadas são: cadastrar Organização, cadastrar aspectos variáveis da organização, cadastrar desdobramento de processos, cadastrar conhecimentos utilizados, cadastrar documento de decisão e viabilidade. Conforme é apresentado a seguir:

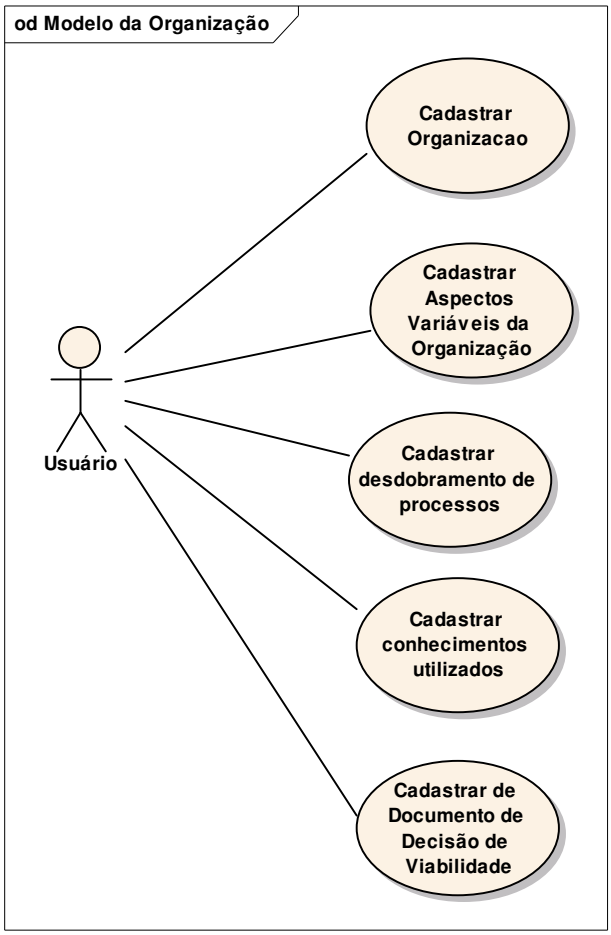

Figura 7: Diagrama de Casos de Uso da área: Modelo da Organização

Os casos de uso apresentados na área Modelo da Organização são descritos em mais detalhes logo a seguir:

Tabela 13: Caso de Uso: Cadastrar Organização

Caso de Uso: Cadastrar Organização	
Breve Descrição:	Cadastro de informações referentes à tabela 1 do Modelo Organizacional (Tabela MO-1)
Ator:	Usuário do Sistema
Pré-condições:	Um projeto deve estar aberto. O usuário deve possuir as informações sobre a Organização
Fluxo Principal:	1. O usuário informa ao sistema que deseja iniciar o cadastramento das informações do modelo organizacional. 2. O sistema abrirá a tela do modelo organizacional. 3. O usuário criará uma pequena lista dos problemas e oportunidades, baseando-se em entrevistas, *brainstorm*s, conversas com gerentes, etc. 4. O usuário irá identificar de uma maneira concisa "características-chave" do contexto organizacional, assim como colocar as oportunidades e problemas listados numa perspectiva própria. Características importantes que serão consideradas: • Missão, visão e objetivos da organização; • Fatores externos à organização que podem está ligados com estratégias da organização; • Estratégias da organização; • Cadeias de valores que conduzem à empresa. 5. O usuário listará as possíveis soluções para os problemas percebidos, como sugerido nas entrevistas e discussões, e sobre as características do contexto organizacional. 6. O usuário clicará no botão "Salvar" para que o sistema armazene as informações no banco de dados.
Fluxos Alternativos e Exceções:	2a. Caso haja informações já cadastradas no banco de dados os sistema deverá apresentar estas informações.

Tabela 14: Caso de Uso: Cadastrar Aspectos Variáveis da Organização

Caso de Uso: Cadastrar Aspectos Variáveis da Organização	
Breve Descrição:	Cadastro de informações referentes à tabela 2 do Modelo Organizacional (Tabela MO-2), que descreve os aspectos variáveis da empresa: estrutura, processos, pessoas, recursos, conhecimento, cultura e poder.
Ator:	Usuário do Sistema
Fluxo Principal:	1. O usuário informa ao sistema que deseja iniciar o cadastramento das informações do modelo organizacional. 2. O usuário faz uma descrição da estrutura da organização, pra isso pode usar texto e criar uma representação gráfica da organização (ou parte dela) mostrando departamentos, grupos, unidades, seções, etc. 3. O usuário faz uma descrição textual ou esboço do layout (com a ajuda de UML) da manipulação dos processos de negócios. 4. O usuário indica que grupo de funcionários está envolvido, como atores ou patrocinadores, incluindo tomadores de decisões, fornecedores, usuários e beneficiários (clientes) do conhecimento. 5. O usuário descreve os recursos que são utilizados nos processos de negócios. 6. O usuário descreve superficialmente os conhecimentos envolvidos nos processos de negócios. 7. O usuário inclui informações sobre fatores de cultura e poder na organização. 8. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados.
Pós-condições:	A segunda tabela do modelo organizacional do CommonKADS (Tabela MO-2) terá seu conteúdo armazenado no banco de dados.
Considerações:	Diagramas UML e imagens não serão criados através do sistema. Serão criados em alguma ferramenta específica.

Tabela 15: Caso de Uso: Cadastrar desdobramento de processos

Caso de Uso: Cadastrar desdobramento de processos	
Breve Descrição:	Cadastro de informações referentes à tabela 3 do Modelo Organizacional (Tabela MO-3), que descreve o desdobramento dos processos definidos na tabela MO-2 em tarefas.
Ator:	Usuário do Sistema
Fluxo Principal:	1. O usuário informa ao sistema que deseja iniciar o cadastramento das informações do modelo organizacional. E executa as seguintes operações: 2. Seleciona qual será o processo que será desdobrado. 3. Define o nome da tarefa. 4. Seleciona o(s) agente(s) que executa(m) a tarefa. 5. Informa o local onde a tarefa é executada. 6. Define a lista de conhecimentos que é utilizada nesta tarefa 7. Determina se a tarefa é de conhecimento intensivo ou não 8. Determina o nível de significância da tarefa. 9. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados. 10. O usuário repete os passos 3 a 9 até que não haja mais tarefas para o processo selecionado.
Fluxos Alternativos e Exceções:	2a. Se não existir um processo cadastrado, o sistema permitirá ao usuário cadastrar e salvar o nome do processo. 2b. Se o processo já tiver tarefas cadastradas, o sistema apresentará as informações cadastradas. 3a. Se nenhum agente estiver cadastrado, o sistema permitirá ao usuário cadastrar e salvar o agente.
Pós-condições:	A terceira tabela do modelo organizacional do CommonKADS (Tabela MO-3) terá seu conteúdo armazenado no banco de dados.

Tabela 16: Caso de Uso: Cadastrar conhecimentos utilizados

Caso de Uso: Cadastrar conhecimentos utilizados	
Breve Descrição:	Cadastro de informações referentes à tabela 4 do Modelo Organizacional (Tabela MO-4), que refinam os conhecimentos definidos na tabela MO-3
Ator:	Usuário do Sistema
Pré-Condições	Os nomes dos conhecimentos devem estar cadastrados no sistema.
Fluxo Principal:	1. O usuário informa ao sistema que deseja iniciar o cadastramento das informações do modelo organizacional. E executa as seguintes operações: 2. Seleciona o nome do conhecimento que deseja refinar. 3. Seleciona o nome do(s) agente(s) que possuem o conhecimento. 4. Informa onde (local da organização) o conhecimento é utilizado. 5. Informa e comenta se o conhecimento é utilizado no tempo, lugar, forma e qualidade corretos. 6. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados.
Fluxos Alternativos e Exceções:	2a. Se não existir um conhecimento cadastrado, o sistema permitirá ao usuário cadastrar e salvar o nome do conhecimento. 3a. Se nenhum agente estiver cadastrado, o sistema permitirá ao usuário cadastrar e salvar o agente.
Pós-condições:	A quarta tabela do modelo organizacional do CommonKADS (Tabela MO-4) terá seu conteúdo armazenado no banco de dados.

Tabela 17: Caso de Uso: Cadastrar de Documento de Decisão de Viabilidade

Caso de Uso: Cadastrar de Documento de Decisão de Viabilidade	
Breve Descrição:	Cadastro de informações referentes à tabela 5 do Modelo Organizacional (Tabela MO-5), que compõem o documento que leva a decisão de viabilidade ou não do projeto.
Ator:	Usuário do Sistema
Fluxo Principal:	1. O usuário informa ao sistema que deseja iniciar o cadastramento das informações do modelo organizacional. E executa as seguintes operações: 2. Cadastra informações referentes à viabilidade de negócios. 3. Cadastra informações referentes à viabilidade técnica. 4. Cadastra informações referentes à viabilidade de projeto. 5. Define quais são as ações propostas para os problemas, oportunidades e soluções definidas. 6. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados.
Pós-condições:	A quinta tabela do modelo organizacional do CommonKADS (Tabela MO-5) terá seu conteúdo armazenado no banco de dados.
Considerações:	A quinta tabela (MO-5) trata-se de um checklist que pode ser definido de forma que o usuário possa responder perguntas que componham cada parte do documento de viabilidade.

3.3.2.2 Modelo da Tarefa

O Modelo da Tarefa é a área onde o usuário fará o cadastro de todas as informações do Modelo da Tarefa proposto pelo CommonKADS. Nesta área as duas principais tarefas a serem realizadas são: cadastrar análises de tarefas e cadastrar itens de conhecimento. Conforme é apresentado a seguir:

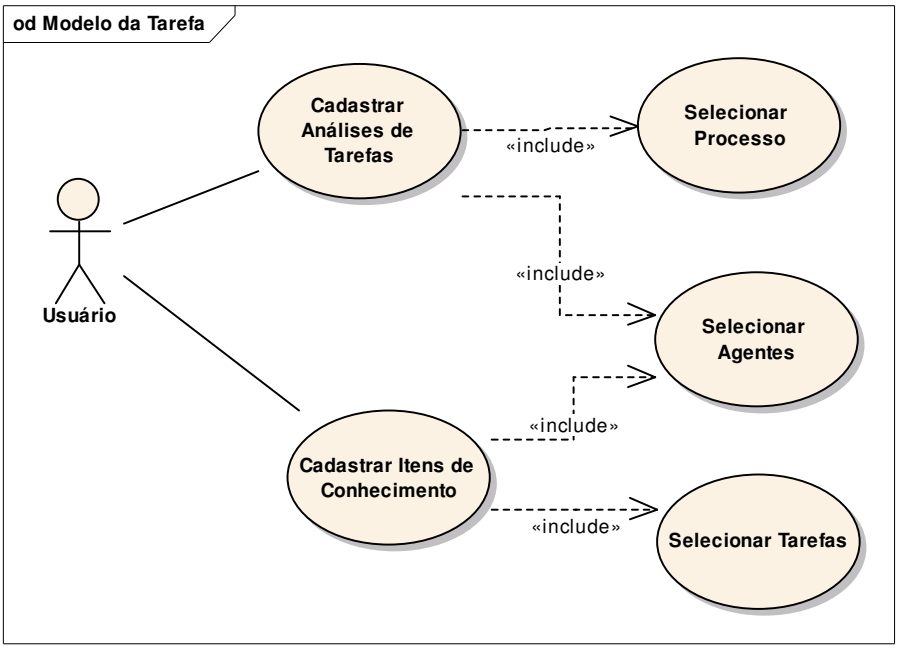

Figura 8: Diagrama de Casos de Uso da área: Modelo da Tarefa

Os casos de uso apresentados na área Modelo da Tarefa são descritos em mais detalhes logo a seguir:

Tabela 18: Caso de Uso: Cadastrar Análises de Tarefas

Caso de Uso: Cadastrar Análises de Tarefas	
Breve Descrição:	Cadastro de informações referentes à tabela 1 do Modelo da Tarefa (Tabela MT-1), para o detalhamento das tarefas que compõem os processos de negócios.
Ator:	Usuário do Sistema
Fluxo Principal:	1. O usuário informa ao sistema que deseja iniciar o cadastramento das informações do modelo da tarefa. E executa as seguintes operações: 2. Indica o processo de negócio do qual a tarefa faz parte, e onde na organização (estrutura, pessoas) ele é executado. 3. Descreve o objetivo da tarefa e o valor que sua execução adiciona ao processo ao qual ela faz parte. 4. Determina o fluxo de dependência entre a tarefa e outras tarefas, informando tarefas de entrada e saída. 5. Lista quais são os objetos manipulados nesta tarefa: objetos de entrada, saída e internos. 6. Descrever freqüência e duração de cada tarefa. 7. Seleciona o nome do(s) agente(s) que são responsáveis pela execução da tarefa. 8. Relaciona as competências necessárias para o sucesso da realização da tarefa. (Para os itens de conhecimento envolvidos, existe uma tabela separada: a MT-2). 9. Lista outras experiências relevantes e competências. Indicar que elementos da tarefa são conhecimento intensivo. 10. Descreve e preferencialmente qualifica os vários recursos consumidos pela tarefa (tempo por pessoa, sistemas e equipamento, materiais, recursos financeiros). Esta descrição é normalmente um refinamento da descrição de recursos da tabela MO-2. 11. Lista e qualifica as medidas de performance que serão usadas pela organização para determinar o sucesso da execução da tarefa 12. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados.
Fluxos Alternativos e Exceções:	7a. Se nenhum agente estiver cadastrado, o sistema permitirá ao usuário cadastrar e salvar o agente.
Pós-condições:	A primeira tabela do modelo da tarefa do CommonKADS (Tabela MT-1) terá seu conteúdo armazenado no banco de dados.

Tabela 19: Caso de Uso: Cadastrar Itens de Conhecimento

Caso de Uso: Cadastrar Itens de Conhecimento	
Breve Descrição:	Cadastro de informações referentes à tabela 2 do Modelo da Tarefa (Tabela MT-2), para o detalhamento dos itens de conhecimentos utilizados nas tarefas que compõem os processos de negócios.
Ator:	Usuário do Sistema
Pré-condições:	As tarefas e agentes deverão estar cadastrados no sistema.
Fluxo Principal:	1. O usuário informa ao sistema que deseja iniciar o cadastramento das informações do modelo da tarefa. E executa as seguintes operações: 2. Informa quem é o agente possuidor do item de conhecimento. 3. Informa qual é a tarefa no qual este conhecimento é utilizado. 4. Informa a área do Domínio onde o conhecimento está inserido (campo especialista, disciplina, ramo da ciência ou da engenharia, comunidade profissional). 5. Responde a um checklist determinando a natureza do conhecimento, se ele possui gargalos e de pode ser melhorado. 6. Informa qual onde o conhecimento está armazenado: papel, mente, eletrônico, etc. 7. Informa qual é a disponibilidade do conhecimento. 8. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados
Fluxos Alternativos e Exceções:	2a. Se o agente não estiver cadastrado, o sistema permitirá ao usuário cadastrar e salvar o agente. 3a. Se a tarefa não estiver cadastrada, o sistema permitirá ao usuário cadastrar e salvar a tarefa.
Pós-condições:	A segunda tabela do modelo da tarefa do CommonKADS (Tabela MT-2) terá seu conteúdo armazenado no banco de dados.

3.3.2.3 Modelo de Agentes

O Modelo de Agentes é a área onde o usuário fará o cadastro refinado das informações referentes aos agentes do Modelo de Agentes proposto pelo CommonKADS. Nesta área há duas tarefas a serem realizadas: cadastrar agente e cadastrar Documento de Decisão de Impactos e Melhorias. Conforme é apresentado a seguir:

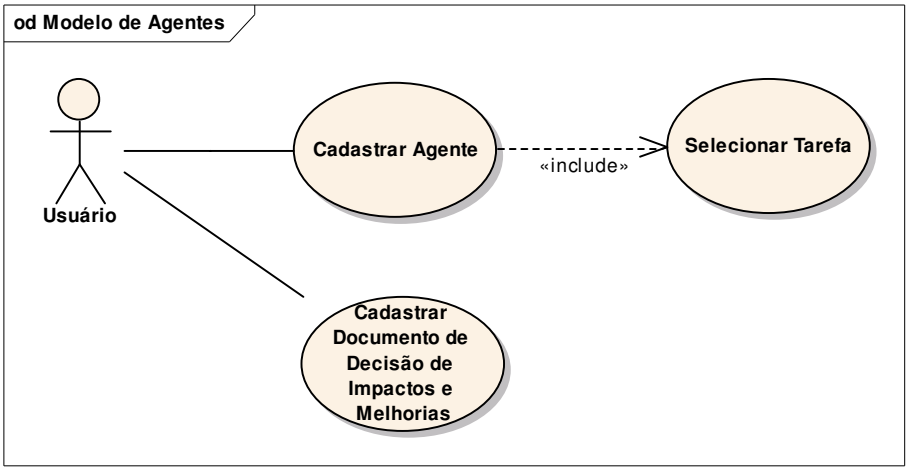

Figura 9: Diagrama de Casos de Uso da área: Modelo de Agentes

Os casos de uso apresentados na área Modelo de Agentes são descritos em mais detalhes logo a seguir:

Tabela 20: Caso de Uso: Cadastrar Agente

Caso de Uso: Cadastrar Agente	
Breve Descrição:	Cadastro de informações referentes à tabela 1 do Modelo de Agentes (Tabela MA-1), para o detalhamento de quem são os agentes da organização, quais suas tarefas e conhecimentos.
Ator:	Usuário do Sistema
Fluxo Principal:	1. O usuário informa ao sistema que deseja iniciar o cadastramento das informações do modelo da tarefa. E executa as seguintes operações: 2. Indica como o agente está posicionado na organização informando se ele é um agente humano ou um sistema de informação. 3. Relaciona quais são as tarefas com as quais o agente está envolvido. 4. Lista outras competências necessárias ou presentes no agente. 5. Lista as responsabilidades que o agente tem na execução da tarefa, e das restrições a este respeito. 6. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados
Fluxos Alternativos e Exceções:	3a. Se a tarefa não estiver cadastrada, o sistema permitirá ao usuário cadastrar e salvar a tarefa.
Pós-condições:	A tabela do modelo de agentes do CommonKADS (Tabela MA-1) terá seu conteúdo armazenado no banco de dados.

Tabela 21: Caso de Uso: Cadastrar Documento de Decisão de Impactos e Melhorias

Caso de Uso: Cadastrar Documento de Decisão de Impactos e Melhorias	
Breve Descrição:	Cadastro de informações referentes ao documento que relaciona os impactos e melhorias que a solução terá sobre a organização.
Ator:	Usuário do sistema
Fluxo Principal:	1. O usuário informa ao sistema que deseja iniciar o cadastramento das informações do documento de decisão de impactos e melhorias. E executa as seguintes operações: 2. Descreve que impactos e mudanças a solução de sistema de conhecimento traz com respeito à organização, pela comparação das diferenças entre o modelo da organização (MO-2) na situação atual, e como ela será vista no futuro. 3. Descreve que impactos e mudanças a solução de sistema de conhecimento trás a respeito de agentes e tarefas individuais, pela comparação das diferenças entre os modelos de tarefas e agentes das tabelas, na situação atual, e como elas serão no futuro. 4. Descreve como será a atitude e comprometimento das pessoas que estão envolvidas direta e indiretamente com a solução. 5. Descreve quais são as ações propostas na solução . 6. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados
Pós-condições:	Os dados da tabela do Documento de Decisão de Impactos e Melhorias terá seu conteúdo armazenado no banco de dados.

3.3.2.4 Modelo de Conhecimento

O Modelo de Conhecimento é a área onde o usuário fará o cadastro de toda a especificação da estrutura conceitual do conhecimento conforme determina o CommonKADS. Nesta área as tarefas a serem realizadas são: cadastrar de Conhecimento de Domínio, cadastrar conhecimento de inferência e cadastrar conhecimento de tarefa. Conforme é apresentado a seguir:

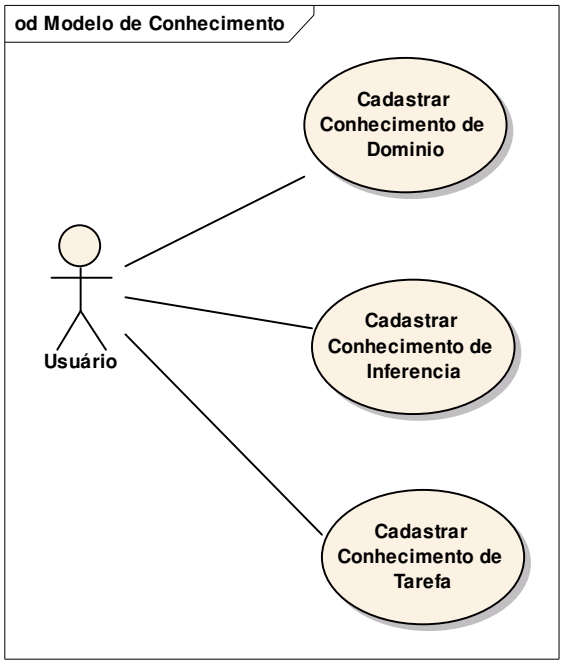

Figura 10: Diagrama de Casos de Uso da área: Modelo de Conhecimento

Os casos de uso apresentados na área “Modelo de Conhecimento” são descritos em mais detalhes logo a seguir:

Tabela 22: Caso de Uso: Cadastrar Conhecimento de Domínio

Caso de Uso: Cadastrar Conhecimento de Domínio	
Breve Descrição:	Cadastro da especificação CML correspondente ao conhecimento de domínio do Modelo de Conhecimento.
Ator:	Usuário do Sistema
Fluxo Principal:	1. O usuário informa ao sistema de que deseja cadastrar as informações referentes ao Modelo de Conhecimento. E executa as seguintes operações: 2. Descreve em CML o Conhecimento de Domínio definindo os Conceitos, Relações e Tipos de Regras existentes no Domínio do projeto em questão. 3. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados

Tabela 23: Caso de Uso: Cadastrar Conhecimento de Inferência

Caso de Uso: Cadastrar Conhecimento de Inferência	
Breve Descrição:	Cadastro da especificação CML correspondente ao conhecimento de inferência do Modelo de Conhecimento.
Ator:	Usuário do Sistema
Fluxo Principal:	1. O usuário informa ao sistema de que deseja cadastrar as informações referentes ao Modelo de Conhecimento. E executa as seguintes operações: 2. Descreve em CML o Conhecimento de Inferência definindo as inferências existentes no Domínio do projeto e suas dependências e funções de transferência. 3. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados

Tabela 24: Caso de Uso: Cadastrar Conhecimento de Tarefa

Caso de Uso: Cadastrar Conhecimento de Tarefa	
Breve Descrição:	Cadastro da especificação CML correspondente ao conhecimento de tarefa do Modelo de Conhecimento.
Ator:	Usuário do Sistema
Fluxo Principal:	1. O usuário informa ao sistema de que deseja cadastrar as informações referentes ao Modelo de Conhecimento. E executa as seguintes operações: 2. Descreve em CML o conhecimento de tarefa definindo os métodos de tarefas, para as tarefas de conhecimento intensivo. Esta definição pode basear-se num dos *templates* disponibilizados no CommonKADS. 3. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados

3.3.2.5 Modelo de Comunicação

O Modelo de Comunicação é a área onde o usuário fará o cadastro de todas as especificações da estrutura de comunicação entre os agentes que estão executando tarefas. Nesta área, as tarefas a serem realizadas são: cadastrar transação e cadastrar mensagem. Conforme é apresentado a seguir:

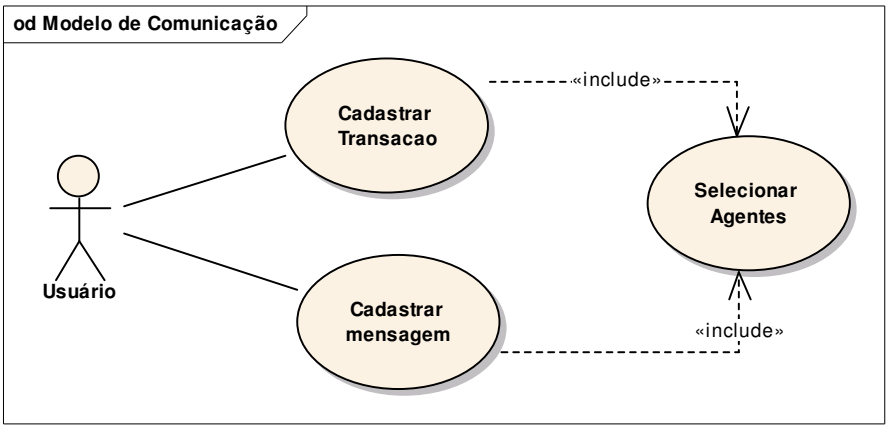

Figura 11: Diagrama de Casos de Uso da área: Modelo de Conhecimento

Os casos de uso apresentados na área Modelo de Conhecimento são descritos em mais detalhes logo a seguir:

Tabela 25: Caso de Uso: Cadastrar Transação

Caso de Uso: Cadastrar Transação	
Breve Descrição:	Cadastro da transação de comunicação entre dois agentes, correspondente à tabela 1 do Modelo de Comunicação (MC-1).
Ator:	Usuário do sistema.
Pré-condições:	Agentes devem estar cadastrados no sistema

Tabela 26: Caso de Uso: Cadastrar Mensagem

Caso de Uso: Cadastrar Mensagem	
Breve Descrição:	Cadastro das mensagens e itens de informação que compõem uma transação. Corresponde a tabela 2 do Modelo de Comunicação (MC-2).
Ator:	Usuário do Sistema
Pré-condições:	1. A transação deve estar cadastrada no sistema. 2. Os agentes devem estar cadastrados no sistema.
Fluxo Principal:	1. O usuário informa ao sistema que deseja cadastrar informações no Modelo de Comunicação. E executa as seguintes operações: 2. Informa o identificador e o nome da transação da qual esta especificação de informação transferida faz parte. 3. Determina quem são os agentes envolvidos. 4. Lista todos os itens de informação que serão transmitidos nesta transação. Isto inclui os principais objetos de informação cuja transferência é objetivo desta transação 5. Descreve as especificações da mensagem 6. Faz uma especificação de controle sobre as mensagens dentro da transação. Isto pode ser feito em formato de pseudo-código ou num diagrama de estados, parecido com como o controle sobre as transações dentro do plano de comunicação é especificado. 7. O usuário clica no botão “Salvar” para que o sistema armazene as informações no banco de dados
Considerações:	Para cada item de informação, é feita a seguinte descrição: 1. **Papel**: se ele é o objeto principal, ou um item de suporte. 2. **Forma:** a forma sintática em que é transmitido para outro agente, por exemplo, dados em string, um texto gravado, um certo tipo de diagrama. 3. **Meio:** o meio através do qual ele será carregado na interação a agente - agente, por exemplo, uma janela “pop-up”, um menu de navegação e seleção, interface de linha de comando, intervenção humana, etc.

	Para cada mensagem individual é descrito: 1. **Tipo de Comunicação:** o tipo de comunicação, descrevendo sua intenção. 2. **Conteúdo:** uma declaração ou proposição contida na mensagem. 3. **Referência:** em certos casos, isto pode ser útil para adicionar uma referência para, por exemplo, qual domínio de conhecimento ou capacidade de agente é requerida para estar disponível para enviar ou processar a mensagem.

Após definidos todos os casos de uso essenciais do sistema, temos o diagrama completo dos pacotes que agrupam os casos de uso em áreas. Conforme mostra a figura a seguir:

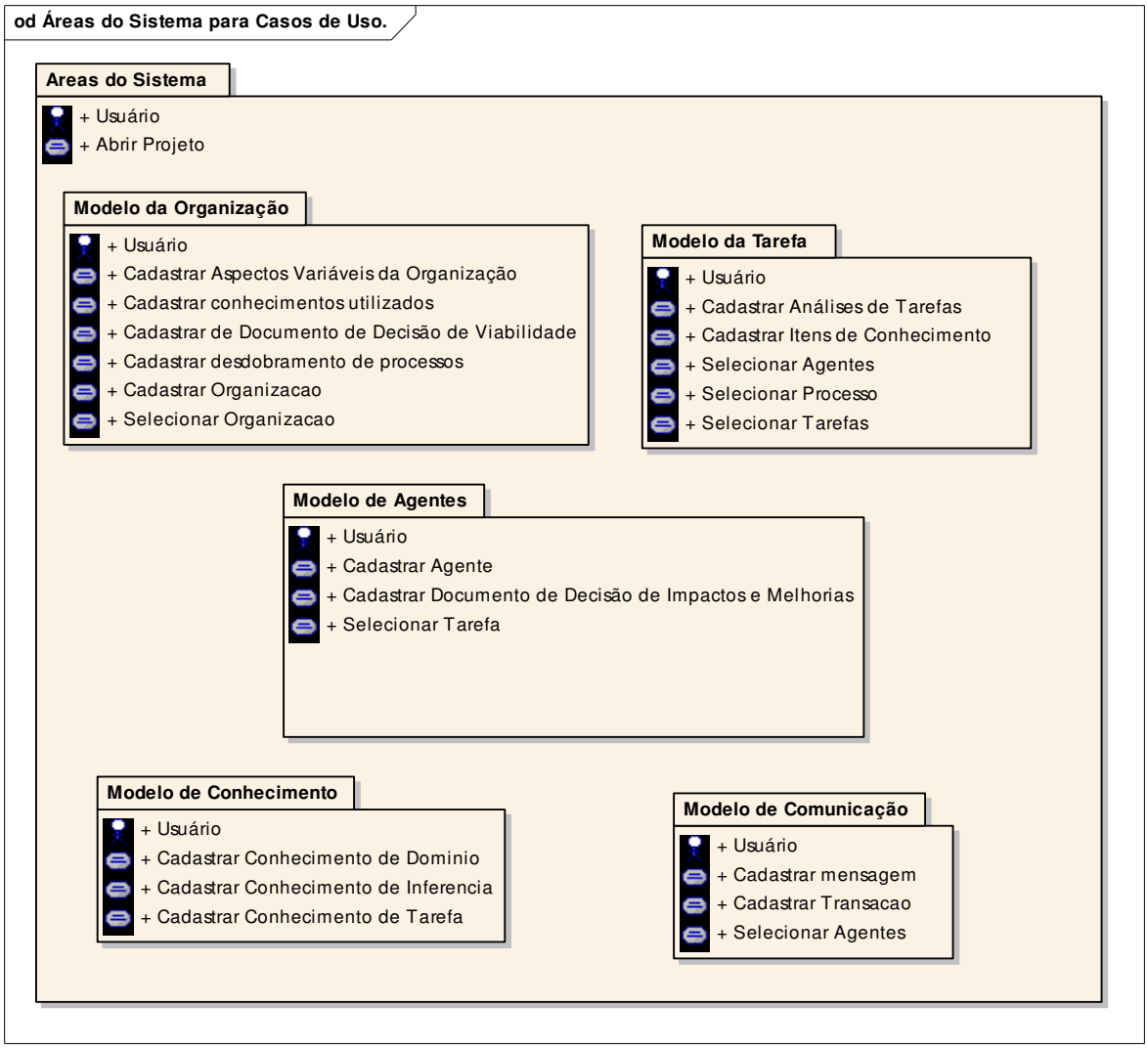

Tabela 27: Diagrama de Casos de Uso agrupados em áreas.

3.4 Modelo Conceitual

Através da descrição feita nos casos de uso e das informações contidas nas tabelas dos modelos do CommonKADS foi possível definir o Modelo Conceitual do domínio do problema.

Este modelo vista abstrair os elementos do mundo real e não componentes de software, com o objetivo de facilitar o entendimento de como estes elementos estão

associados dentro do domínio do problema. A figura a seguir ilustra o modelo conceitual do sistema:

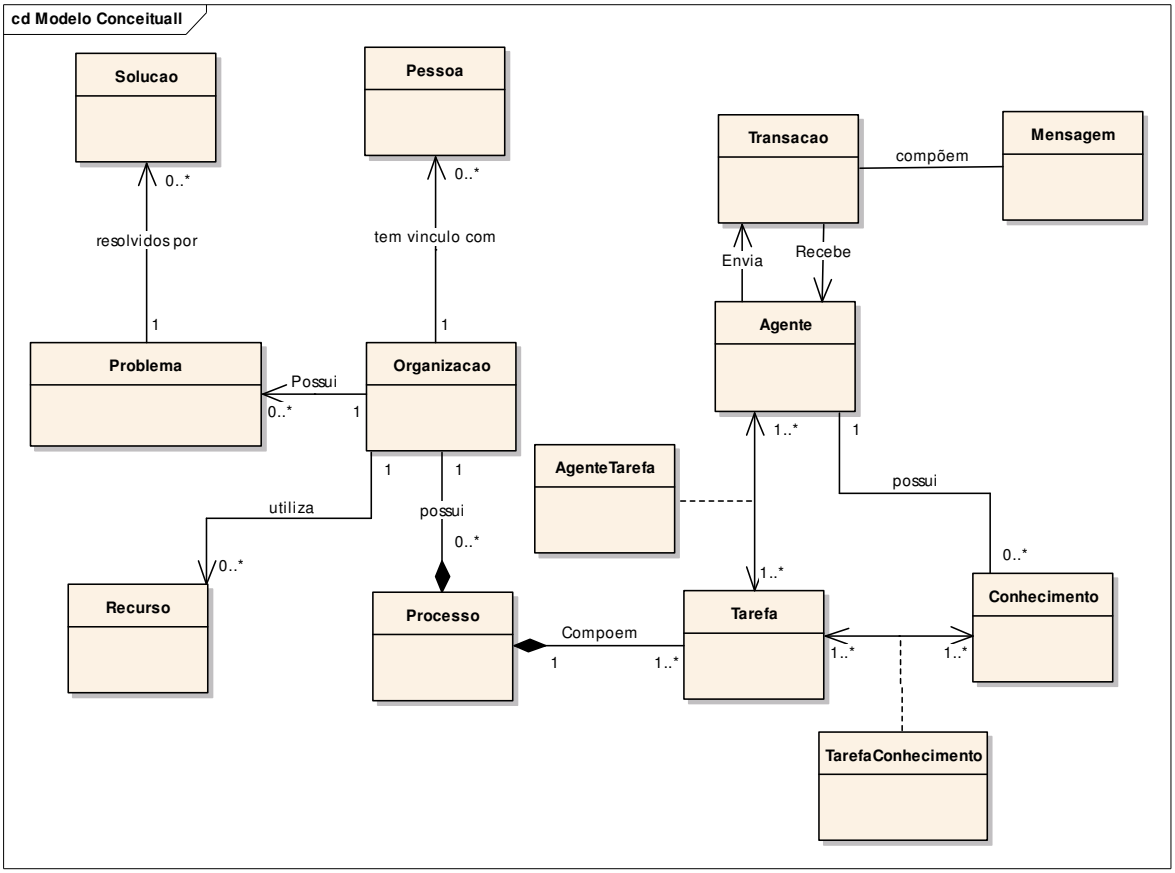

Figura 12: Modelo conceitual

3.5 Definição do Banco de Dados

A partir da especificação do domínio do problema, da definição do modelo conceitual e das informações contidas nas tabelas dos modelos do CommonKADS, foi possível projetar o banco de dados relacional. A figura a seguir representa o modelo de entidade relacional obtido neste domínio de problema:

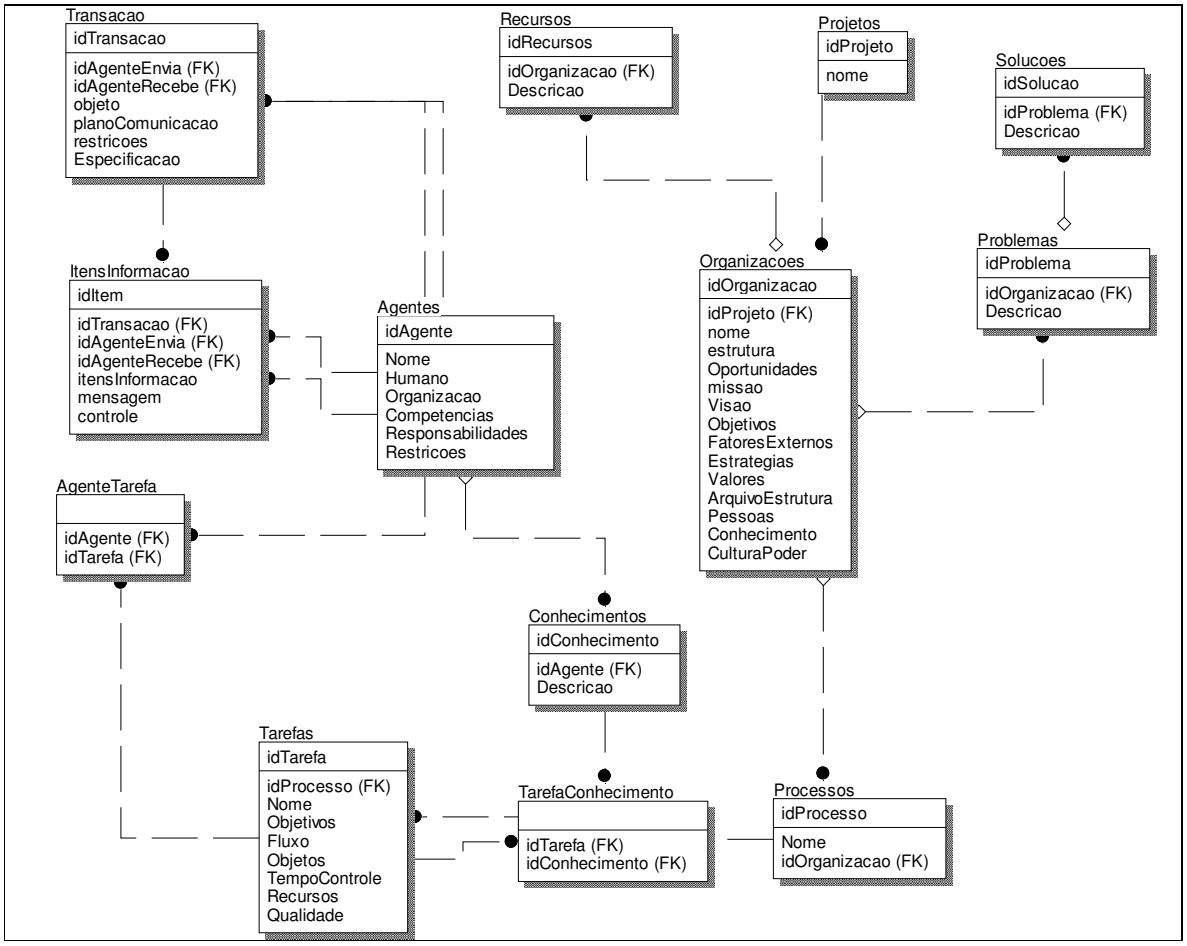

Figura 13: Diagrama de Entidade Relacional do Banco de Dados.

3.6 Considerações sobre a análise e projeto

As etapas de análise e projeto resultaram nas descrições de casos de usos, diagramas e modelos. É importante ressaltar que os resultados obtidos nos diagramas surgiram através de vários ciclos de desenvolvimento do sistema, onde cada ciclo foi composto pelas etapas: análise, projeto e desenvolvimento.

Dessa forma, em cada ciclo novas observações e detalhes do sistema foram surgindo e os modelos e diagramas foram sendo adequados.

3.7 Considerações sobre a implementação

Para realizar a implementação do projeto optou-se pelos seguintes recursos e ferramentas:

- Linguagem Java (jre1.5.0.04) com IDE NetBeans 4.0;

- Banco de dados MS-SQL Server 2000;

- Framework de persistência Hibernate 3.

- Interface Swing;

A determinação da arquitetura na qual uma aplicação será desenvolvida tem papel fundamental, esta decisão é tomada no momento do projeto da aplicação e deve ser seguida durante toda a implementação do projeto.

O projeto proposto neste trabalho seguiu a arquitetura de camadas MVC (Model View Control), na qual a aplicação é estruturada em três camandas diferentes:

a) Model: camada na qual está estruturada toda a lógica de negócios do domínio do problema. Pode-se considerar que o coração da aplicação está nesta camada, pois nela é

implementada todas as características, regras, cálculos, validações,etc., que foram levantados durante o projeto do sistema.

b) View: camada responsável pela apresentação da informação. No caso do projeto proposto neste trabalho, optou-se por utilizar a interface Swing.

c) Control: camada responsável pelos dados da que são incluídos, alterados e excluídos do banco de dados. Esta camada faz a comunicação com o banco de dados e repassa as informações para a camada superior (Model). No projeto proposto neste trabalho, optou-se por utilizar o framework de persistência Hibernate como camada Control.

3.8 Interfaces do Sistema

3.8.1 Janela inicial do Sistema

Conforme estabelecido nos requisitos do sistema, quando o sistema é executado abre-se a tela para que seja selecionado/criado o projeto com o qual deseja-se trabalhar.

Figura 14: Janela inicial do sistema.

3.8.2 Janela Principal do Sistema

Após a seleção / criação do projeto a janela principal do sistema é apresentada mostrando as informações que já estão cadastradas no projeto. As interfaces do sistema

foram construídas buscando o máximo de similaridade com as tabelas dos modelos do CommonKADs.

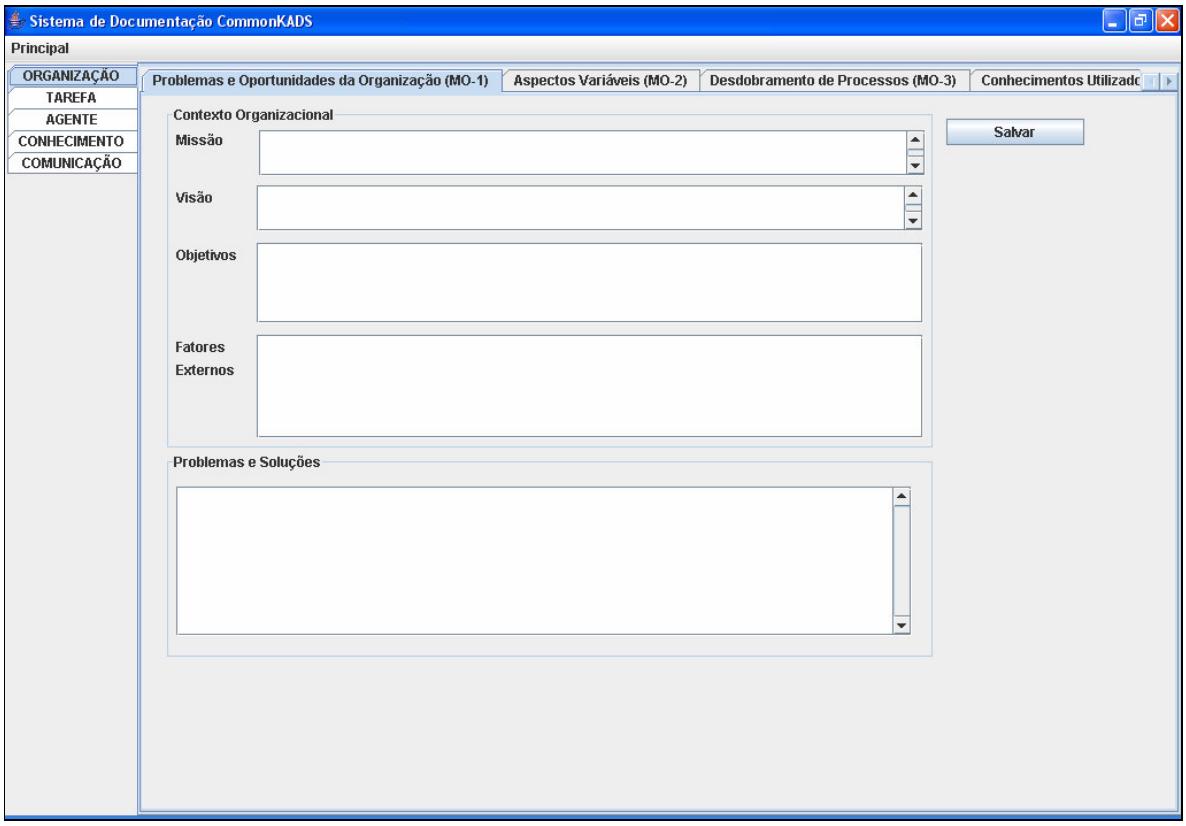

Figura 15: Janela principal do Sistema.

Ao lado esquerdo a janela são apresentados os modelos do CommonKADS, clicando-se o nome do modelo tem-se acesso as informações das tabelas do mesmo.

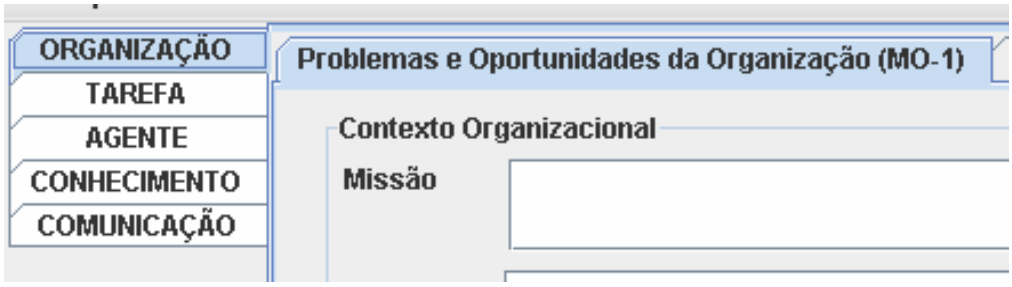

Figura 16: Detalhamento das guias laterais para acesso aos modelos

A seguir serão apresentados os detalhes de cada uma das guias dos modelos. Para saber mais sobre cada modelo do CommonKADS consulte o capítulo 2.1 deste trabalho.

3.8.2.1 Guia Organização – tabelas do Modelo da Organização

A guia “Organização” apresentada à esquerda da janela principal é composta de outras 5 guias dispostas na horizontal, estas guias apresentam todas as interfaces necessárias para registras as informações das tabelas do Modelo da Organização do CommonKADS. Conforme é apresentado a seguir:

i. Problemas e Oportunidades da Organização (MO-1)

A primeira guia do modelo de organização traz as informações geradas pela aplicação da tabela MO-1: serão cadastradas as informações da organização como: missão, visão, objetivos, fatores externos, os problemas e oportunidades e as soluções para os problemas.

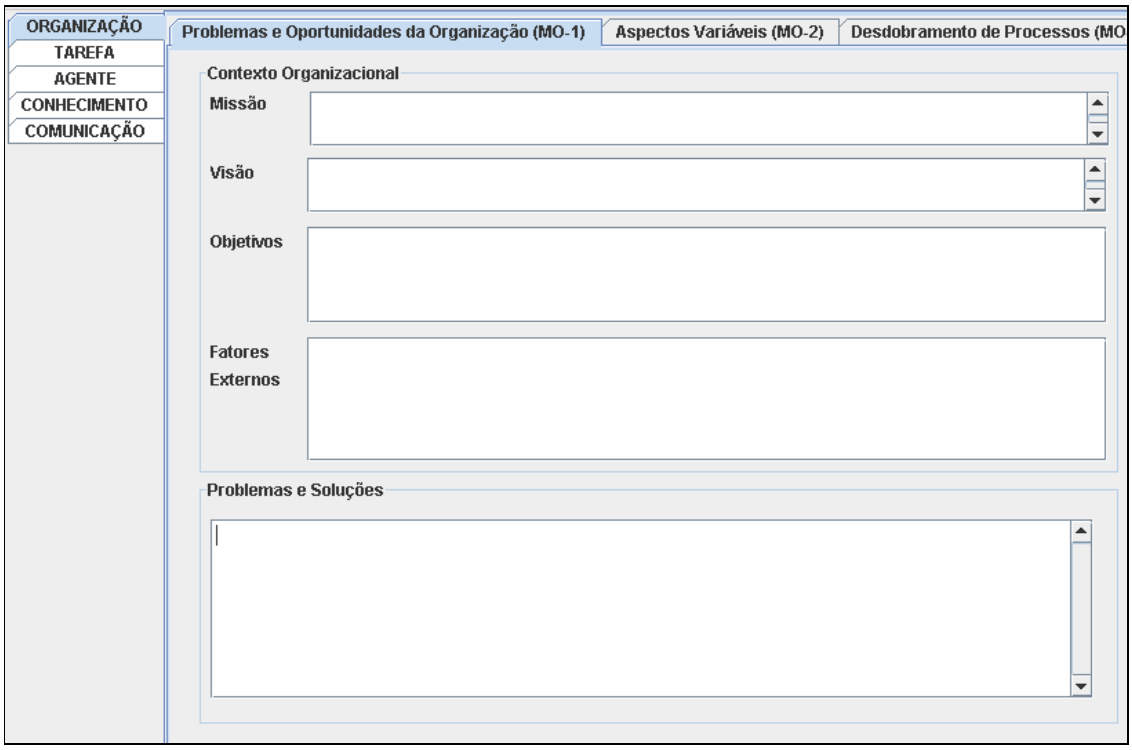

Figura 17:Guia “Problemas e Oportunidades da Organização (MO-1)”

ii. Aspectos Variáveis (MO-2)

A segunda guia do modelo da organização permite o registro das informações obtidas com a aplicação da tabela MO-2 que diz respeito aos aspectos variáveis da organização.

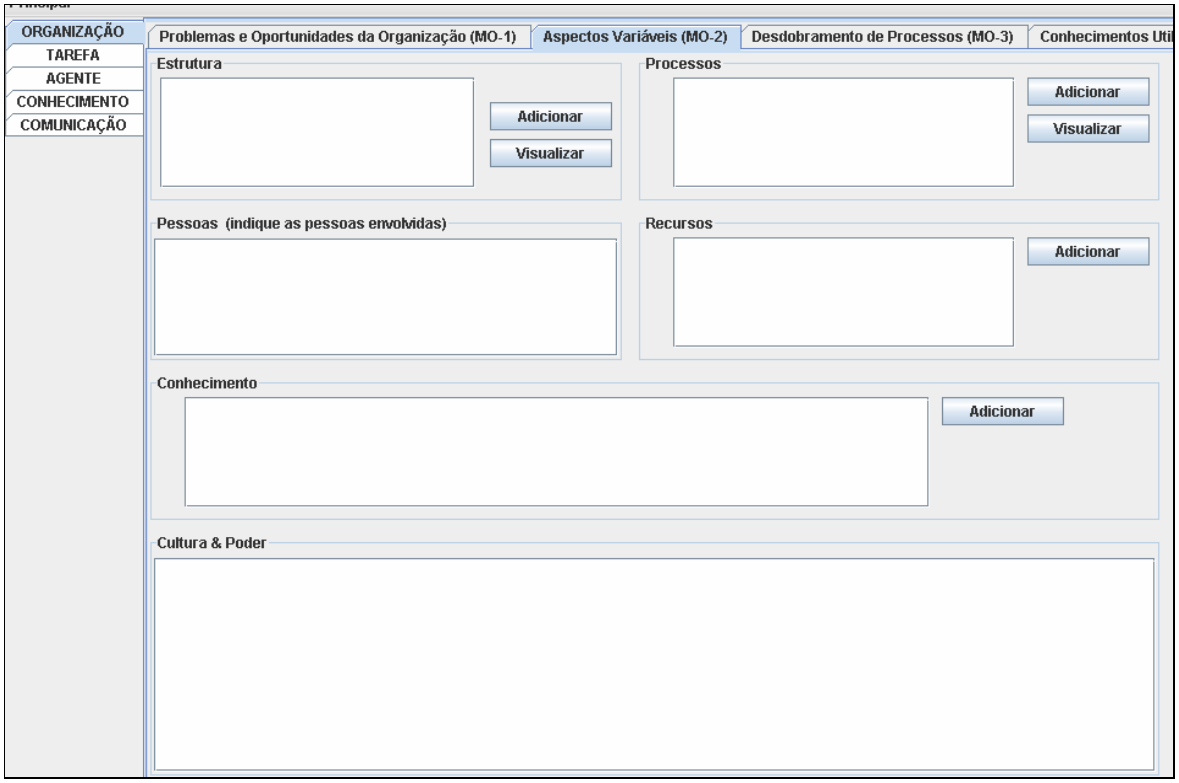

Figura 18: Guia “Aspectos Variáveis (MO-2)”.

Na área “Estrutura”, através do botão “Adicionar” o usuário pode adicionar a imagem que represente a estrutura da empresa.

Os recursos, processos e conhecimentos podem ser adicionados através do botão “Adicionar” que se encontra ao lado de cada uma das caixas de texto dos mesmos.

iii. Desdobramento de Processos (MO-3)

A terceira guia permite o registro das informações obtidas com a aplicação da terceira tabela do Modelo da Organização. Nela é possível registrar o desdobramento dos processos em tarefas, informando se a tarefa é de conhecimento intensivo e qual o seu grau de significância. Nesta etapa, os processos que foram cadastrados anteriormente estarão disponíveis para seleção.

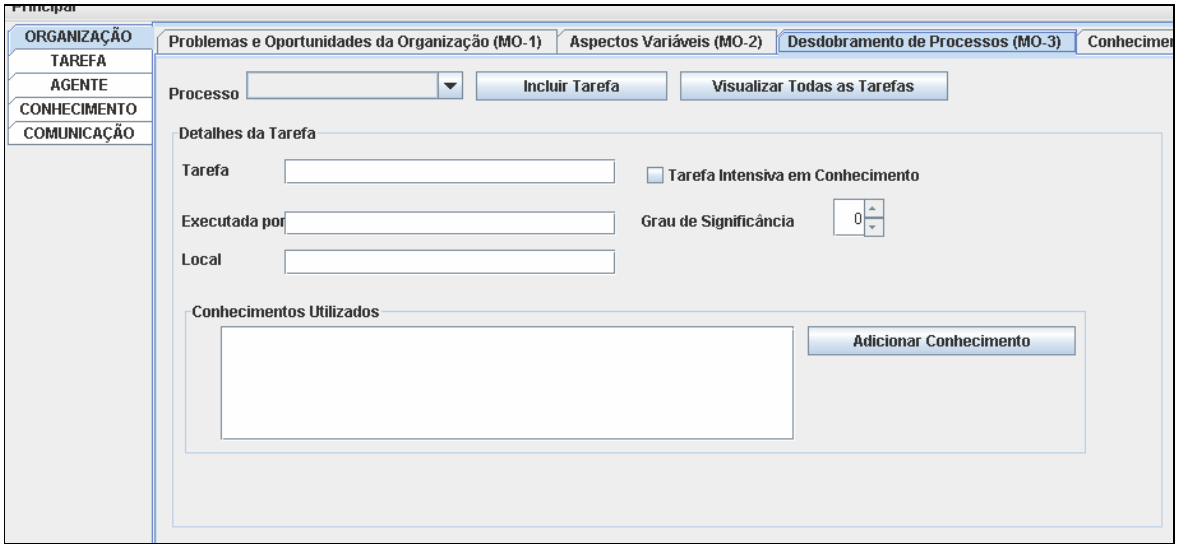

Figura 19: Guia “Desdobramento de Processos (MO-3)”

Através do botão “Incluir Tarefa” o usuário pode adicionar quantas tarefas forem necessárias para fazer o desdobramento do processo que for selecionado. Neste momento também será possível relacionar quais são os conhecimentos utilizados na execução da tarefa, para isso basta clicar no botão “Adicionar Conhecimento”.

iv. Conhecimentos Utilizados (MO-4)

A quarta guia permite o registro das informações obtidas através da aplicação da quarta tabela do Modelo da Organização, que faz o detalhamento dos conhecimentos registrados na tabela MO-3. Nesta etapa, os conhecimentos registrados na guia “Desdobramento de Processos” estarão disponíveis para serem acessados e detalhados.

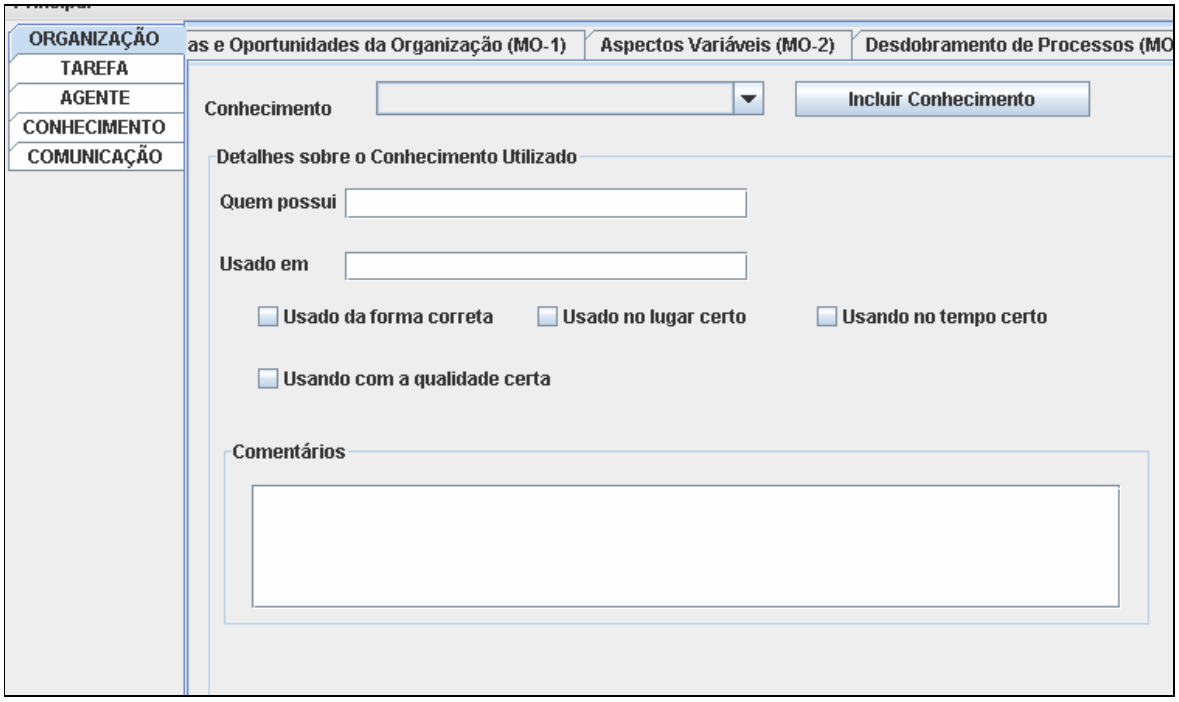

Figura 20: Guia "Conhecimentos Utilizados (MO-4)".

Através do botão “Incluir Conhecimento” é possível incluir novo conhecimento.

v. Documento de decisão de viabilidade (MO-5)

A quinta guia traz os questionamentos para o checklist de decisão de viabilidade. Esta guia foi sub-dividida em 4 outras guias para que o usuário possa trabalhar com textos longos. Cada uma dessas guias traz uma série de questionamentos que visam auxiliar o usuário na criação do documento de viabilidade. Estes questionamentos são que os mesmos que Schreiber et. al.,2002, apresentam na tabela MO-5 do Modelo da Organização do CommonKADS.

As 4 guias são apresentadas a seguir:

Viabilidade de Negócios:

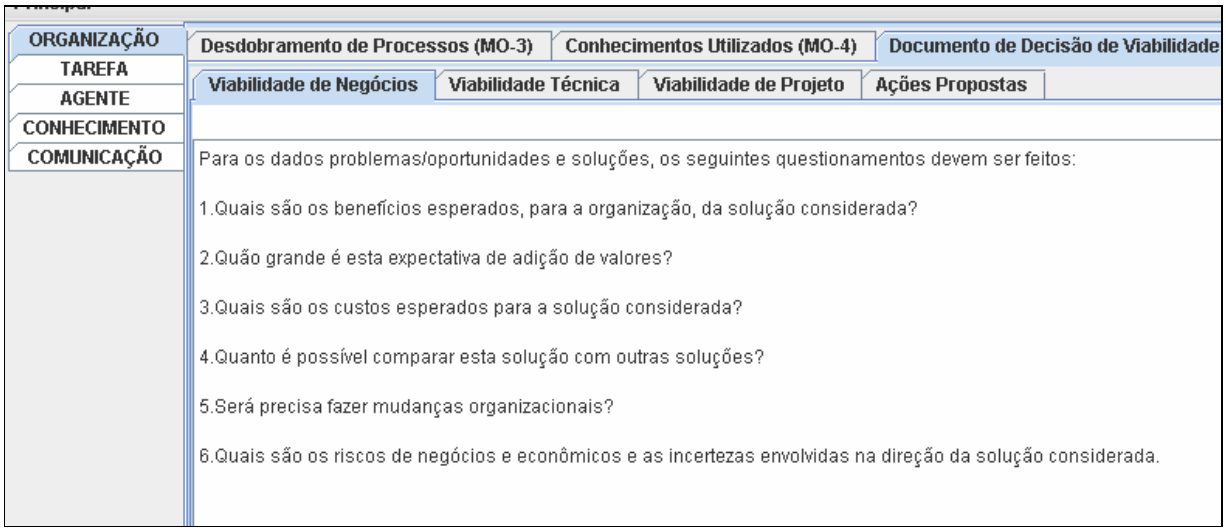

Figura 21: Guia "Viabilidade de négócios" do Documento de Decisão de Viabilidade.

Viabilidade Técnica:

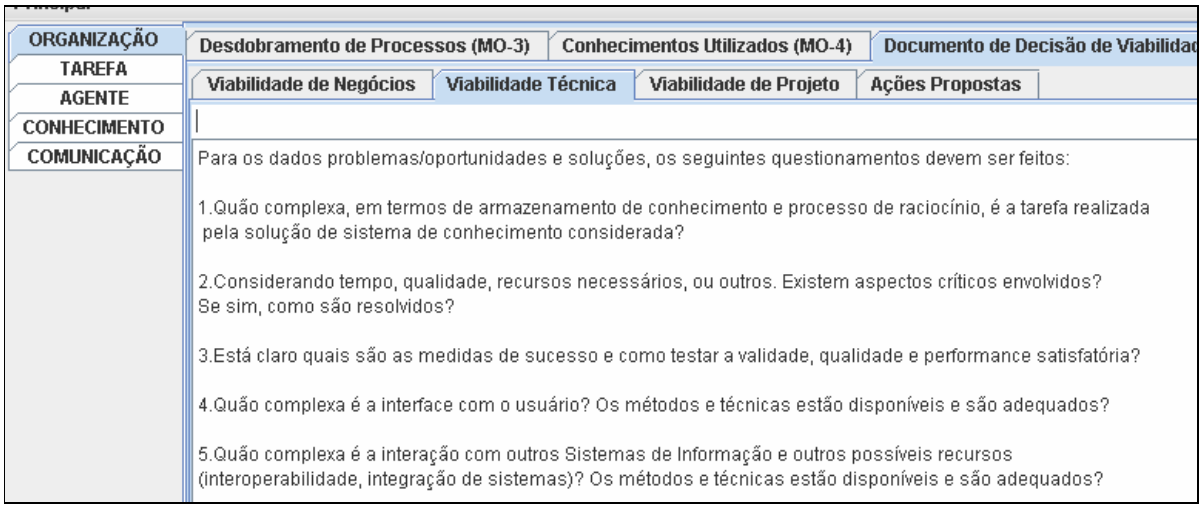

Figura 22: Guia: "Viabilidade Técnica" do Documento de Decisão de Viabilidade.

Viabilidade de Projeto:

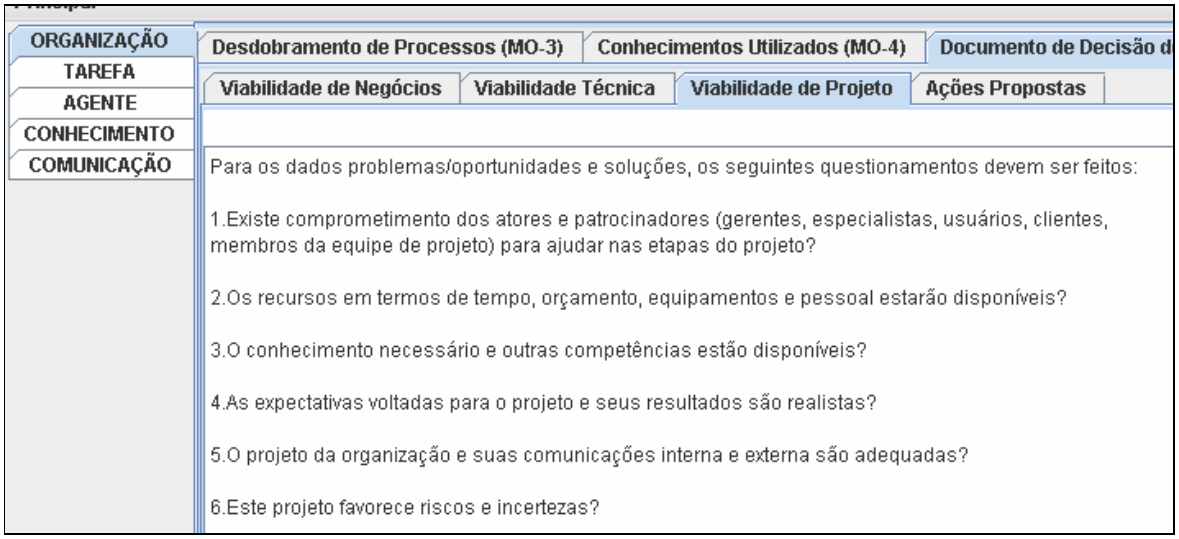

Figura 23: Guia "Viabilidade de Projeto" do Documento de Decisão de Viabilidade.

Ações Propostas:

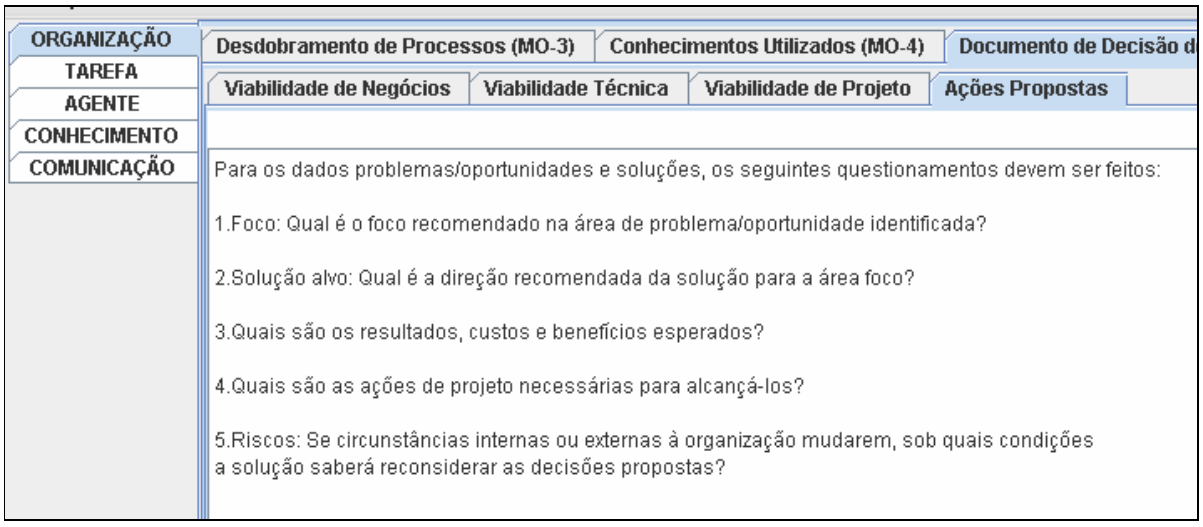

Figura 24: Guia "Ações Propostas" do Documento de Decisão de Viabilidade.

3.8.2.2 Guia Tarefa – tabelas do Modelo da Tarefa

A segunda guia localizada à esquerda da janela principal é a guia “Tarefa”, e traz mais duas guias dispostas horizontalmente, conforme é apresentado a seguir:

i. Análise da Tarefa (MT-1)

A primeira guia permite o registro das informações obtidas com a aplicação da tabela MT-1 do Modelo da Tarefa. Neste momento as tarefas cadastradas anteriormente poderão ser acessadas para que seja realizado seu detalhamento.

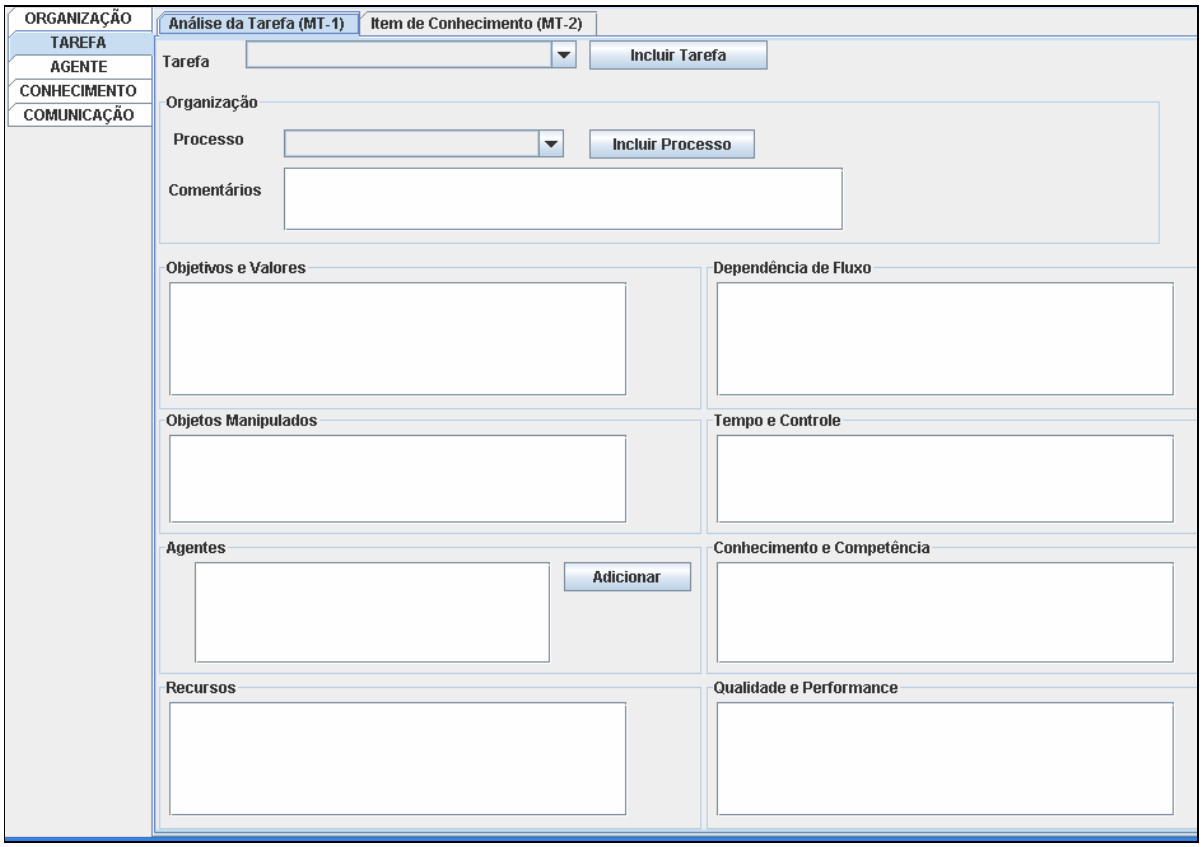

Figura 25: Guia "Análise da Tarefa (MT-1)"

Através do botão “Incluir Tarefa” o usuário poderá cadastrar uma tarefa que ainda não está cadastrada no sistema, bem como, através do botão “Incluir Processo”, cadastrar e relacionar um novo processo à tarefa selecionada no momento.

Na área “Agentes” é possível cadastrar e relacionar quais são os agentes que executam esta tarefa.

ii. Item de Conhecimento (MT-2)

A segunda guia permite o registro das informações obtidas com a aplicação da tabela MT-2 do Modelo da Tarefa. Neste momento o usuário poderá detalhar os itens de conhecimento que estão relacionados com a tarefa.

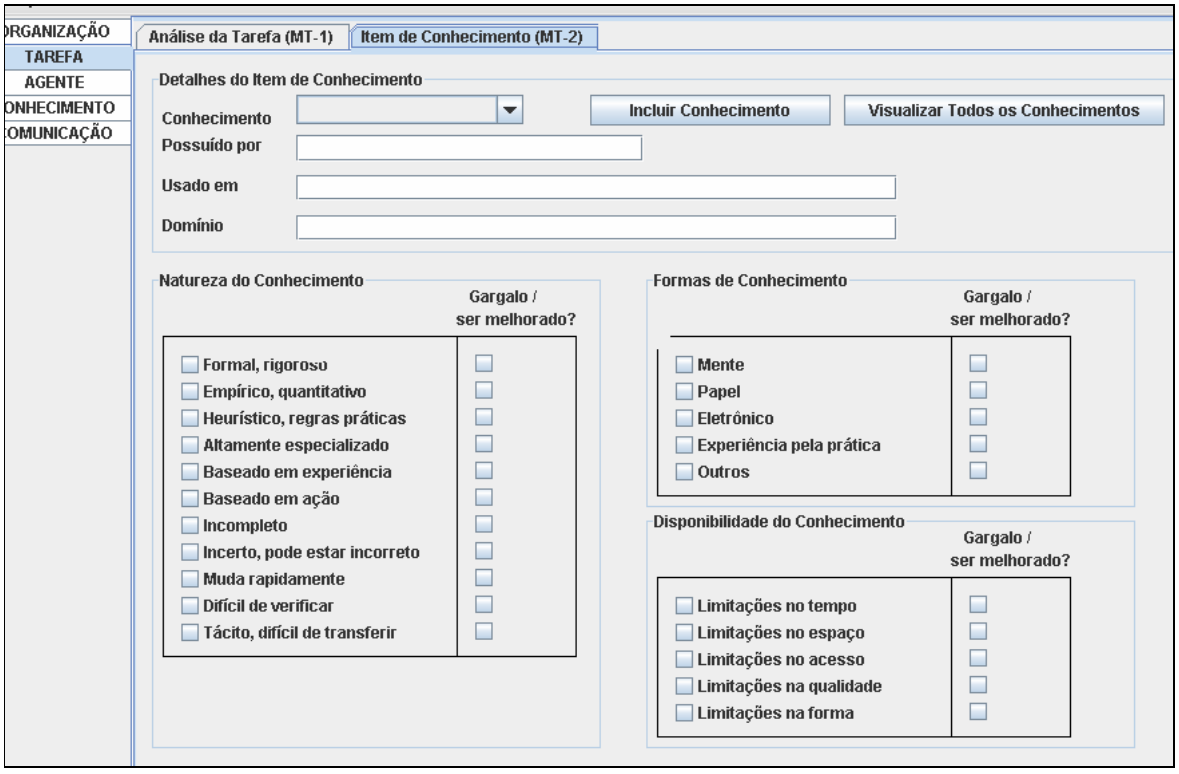

Figura 26: Guia "Item de Conhecimento (MT-2)"

3.8.2.3 Guia Agente – tabelas do Modelo de Agentes e Documento de Decisão

A terceira guia localizada à esquerda da janela principal é a guia “Agente”, e traz duas guias dispostas horizontalmente, conforme é apresentado a seguir:

i. Agentes (MA-1)

A primeira guia permite o registro das informações obtidas através da aplicação da tabela MA-1 do Modelo de Agentes.

Nesta guia é possível cadastrar agentes e relacioná-los a tarefas, conhecimento e outros agentes.

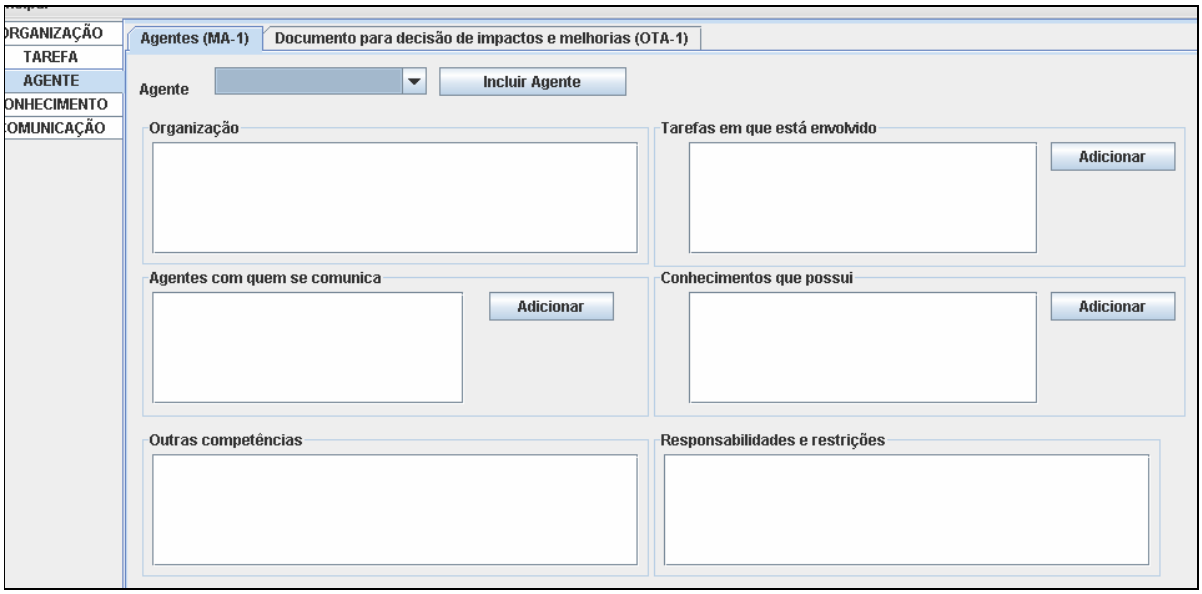

Figura 27: Guia "Agentes (MA-1)"

ii. Documento para decisão de impactos e melhorias (OTA-1)

A quinta guia traz os questionamentos para o checklist de decisão de impactos e melhorias. Trata-se do registro das informações da tabela que unifica os modelos da Organização, Tarefa e Agentes. Esta guia foi sub-dividida em 4 outras guias para que o

usuário possa trabalhar com textos longos. Cada uma dessas guias traz as explicações e questionamentos apresentados por Schreiber et. al.,2002, na tabela OTA-1.

As 4 guias são apresentadas a seguir:

Impactos e Mudanças na Organização:

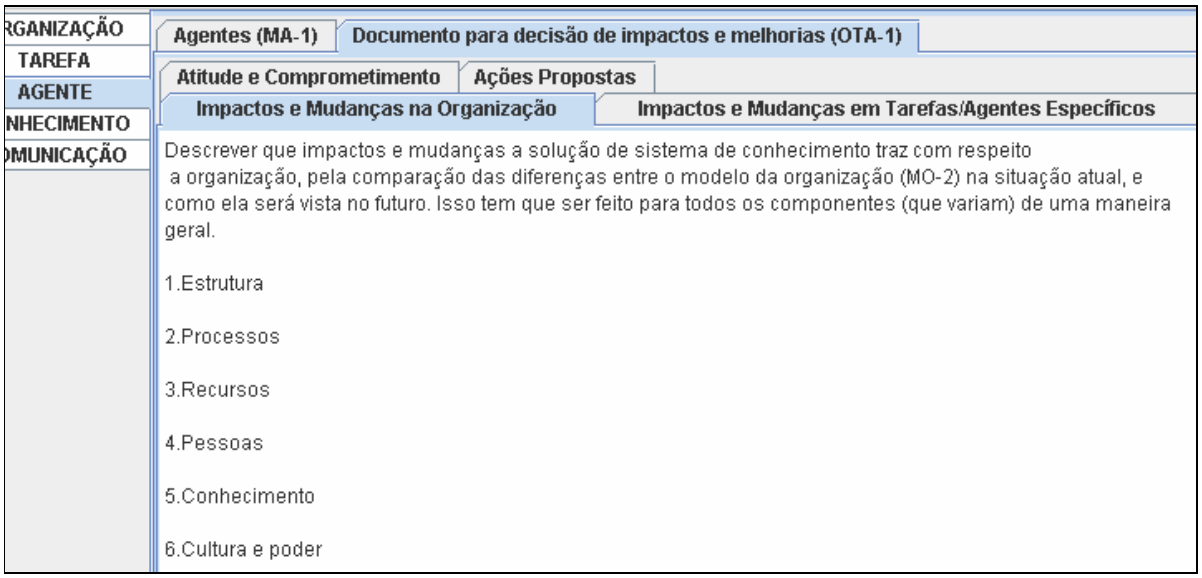

Figura 28: Guia "Impactos e Mudanças" do Documento para decisão de impactos e melhorias (OTA-1)

Impactos e Mudanças em Tarefas/Agentes específicos:

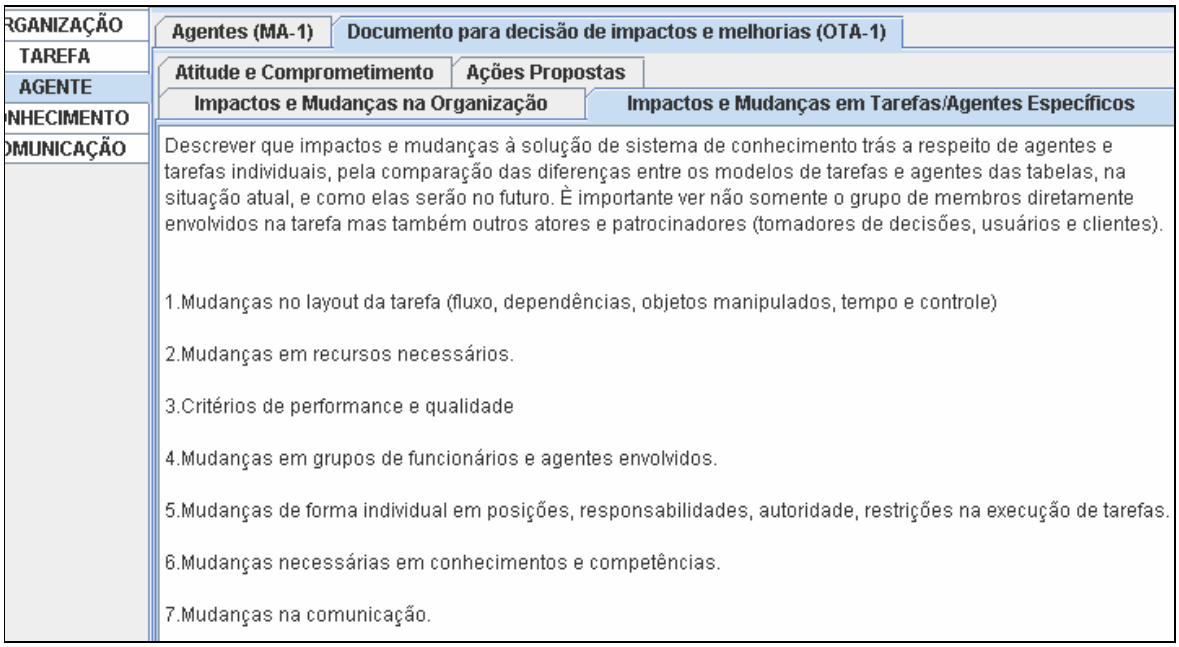

Figura 29: Guia "Impactos e Mudanças em Tarefas/ Agentes Específicos" do Documento para decisão de impactos e melhorias (OTA-1)

Atitude e Comprometimento:

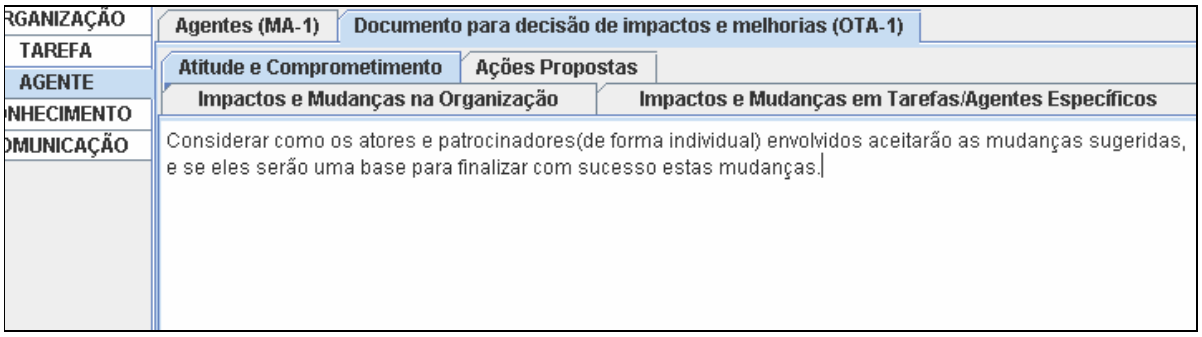

Figura 30: Guia "Atitude e Comprometimento" do Documento para decisão de impactos e melhorias (OTA-1).

Ações Propostas:

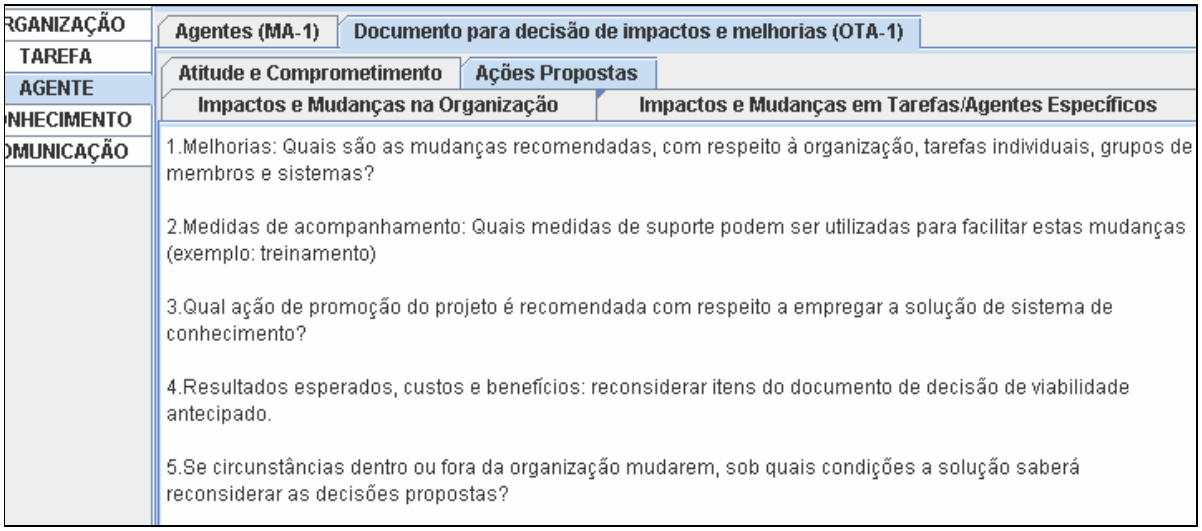

Figura 31: Guia “Ações Propostas” do Documento para decisão de impactos e melhorias (OTA-1).

3.8.2.4 Guia Conhecimento – Modelo do Conhecimento

A guia “Conhecimento” permite fazer o registro dos modelos dos conhecimentos cadastrados no sistema. Estes modelos devem ser definidos através da linguagem CML.

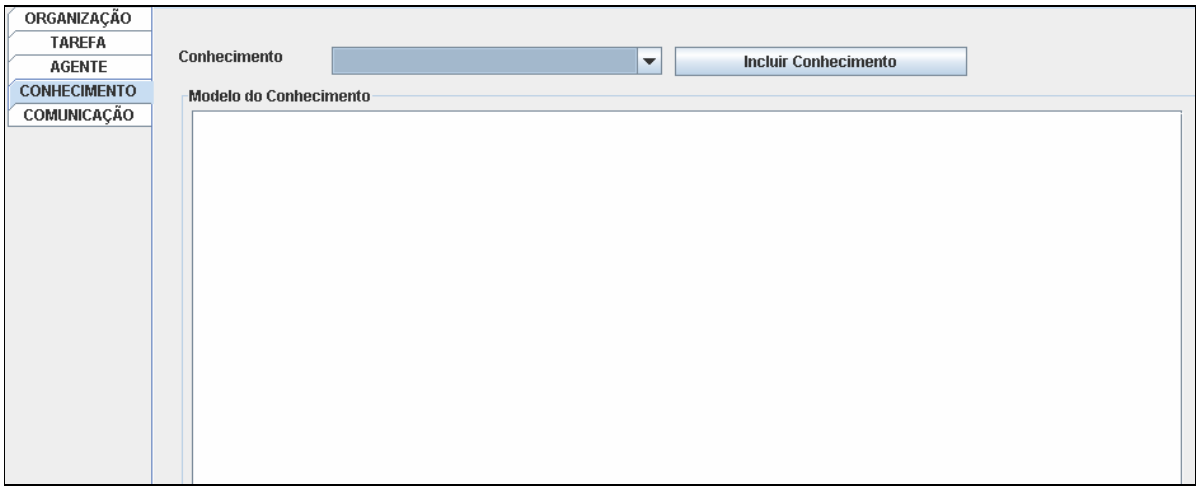

Figura 32: Guia “Conhecimento”.

Através do botão “Incluir Conhecimento” o usuário poderá incluir um item de conhecimento que deseje fazer a modelagem em CML.

3.8.2.5 Guia Comunicação – tabelas do Modelo da Comunicação

A quinta guia localizada à esquerda da janela principal é a guia “Comunicação”, e traz duas guias dispostas horizontalmente, conforme é apresentado a seguir:

i. Descrição da Transação (MC-1)

A guia “Descrição da Transação” permite registrar as informações obtidas através da aplicação da tabela MC-1 do Modelo da Comunicação.

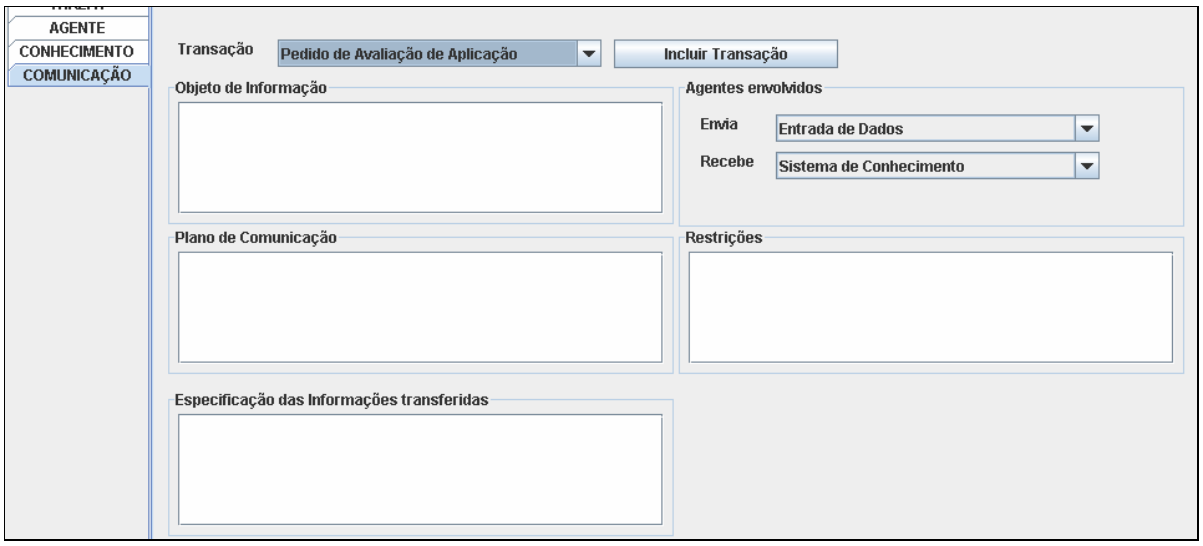

Figura 33: Guia “Descrição da Transação (MC-1)”

Através do botão “Incluir Transação” o usuário poderá cadastrar as transações. Na área “Agentes envolvidos” é possível relacionar quem é o agente que envia, e quem é o agente que recebe as mensagens.

ii. Especificação das Informações transferidas (MC-2)

A guia “Especificação das Informações transferidas” permite definir e quais são os itens de informação que são transferidos na comunicação entre os agentes.

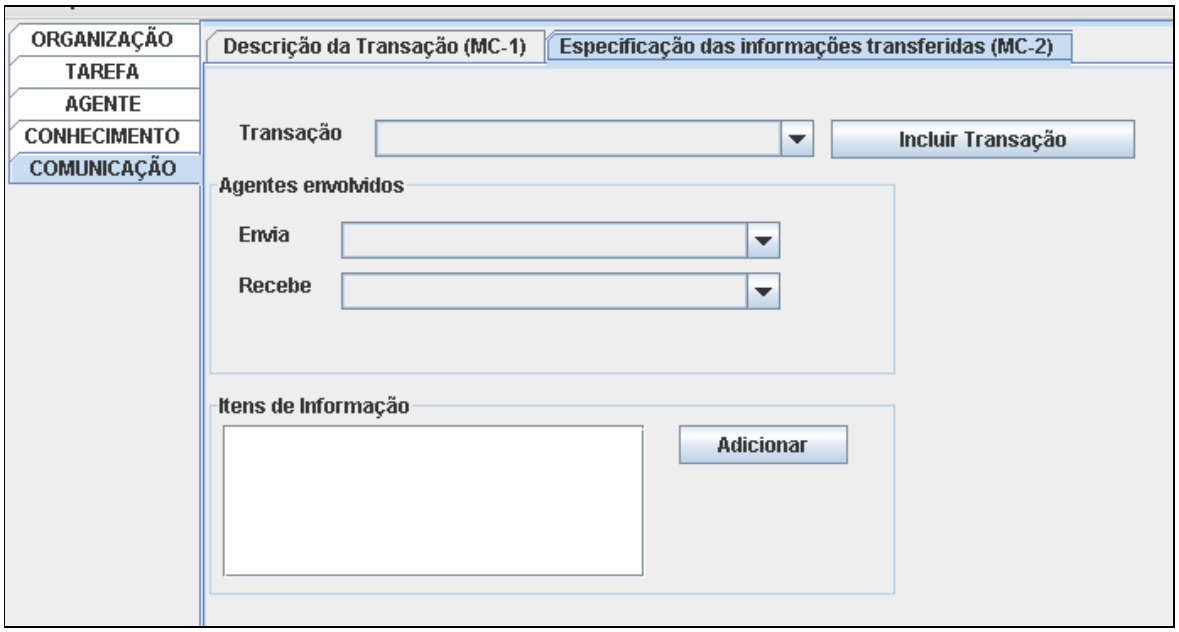

Figura 34: Guia “Especificação das Informações transferidas (MC-2)”.

4 TESTES

4.1 Introdução

A fase de testes corresponde a uma parcela importante no sucesso do desenvolvimento de um sistema de informação. É nesta fase que será verificado se os requisitos do sistema foram completamente atendidos e se o sistema está funcionando perfeitamente.

Para que o teste do sistema proposto neste trabalho tivesse efetividade, foram coletados alguns estudos de casos contidos na literatura e as informações foram cadastradas no sistema. A seguir são apresentados dois estudos de caso que foram incluídos no sistema de forma a mostrar sua efetividade na documentação dos modelos do CommonKADS.

4.2 Teste 1: Estudo de Caso: “Ice-Cream Product Development”

O primeiro teste realizado no sistema foi feito com um estudo de caso apresentado por Schreiber et. al., 2002, sobre o uso do CommonKADS para verificar a viabilidade de se implementar a Gestão do Conhecimento nos negócios de sorvete da empresa Unilever, com relação ao desenvolvimento de produtos.

Este estudo de caso foi importante no teste do sistema, pois serviu como primeiro validador das interfaces e recursos disponíveis no sistema. O fato do estudo de caso apresentar um número considerável de tarefas dentro processo de desenvolvimento de produtos, contribuiu para verificar a efetividade do sistema na documentação de informações. Infelizmente o estudo de casos não foi apresentado na integra, não permitindo que todas as tarefas de conhecimento intensivo fossem detalhadas e registradas no sistema.

O segundo teste realizado no sistema permitiu verificar com maior completeza qual realmente foi a adequação do sistema aos casos práticos de aplicação da metodologia CommonKADS.

4.3 Teste 2: Estudo de Caso: “The Housing Application”

O segundo e mais completo teste do sistema foi realizado através do estudo de caso “Housing Application” apresentado por Schreiber et. al., 2002. Trata-se da análise de um problema cujo domínio é a alocação de casas de aluguel para pessoas que necessitam de uma residência. Após uma avaliação das pessoas inscritas o governo cede as casas para aqueles que melhor se enquadram no perfil de quem realmente precisa do benefício.

A tabela MO-1 do Modelo da Organização ajuda a entender o domínio do problema:

Modelo da Organização	***Problemas e oportunidades – Tabela MO-1***
PROBLEMAS E OPORTUNIDADES	• A avaliação de cada caso para um determinada residência leva muito tempo; • Não há funcionários suficientes para avaliar casos urgentes.
CONTEXTO ORGANIZACIONAL	Missão: • Capacitar pessoas para conseguir tanto quanto possível a própria responsabilidade para encontrar uma casa adequada. • Permitir clareza na dinâmica de locação de casas. Fatores externos • Conselho local; • Regulamentos nacionais; • Candidatos à locação / opinião pública; • Agências de Locação; Estratégia: • Fornecer alta qualidade por um preço razoável; • Mudar para serviços (semi) privados; • Estender o escopo, ou seja, incluir o segmento de locação de imóveis privados.
SOLUÇÕES	Solução 1: - Desenvolver um sistema que faça a avaliação dos candidatos de forma automática. - Definir um programa de treinamento para que um grupo de funcionários se especialize em tratar casos urgentes.

Tabela 28: MO-1 Identificação dos problemas e oportunidades da organização (Fonte: (Schreiber et al., 2002))

O registro das informações no sistema deu-se de forma natural, sendo que em alguns casos, imagens foram substituídas por descrições textuais, como foi o caso de uma imagem que fazia a representação dos processos da organização. A figura a seguir mostra a tabela MO-1 registrada no Sistema:

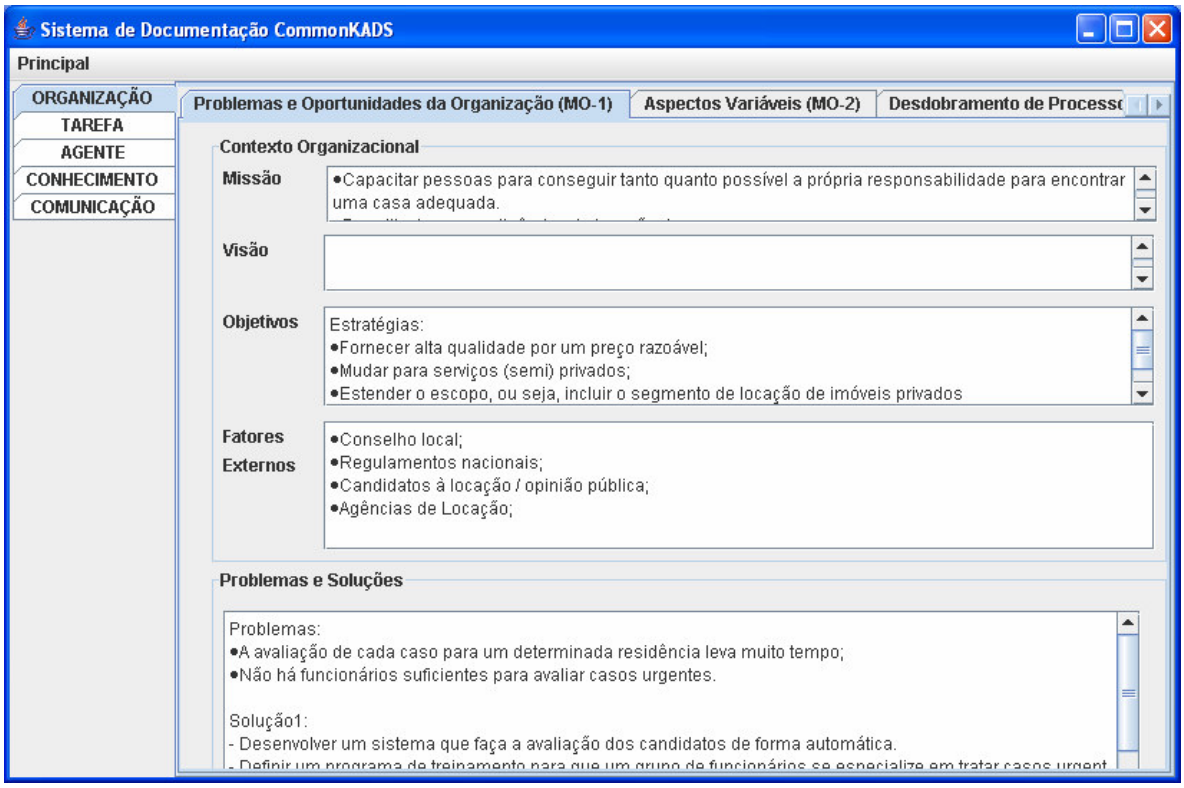

Figura 35: Teste da Interface do Modelo da Organização. Tabela MO-1

A facilidade de se documentar informações através de um sistema foi percebida no momento de se reutilizar informações que já haviam sido cadastradas.

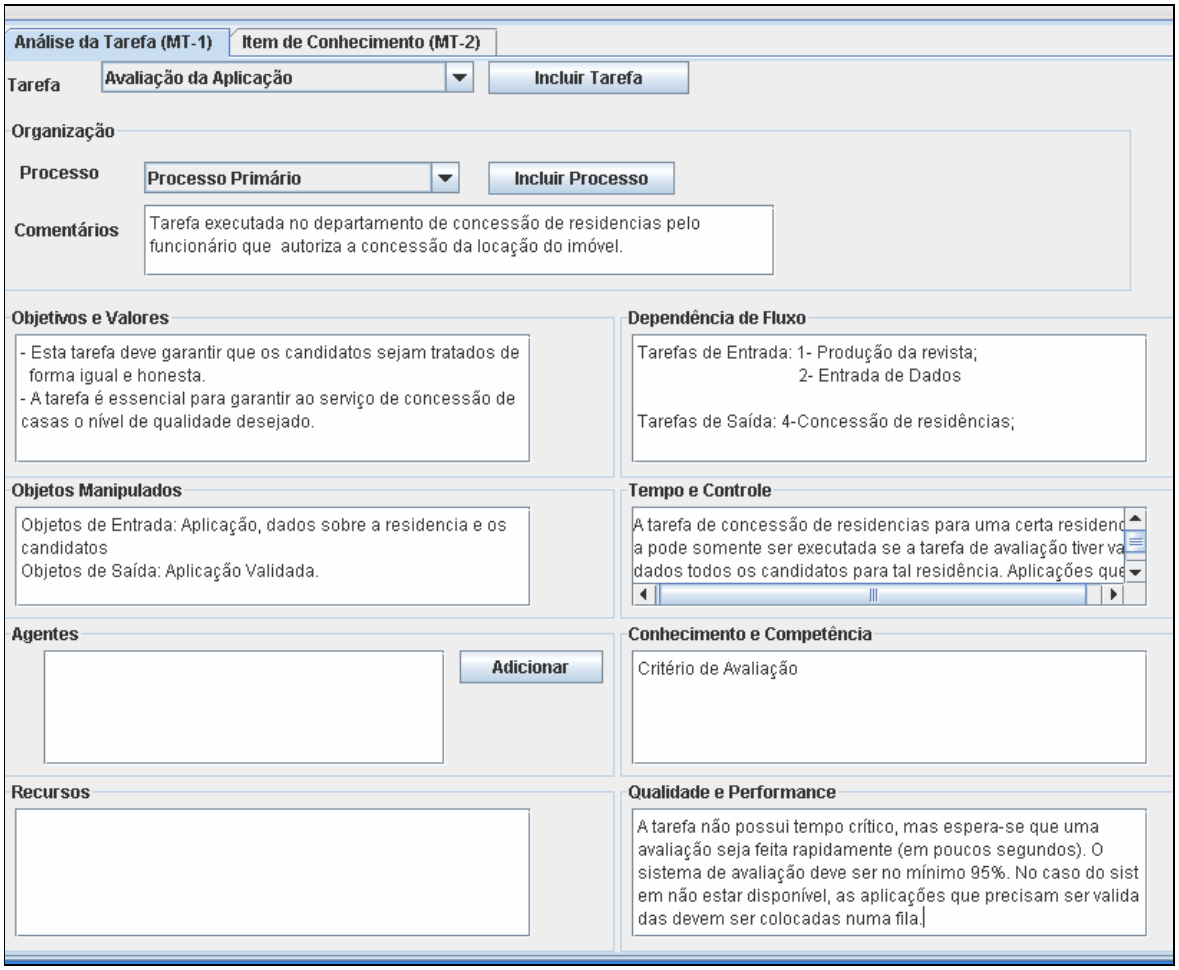

Figura 36: Cadastramento de informações registradas na tabela TM-1 do Modelo da Tarefa.

A figura a seguir mostra a identificação de gargalos de conhecimentos na tarefa “Critério de avaliação”.

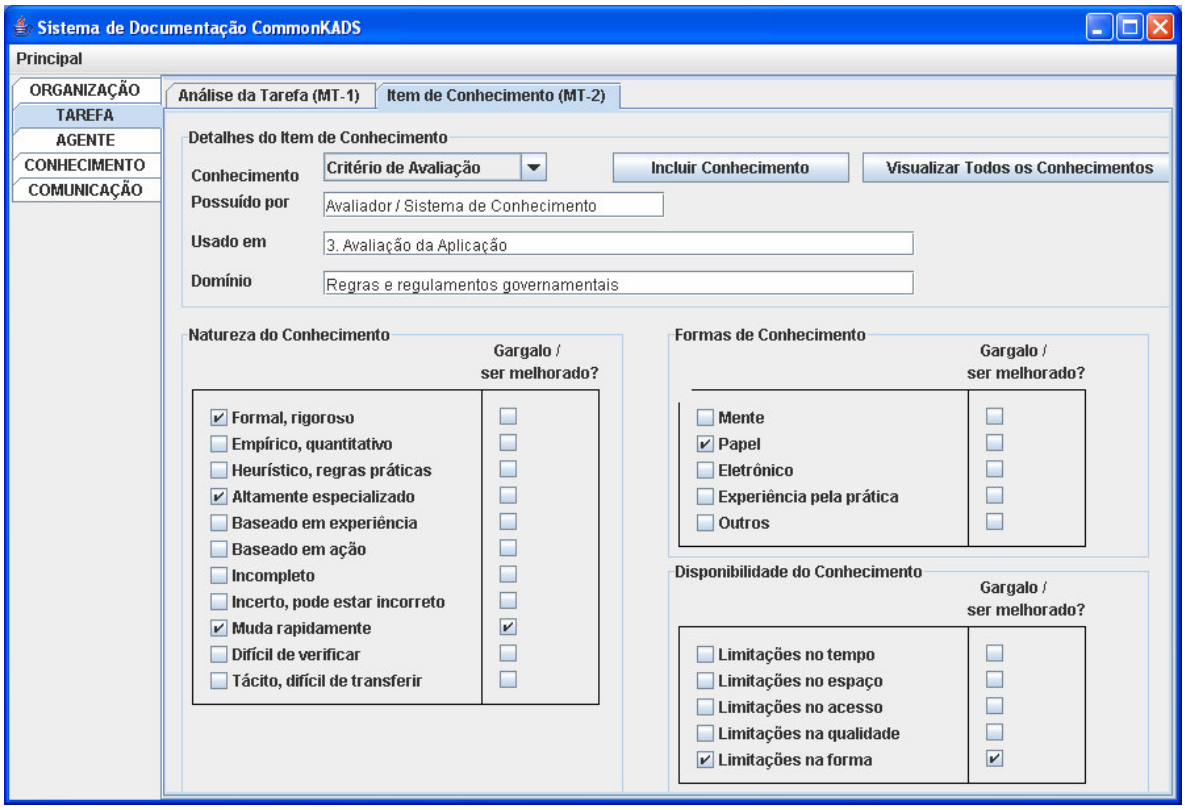

Figura 37: Cadastramento de informações registradas na tabela MT-2 do Modelo da Tarefa

O Modelo de Comunicação do estudo de casos utilizados para fazer o teste do sistema é relativamente simples, porém foi possível verificar a adequação dos campos às informações que foram coletadas nos modelos.

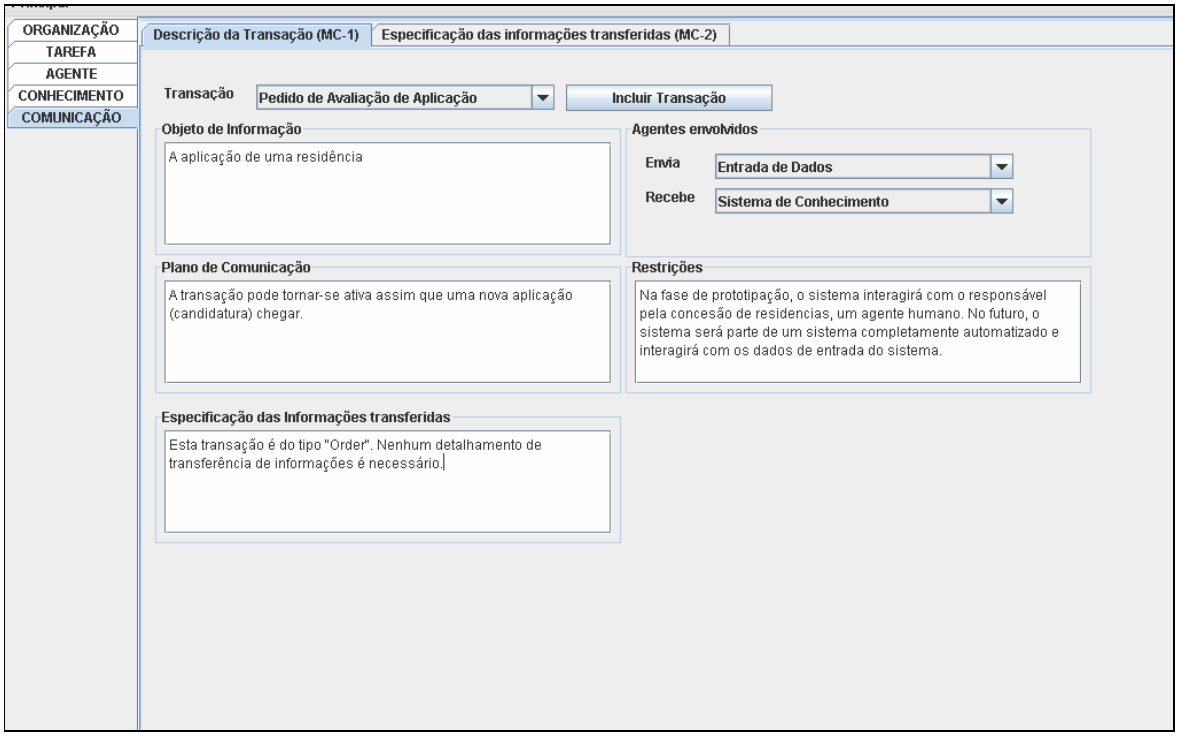

Figura 38: Cadastramento de informações registradas na tabela MC-1 do Modelo da Comunicação

4.4 Considerações sobre os testes realizados

Com a aplicação dos testes no sistema foi possível constatar que existe a efetividade do registro das informações obtidas através da aplicação da metodologia CommonKADS. Contudo, pode-se perceber a necessidade de ter-se campos adicionais para registrar comentários sobre informações registradas nos campos tidos como “padrão” de cada modelo. Outro recurso necessário que foi verificado foi a possibilidade de se anexar documentos externos, como planilhas, textos e imagens.

Certamente, através dos testes realizados pode-se perceber a facilidade que esta ferramenta pode proporcionar para o Engenheiro do Conhecimento no que diz respeito a registrar e recuperar as informações da análise. De forma geral, a fase de teste foi bem sucedida podendo-se considerar o sistema apto a registrar as informações dos modelos do CommonKADS.

5 CONCLUSÕES E TRABALHOS FUTUROS

A Metodologia CommonKADS mostra-se extremamente eficiente em projetos de Engenharia e Gestão do Conhecimento. Sua utilização e disseminação vem crescendo devido a grande popularização da Gestão do Conhecimento em ambientes corporativos.

A aplicação da metodologia CommonKADS em projetos de Engenharia e Gestão do Conhecimento ainda é trabalhosa pela falta de recursos que facilitem o registro e gerenciamento das informações coletadas durante a fase de análise.

O objetivo deste trabalho foi criar uma ferramenta capaz de dirimir ao máximo o trabalho do Engenheiro do Conhecimento na documentação obtida com a análise. Através do estudo das características da metodologia CommonKADS e dos estudos de caso disponíveis na literatura proposta por Schreiber et al. 2002, foi possível fazer a especificação do projeto em UML definindo cada elemento informacional existente e os casos de uso envolvidos na tarefa de documentação das informações.

O desenvolvimento do sistema deu-se através de uma série de ciclos onde, ao final de cada ciclo, foram aplicadas as informações coletadas nos estudos de caso como forma de testar sua efetividade e possibilitar possíveis melhorias.

Os testes aplicados mostraram que o sistema proporciona ao Engenheiro do Conhecimento grande mobilidade tanto no registro de informações, onde é possível reaproveitar informações já cadastradas em outros modelos, como para recuperar os projetos cadastrados de maneira rápida e simples, permitindo ao Engenheiro visualizar suas experiências em projetos anteriores.

Certamente muitas melhorias podem ser feitas na ferramenta concebida a partir deste trabalho, entre estas melhorias é possível destacar:

- O controle de acesso a usuários e a implementação de recursos de segurança como a restrição de acesso a informações;
- A possibilidade de se anexar arquivos em todas as tabelas existentes, como forma de oferecer ao usuário a chance de incluir eventuais documentos que sejam necessários para complementar a documentação;

- A emissão de “alertas” ao usuário informando-o sobre a falta de completeza das informações cadastradas, como por exemplo, uma tarefa que foi especificada no modelo organizacional mas não foi detalhada no modelo da tarefa;
- Uma interface que facilite a decisão da solução mais viável a partir do cruzamento de informações, como a preferência e apoio de cada um dos membros da organização diante de cada solução apresentada;
- O cadastramento de um mesmo projeto em dois tempos: o presente e o futuro de forma facilitar a visualização das mudanças desejadas e seus impactos;
- A transformação do banco de dados numa base de conhecimento que permita a utilização de experiências anteriores.

A modesta contribuição deste trabalho pode ser considerada mais um passo para que a Engenharia e Gestão do Conhecimento tornem-se mais difundidas dentro do meio acadêmico e entre os profissionais de Tecnologias de Informação.

6 Referências Bibliográficas

ABEL M. **Sistemas de Conhecimento**. 2003. Informática, UFRGS

ALKAIM, João L. **Metodologia para Incorporar Conhecimento Intenso às Tarefas de Manutenção Centrada na Confiabilidade Aplicada em Ativos de Sistemas Elétricos**. 2003. Tese (Doutorado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, UFSC.

ANGELONI, M. T. (org.) **Organizações do Conhecimento**: infra-estrutura, pessoas e tecnologias. São Paulo: Saraiva, 2002.

BOOCH, G; RUMBAUGH, J e JACOBSON, I: **UML, Guia do Usuário: tradução**; Fábio Freitas da Silva, Rio de Janeiro, Campus ,2000.

CAVANESS, C. Programming Jakarta Struts. O'Reilly, 2003. ISBN: 0-596-00328-5.

Epistemics. CommonKADS Editor. Disponível na internet. <http://www.epistemics.co.uk/ekads/> Acesso em: 10/10/2005

Epistemics. **CommonKADS**. Disponível na internet <http://www.cs.helsinki.fi/u/linden/research/vital92-95/> Acesso em: 10/10/2005

FREITAS JÚNIOR, Olival de Gusmão. **Um Modelo de Sistema de Gestão do Conhecimento para Grupos de Pesquisa e Desenvolvimento**. Florianópolis, 2003. 310 f. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis.

LARMAN, Craig. **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process**. Prentice Hall PTR. 2002

PACHECO, R. C. e SANTO, Neri dos. **Introdução à Engenharia e Gestão do Conhecimento**. 2004. Curso de Pós-Graduação em Engenharia e Gestão do Conhecimento, UFSC

SCHREIBER, G.; Akkermans, H.; Anjewierden, A.; Hoog, R.; Shadbolt, N.; de Velde, W. V.; and Wielinga, B.. **Knowledge Engineering and Management: the CommonKADS Methodology**. MIT Press. Cambridge. Massachussets. 2002

ZAMAI, Kristiany Kukert, **Framework para Gestão de Informações Institucionais em Sistemas de Governo Aplicados aos Sistema Nacional de Inovação.** Dissertação de Mestrado. Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina. 2005

ANEXOS

ANEXO 1 – Artigo

Sistema de Apoio à Aplicação da Metodologia CommonKADS em projetos de Engenharia e Gestão do Conhecimento

Cleosvaldo Gabriel Vieira Jr.
Universidade Federal de Santa Catarina
Sistemas de Informação
cgvjr@inf.ufsc.br

Resumo

A aplicação da Gestão do Conhecimento nas organizações tem crescido constantemente. Cada vez mais empresas procuram gerir o conhecimento de forma estratégica para tornarem-se mais competitivas. A Gestão do Conhecimento possui íntima relação com a Engenharia do Conhecimento, visto que toda a análise, modelagem e estudos necessários para gerar sistemas de Gestão do Conhecimento estão fundamentados nas teorias definidas na Engenharia do Conhecimento. Diferentes metodologias foram criadas para auxiliar o Engenheiro do Conhecimento. Atualmente a metodologia mais completa e utilizada é a CommonKADS, contudo, não se encontram disponíveis muitas ferramentas capazes de auxiliar o Engenheiro do Conhecimento quanto à gestão da documentação gerada a partir da aplicação desta metodologia..
Nesse contexto, o trabalho descrito neste artigo propõe um sistema de informação com a finalidade de armazenar e gerenciar as análises e resultados gerados a partir da aplicação da metodologia CommonKADS. Serão apresentados os conceitos desta metodologia, os requisitos e o sistema desenvolvido além dos testes realizados que mostram que o sistema permite a fácil recuperação e reutilização das informações registradas, diminuindo significativamente o trabalho que o Engenheiro do Conhecimento tem ao documentar sua análise.

1. Introdução

A Engenharia do Conhecimento deu seus primeiros passos na década de 60 com a criação de sistemas especialistas setorizados. Estes sistemas resumiam-se na transferência do conhecimento humano para uma base de conhecimento que era implementada de forma a tornar tal conhecimento acessível. O foco principal estava nos profissionais especialistas que através de entrevistas explicitavam como resolviam tarefas específicas. A partir dai estas informações eram armazenadas e criavam-se regras para manipulá-las.

Com a evolução das pesquisas, a partir da década de 80, deu-se início a segunda geração de sistemas especialistas, onde a proposta de transferência de conhecimento cedeu espaço para a proposta de modelagem conhecimento (Benjamins apud Alkaim, 2003, p.107), que trouxe consigo maior suporte para a área de aquisição de conhecimento.

A percepção de que o conhecimento não deve ser perdido e pode ser utilizado de forma estratégica, desde que gerido corretamente, fez com que a produção de sistemas para gestão de conhecimento aumentasse e encontrasse na Engenharia do Conhecimento os fundamentos e métodos necessários para conceber sistemas com a capacidade de atender as diversas situações e necessidades das grandes corporações.

A Engenharia do Conhecimento guarda íntima relação com a Gestão do Conhecimento, considerando as organizações como inimagináveis sem a utilização de avançados sistemas de informação, baseados nos avanços da Engenharia de Software, aplicados a projetos em que conhecimento e raciocínio são centrais (Schreiber et. al.,2002).

Ao longo da história de desenvolvimento de sistemas de conhecimento, novas abordagens e ferramentas de análise, aquisição de conhecimento e projeto de sistemas inteligentes fora propostas. Para a modelagem de sistemas algumas metodologias foram criadas, dentre elas é possível destacar a CommonKADS que traz a junção métodos e conceitos utilizados em outras metodologias, tornando-se uma das metodologias mais difundida e testada em projetos reais.

2. A Metodologia CommonKADS

A Metodologia CommonKADS integra características de outras metodologias orientadas a modelos e abrange diversos aspectos do projeto de desenvolvimento de um sistema de conhecimento, incluindo: análise organizacional; gerenciamento de

projetos; aquisição, representação e modelagem do conhecimento; integração e implementação de sistemas (Freitas, 2003).

O CommonKADS possui um conjunto de 6 modelos que especificam todos os aspectos ligados à aplicação a ser desenvolvida, incluindo a organização, os recursos humanos, os aspectos de implementação e a interação entre eles. Além disso, oferece suporte à realização de atividades de modelagem, atividades de gestão de projetos e reusabilidade (Schreiber et al., 2002).

A abrangência da metodologia CommonKADS é significativamente grande. Ela fornece recursos para todos os profissionais do conhecimento: analista, gerente, desenvolvedor e gerente de projetos. Outras características importantes do CommonKADS:

- Traz métodos para a análise organizacional (incluindo problemas / identificando oportunidades), teste de viabilidade e aderência de soluções;
- Modelos para identificação de tarefas, processos, agentes e a comunicação dentro da organização.
- Técnicas de aquisição de conhecimento;
- Possui templates de modelos de conhecimento;
- Notação UML para definição de todas as etapas de análise e projeto

Os modelos da metodologia CommonKADS são:

- *Modelo da Organização* - Apóia a análise das maiores características da organização, a fim de descobrir problemas e oportunidades para sistemas de conhecimento, estabelecer sua viabilidade e acessar o impacto das ações de conhecimento pretendidas na organização.
- *Modelo da Tarefa* - analisa o layout das principais tarefas do domínio, suas entradas, saídas, pré-condições e critérios de performance, bem como recursos e competências necessários. Com a aplicação deste modelo tem-se a identificação de quais tarefas possuem conhecimento intensivo.
- *Modelo do Agente* - descreve as características dos agentes, em particular suas competências, autoridades e restrições para agir. Além disso, relaciona os links de comunicação entre agentes necessários para executar uma tarefa.
- *Modelo do Conhecimento* – descreve o conhecimento envolvido no domínio do projeto. Com este modelo é possível detalhar como o conhecimento está relacionado em cada tarefa, quais agentes o possuem e como seus componentes relacionam-se entre si.
- *Modelo de Comunicação* - Dado que muitos agentes podem estar envolvidos em uma tarefa, é importante modelar a transação de comunicação entre os agentes envolvidos. Isso é feito pelo modelo de comunicação, de forma independente de implementação ou de conceito, como ocorre no modelo de conhecimento.

- *Modelo do Projeto* – Os modelos do CommonKADS compõem a especificação necessária para a criação de um sistema de conhecimento. O modelo do projeto conterá, então, a conversão das informações contidas nos modelos em especificações técnicas do sistema quanto a arquitetura, plataforma de implementação, módulos de softwares, construtores de representação, e mecanismos computacionais necessários para implementar as funções verificadas nos modelos de conhecimento e comunicação (Alkaim,2003).

2.1 Atividades de Modelagem

A aplicação dos modelos do CommonKADS na organização tem o objetivo de responder três questionamentos principais, através dos quais serão definidos os caminhos a serem seguidos e as soluções a serem escolhidas. Schreiber et al., 2002, apresentam estes questionamentos da seguinte forma:

a) Por quê? – Porque um sistema de conhecimento é uma solução em potencial? Para quais problemas? Que benefícios, custos e impactos organizacionais ele terá? O entendimento do ambiente e do contexto organizacional é o ponto mais importante deste questionamento.

b) Qual? – consiste em entender qual é a natureza e estrutura do conhecimento envolvido, bem como a natureza e estrutura de comunicação correspondente. Obter a descrição conceitual do conhecimento utilizado na realização de uma tarefa é um dos pontos chaves deste questionamento.

c) Como? – Como o conhecimento deve ser implementado no sistema computacional? Como deve ser a infra-estrutura tecnológica necessária para a construção e execução do sistema? Os aspectos técnicos da implementação são o principal foco neste questionamento.

3. O sistema proposto

Para se estabelecer as delimitações de um projeto é necessário que se tenha todo o conhecimento sobre o domínio do problema em questão e quais são os requisitos definidos para aplicação. Sendo assim os tópicos, a seguir, apresentarão as informações coletadas sobre o domínio do problema.

3.1 Descrição do Domínio do Problema

A aplicação da metodologia CommonKADS em projetos de Engenharia do Conhecimento resulta em

uma série de planilhas com análises descritivas. Normalmente estas planilhas são criadas em algum tipo de editor, como o MS Excel, por exemplo, o que gera alguns problemas e desconfortos:

- Aumento da dificuldade em visualizar de forma sistêmica as informações como um todo.
- Retrabalho na digitação de informações, já que algumas planilhas são refinamento de informações e utilizam dados que constam em outras planilhas.
- No caso de projetos com grande número de processos, tarefas, agentes e conhecimentos envolvidos, o número e tamanho das planilhas resultantes também será grande, com isso, a dificuldade em encontrar informações também aumentará.
- As informações coletadas através da aplicação do CommonKADS também compõe um tipo de conhecimento que, estando somente em planilhas, não poderá ser reaproveitado. Dessa forma, experiências em projetos anteriores ficam perdidas.

3.2 Sistema desenvolvido

Diante dos problemas descritos acima e dos requisitos coletados através dos estudos de casos e da literatura disponível, foi implementada uma ferramenta que possui a capacidade de armazenar as informações obtidas durante a aplicação do CommonKADS, em banco de dados, de forma a poder recuperá-las e analisá-las por vários prismas, a fim de obter-se maior agilidade na elaboração dos documentos e definição de soluções que possam ser viáveis e, cujo impacto, possa ser previsto de forma completa.

A implementação da interface deu-se seguindo o agrupamento de informações proposto na Metodologia CommonKADS, sendo assim foram criadas áreas para cada um dos modelos, e dentro destas áreas foram adicionadas as informações de cada uma das tabelas dos modelos. Conforme é apresentado nas figuras a seguir:

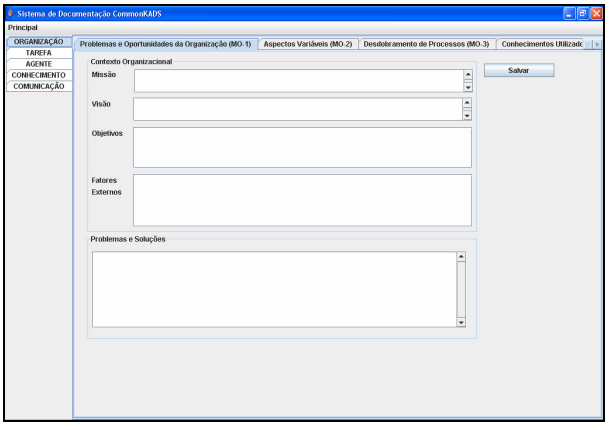

Figura 39: Interface principal do sistema.

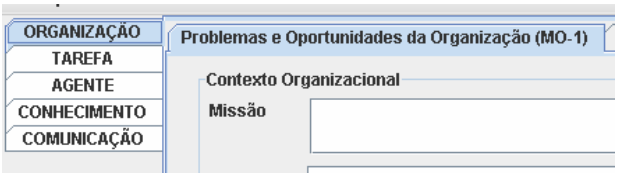

Figura 40: Detalhe das guias que representam cada modelo.

3.3 Testes

Para que o teste do sistema proposto no trabalho descrito neste artigo tivesse efetividade, foram coletados alguns estudos de casos contidos na literatura e as informações foram cadastradas no sistema. A seguir são apresentados dois estudos de caso que foram incluídos no sistema de forma a mostrar sua efetividade na documentação dos modelos do CommonKADS

O primeiro teste realizado no sistema foi feito com um estudo de caso apresentado por Schreiber et. al., 2002, sobre o uso do CommonKADS para verificar a viabilidade de se implementar a Gestão do Conhecimento nos negócios de sorvete da empresa Unilever, com relação ao desenvolvimento de produtos.

Este estudo de caso foi importante no teste do sistema, pois serviu como primeiro validador das interfaces e recursos disponíveis no sistema. O fato do estudo de caso apresentar um número considerável de tarefas dentro processo de desenvolvimento de produtos, contribuiu para verificar a efetividade do sistema na documentação de informações. Infelizmente o estudo de casos não foi apresentado na integra, não permitindo que todas as tarefas de conhecimento intensivo fossem detalhadas e registradas no sistema.

O segundo e mais completo teste do sistema foi realizado através do estudo de caso "Housing Application" apresentado por Schreiber et. al., 2002. Trata-se da análise de um problema cujo domínio é a alocação de casas de aluguel para pessoas que necessitam de uma residência. Após uma avaliação das pessoas inscritas o governo cede as casas para aqueles que melhor se enquadram no perfil de quem realmente precisa do benefício.

Com a aplicação dos testes no sistema foi possível constatar que existe a efetividade do registro das informações obtidas através da aplicação da metodologia CommonKADS. Contudo, pode-se perceber a necessidade de ter-se campos adicionais para registrar comentários sobre informações registradas nos campos tidos como "padrão" de cada modelo. Outro recurso necessário que foi verificado foi

a possibilidade de se anexar documentos externos, como planilhas, textos e imagens.

Certamente, através dos testes realizados pode-se perceber a facilidade que esta ferramenta pode proporcionar para o Engenheiro do Conhecimento no que diz respeito a registrar e recuperar as informações da análise. De forma geral, a fase de teste foi bem sucedida podendo-se considerar o sistema apto a registrar as informações dos modelos do CommonKADS.

4. Conclusões

A Metodologia CommonKADS mostra-se extremamente eficiente em projetos de Engenharia e Gestão do Conhecimento. Sua utilização e disseminação vem crescendo devido a grande popularização da Gestão do Conhecimento em ambientes corporativos.

A aplicação da metodologia CommonKADS em projetos de Engenharia e Gestão do Conhecimento ainda é trabalhosa pela falta de recursos que facilitem o registro e gerenciamento das informações coletadas durante a fase de análise.

O objetivo do trabalho descrito neste artigo foi criar uma ferramenta capaz de dirimir ao máximo o trabalho do Engenheiro do Conhecimento na documentação obtida com a análise. Através do estudo das características da metodologia CommonKADS e dos estudos de caso disponíveis na literatura proposta por Schreiber et al. 2002, foi possível fazer a especificação do projeto em UML definindo cada elemento informacional existente e os casos de uso envolvidos na tarefa de documentação das informações.

O desenvolvimento do sistema deu-se através de uma série de ciclos onde, ao final de cada ciclo, foram aplicadas as informações coletadas nos estudos de caso como forma de testar sua efetividade e possibilitar possíveis melhorias.

Os testes aplicados mostraram que o sistema proporciona ao Engenheiro do Conhecimento grande mobilidade tanto no registro de informações, onde é possível reaproveitar informações já cadastradas em outros modelos, como para recuperar os projetos cadastrados de maneira rápida e simples, permitindo ao Engenheiro visualizar suas experiências em projetos anteriores.

5. Referências

ALKAIM, João L. **Metodologia para Incorporar Conhecimento Intenso às Tarefas de Manutenção Centrada na Confiabilidade Aplicada em Ativos de Sistemas Elétricos.** 2003. Tese (Doutorado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, UFSC.

FREITAS JÚNIOR, Olival de Gusmão. **Um Modelo de Sistema de Gestão do Conhecimento para Grupos de Pesquisa e Desenvolvimento.** Florianópolis, 2003. 310 f. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis.

PACHECO, R. C. e SANTO, Neri dos**. Introdução à Engenharia e Gestão do Conhecimento**. 2004. Curso de Pós-Graduação em Engenharia e Gestão do Conhecimento, UFSC

SCHREIBER, G.; Akkermans, H.; Anjewierden, A.; Hoog, R.; Shadbolt, N.; de Velde, W. V.; and Wielinga, B.. **Knowledge Engineering and Management: the CommonKADS Methodology**. MIT Press. Cambridge. Massachussets. 2002

ANEXO 2 – Código Fonte

- Camada de Interface

```
/*
 * Main.java
 *
 * Created on 6 de Setembro de 2005, 21:06
 */

package commonkads;

import commonkads.interfaces.Principal;
import commonkads.logica.fachada.FabricaDeFachadaLogica;
import commonkads.logica.fachada.FachadaLogica;
import javax.swing.UIManager;
/**
 *
 * @author JR
 */
public class Main {

    /** Creates a new instance of Main */
    public Main() {

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {


        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
              try {


              UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
              }catch(Exception e){
                 System.out.println(e.getMessage());
              }
               FachadaLogica fachada = FabricaDeFachadaLogica.crie();

               new Principal(fachada).setVisible(true);
            }
        });

    }

}
```

```
/*
 * Projeto.java
 *
 * Created on 18 de Outubro de 2005, 03:13
 */

package commonkads.interfaces;

import commonkads.logica.fachada.FachadaLogica;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.List;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

/**
 *
 * @author  JR
 */
public class Projeto extends javax.swing.JDialog{
    FachadaLogica fachada;
    /** Creates new form Projeto */
    public Projeto() {
        initComponents();
        this.setSize(450, 200);
        this.setAlwaysOnTop(true);

    }

    public Projeto(FachadaLogica fachada,JFrame form) {
        super(form,true);
        initComponents();
        this.setSize(450, 200);
        this.setAlwaysOnTop(true);
        this.fachada = fachada;
        this.carregaDadosProjetos();
        }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {
        cmbProjetos = new javax.swing.JComboBox();
        jLabel1 = new javax.swing.JLabel();
        btnAbrir = new javax.swing.JButton();
        btnNovo = new javax.swing.JButton();

        getContentPane().setLayout(null);

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Sistema de Documenta\u00e7\u00e3o CommonKADS");
        setAlwaysOnTop(true);
        setLocationByPlatform(true);
        getContentPane().add(cmbProjetos);
        cmbProjetos.setBounds(10, 30, 370, 22);
```

```
    jLabel1.setText("Selecione o Projeto");
    getContentPane().add(jLabel1);
    jLabel1.setBounds(10, 10, 370, 14);

    btnAbrir.setText("Abrir Projeto");
    btnAbrir.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnAbrirActionPerformed(evt);
        }
    });

    getContentPane().add(btnAbrir);
    btnAbrir.setBounds(20, 110, 170, 23);

    btnNovo.setText("Novo Projeto");
    getContentPane().add(btnNovo);
    btnNovo.setBounds(230, 110, 190, 23);

    pack();
}

private void btnAbrirActionPerformed(java.awt.event.ActionEvent evt) {

    int index = cmbProjetos.getSelectedIndex();

    System.out.println(((commonkads.logica.modelo.Projeto)projetos.get(index)).getNome());
    fachada.setIdProjetoAtual(((commonkads.logica.modelo.Projeto)projetos.get(index)).getIdProjeto());
    //setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    this.setVisible(false);


}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Projeto().setVisible(true);
        }
    });
}

public void carregaDadosProjetos(){
    try {
        projetos = fachada.retorneProjetos();

       for (int i = 0; i < projetos.size();i++){
            cmbProjetos.addItem(((commonkads.logica.modelo.Projeto)projetos.get(i)).getNome());
        }


    } catch (ExcecaoBancoIndisponivel e){
        JOptionPane.showMessageDialog(this,e.getMessage(),"Atenção",JOptionPane.WARNING_MESSAGE);
```

```
        }


    }

    // Variables declaration - do not modify
    private javax.swing.JButton btnAbrir;
    private javax.swing.JButton btnNovo;
    private javax.swing.JComboBox cmbProjetos;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration
    private List projetos;
}


/*
 * Principal.java
 *
 * Created on 6 de Setembro de 2005, 21:09
 */

package commonkads.interfaces;

import commonkads.logica.fachada.FachadaLogica;
import commonkads.logica.modelo.Conhecimento;
import commonkads.logica.modelo.Organizacao;
import commonkads.logica.modelo.Processo;
import commonkads.logica.modelo.Recurso;
import commonkads.logica.modelo.Tarefa;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.awt.FileDialog;
import java.util.List;
import javax.swing.DefaultListModel;
import javax.swing.ImageIcon;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.ListModel;




/**
 *
 * @author  JR
 */
public class Principal extends javax.swing.JFrame {
    FachadaLogica fachada;

    /** Creates new form Principal */

    public Principal() {
        initComponents();

        this.setExtendedState(this.MAXIMIZED_BOTH);

    }
```

```
public Principal(FachadaLogica fachada) {

    initComponents();

    this.setExtendedState(this.MAXIMIZED_BOTH);
    this.fachada = fachada;

    new commonkads.interfaces.Projeto(fachada,this).setVisible(true);

    carregaDadosProjeto();

}


/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
private void initComponents() {
    javax.swing.JComboBox jComboBox3;

    jPanel1 = new javax.swing.JPanel();
    jTabbedPane1 = new javax.swing.JTabbedPane();
    jPanel2 = new javax.swing.JPanel();
    jTabbedPane2 = new javax.swing.JTabbedPane();
    jPanel3 = new javax.swing.JPanel();
    jPanel11 = new javax.swing.JPanel();
    jScrollPane8 = new javax.swing.JScrollPane();
    edtProblemas = new javax.swing.JEditorPane();
    btnSalvar = new javax.swing.JButton();
    jPanel14 = new javax.swing.JPanel();
    jScrollPane7 = new javax.swing.JScrollPane();
    edtFatoresExternos = new javax.swing.JEditorPane();
    jScrollPane6 = new javax.swing.JScrollPane();
    edtObjetivos = new javax.swing.JTextPane();
    jScrollPane5 = new javax.swing.JScrollPane();
    edtVisao = new javax.swing.JEditorPane();
    jScrollPane4 = new javax.swing.JScrollPane();
    edtMissao = new javax.swing.JEditorPane();
    lblMissao3 = new javax.swing.JLabel();
    lblMissao1 = new javax.swing.JLabel();
    lblObjetivos = new javax.swing.JLabel();
    lblVisao = new javax.swing.JLabel();
    lblMissao = new javax.swing.JLabel();
    jPanel4 = new javax.swing.JPanel();
    panelEstrutura = new javax.swing.JPanel();
    btnVisualizaEstrutura = new javax.swing.JButton();
    btnAdicionarEstrutura = new javax.swing.JButton();
    txtEstrutura = new javax.swing.JTextField();
    jLabel30 = new javax.swing.JLabel();
    panelProcessos = new javax.swing.JPanel();
    btnAdicionarProcesso = new javax.swing.JButton();
    jScrollPane2 = new javax.swing.JScrollPane();
    listaProcessos = new JList(new DefaultListModel());
```

```
btnAdicionarProcesso1 = new javax.swing.JButton();
panelPessoas = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
edtPessoas = new javax.swing.JEditorPane();
panelRecursos = new javax.swing.JPanel();
jScrollPane3 = new javax.swing.JScrollPane();
listaRecursos = new javax.swing.JList();
btnAdicionarRecurso = new javax.swing.JButton();
panelPessoas1 = new javax.swing.JPanel();
jScrollPane9 = new javax.swing.JScrollPane();
edtCulturaPoder = new javax.swing.JEditorPane();
panelRecursos1 = new javax.swing.JPanel();
jScrollPane10 = new javax.swing.JScrollPane();
listaConhecimentosMO2 = new javax.swing.JList();
btnAdicionarConhecimento = new javax.swing.JButton();
jPanel5 = new javax.swing.JPanel();
jLabel3 = new javax.swing.JLabel();
cmbProcessosMO3 = new javax.swing.JComboBox();
panelRecursos2 = new javax.swing.JPanel();
jLabel4 = new javax.swing.JLabel();
jTextField1 = new javax.swing.JTextField();
txtExecutadaPor = new javax.swing.JTextField();
jLabel5 = new javax.swing.JLabel();
txtLocal = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();
panelRecursos3 = new javax.swing.JPanel();
jScrollPane11 = new javax.swing.JScrollPane();
listaConhecimentosMO3 = new javax.swing.JList();
btnAdicionarConhecimentoMO3 = new javax.swing.JButton();
chkTarefaIntensiva = new javax.swing.JCheckBox();
spnGrauSignificancia = new javax.swing.JSpinner();
jLabel7 = new javax.swing.JLabel();
btnAdicionarTarefaMO3 = new javax.swing.JButton();
jLabel31 = new javax.swing.JLabel();
cmbTarefasMO4 = new javax.swing.JComboBox();
jPanel6 = new javax.swing.JPanel();
jLabel8 = new javax.swing.JLabel();
jComboBox2 = new javax.swing.JComboBox();
btnAdicionarProcesso7 = new javax.swing.JButton();
panelRecursos4 = new javax.swing.JPanel();
jTextField4 = new javax.swing.JTextField();
jLabel9 = new javax.swing.JLabel();
jTextField5 = new javax.swing.JTextField();
jLabel10 = new javax.swing.JLabel();
panelRecursos5 = new javax.swing.JPanel();
jScrollPane12 = new javax.swing.JScrollPane();
jEditorPane3 = new javax.swing.JEditorPane();
jCheckBox2 = new javax.swing.JCheckBox();
jCheckBox3 = new javax.swing.JCheckBox();
jCheckBox4 = new javax.swing.JCheckBox();
jCheckBox5 = new javax.swing.JCheckBox();
jPanel16 = new javax.swing.JPanel();
jTabbedPane4 = new javax.swing.JTabbedPane();
jPanel17 = new javax.swing.JPanel();
jScrollPane13 = new javax.swing.JScrollPane();
jEditorPane4 = new javax.swing.JEditorPane();
```

```
jEditorPane5 = new javax.swing.JEditorPane();
jScrollPane14 = new javax.swing.JScrollPane();
jEditorPane6 = new javax.swing.JEditorPane();
jEditorPane7 = new javax.swing.JEditorPane();
jPanel18 = new javax.swing.JPanel();
jScrollPane15 = new javax.swing.JScrollPane();
jEditorPane8 = new javax.swing.JEditorPane();
jEditorPane9 = new javax.swing.JEditorPane();
jPanel19 = new javax.swing.JPanel();
jScrollPane16 = new javax.swing.JScrollPane();
jEditorPane10 = new javax.swing.JEditorPane();
jEditorPane11 = new javax.swing.JEditorPane();
jPanel20 = new javax.swing.JPanel();
jScrollPane17 = new javax.swing.JScrollPane();
jEditorPane12 = new javax.swing.JEditorPane();
jEditorPane13 = new javax.swing.JEditorPane();
jPanel7 = new javax.swing.JPanel();
jTabbedPane3 = new javax.swing.JTabbedPane();
jPanel12 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();
cmbTarefa = new javax.swing.JComboBox();
cmbTarefa.addItem("Avaliação da Aplicação");
jButton1 = new javax.swing.JButton();
panelEstrutura1 = new javax.swing.JPanel();
jLabel2 = new javax.swing.JLabel();
cmbProcesso = new javax.swing.JComboBox();
cmbProcesso.addItem("Processo Primário");
jButton2 = new javax.swing.JButton();
jScrollPane19 = new javax.swing.JScrollPane();
jEditorPane14 = new javax.swing.JEditorPane();
jLabel11 = new javax.swing.JLabel();
panelEstrutura2 = new javax.swing.JPanel();
jScrollPane20 = new javax.swing.JScrollPane();
jEditorPane15 = new javax.swing.JEditorPane();
panelEstrutura3 = new javax.swing.JPanel();
jScrollPane21 = new javax.swing.JScrollPane();
jEditorPane16 = new javax.swing.JEditorPane();
panelEstrutura4 = new javax.swing.JPanel();
jScrollPane22 = new javax.swing.JScrollPane();
jEditorPane17 = new javax.swing.JEditorPane();
panelEstrutura5 = new javax.swing.JPanel();
jScrollPane24 = new javax.swing.JScrollPane();
jEditorPane20 = new javax.swing.JEditorPane();
panelEstrutura6 = new javax.swing.JPanel();
jScrollPane23 = new javax.swing.JScrollPane();
jEditorPane18 = new javax.swing.JEditorPane();
panelEstrutura7 = new javax.swing.JPanel();
jScrollPane26 = new javax.swing.JScrollPane();
jEditorPane22 = new javax.swing.JEditorPane();
panelRecursos7 = new javax.swing.JPanel();
jScrollPane25 = new javax.swing.JScrollPane();
listaRecursos4 = new javax.swing.JList();
btnAdicionarRecurso4 = new javax.swing.JButton();
panelEstrutura8 = new javax.swing.JPanel();
jScrollPane27 = new javax.swing.JScrollPane();
jEditorPane23 = new javax.swing.JEditorPane();
```

```
panelEstrutura9 = new javax.swing.JPanel();
jScrollPane28 = new javax.swing.JScrollPane();
jEditorPane19 = new javax.swing.JEditorPane();
panelEstrutura10 = new javax.swing.JPanel();
jScrollPane30 = new javax.swing.JScrollPane();
jEditorPane25 = new javax.swing.JEditorPane();
jPanel13 = new javax.swing.JPanel();
jPanel21 = new javax.swing.JPanel();
jPanel22 = new javax.swing.JPanel();
jLabel12 = new javax.swing.JLabel();
jComboBox3 = new javax.swing.JComboBox();
jComboBox3.addItem("Critério de Avaliação");
btnAdicionarProcesso9 = new javax.swing.JButton();
btnAdicionarProcesso10 = new javax.swing.JButton();
jTextField6 = new javax.swing.JTextField();
jLabel13 = new javax.swing.JLabel();
jTextField7 = new javax.swing.JTextField();
jLabel14 = new javax.swing.JLabel();
jTextField8 = new javax.swing.JTextField();
jLabel15 = new javax.swing.JLabel();
jPanel23 = new javax.swing.JPanel();
jPanel24 = new javax.swing.JPanel();
jPanel25 = new javax.swing.JPanel();
jPanel27 = new javax.swing.JPanel();
jCheckBox6 = new javax.swing.JCheckBox();
jCheckBox7 = new javax.swing.JCheckBox();
jCheckBox8 = new javax.swing.JCheckBox();
jCheckBox9 = new javax.swing.JCheckBox();
jCheckBox10 = new javax.swing.JCheckBox();
jCheckBox11 = new javax.swing.JCheckBox();
jCheckBox12 = new javax.swing.JCheckBox();
jCheckBox13 = new javax.swing.JCheckBox();
jCheckBox14 = new javax.swing.JCheckBox();
jCheckBox15 = new javax.swing.JCheckBox();
jCheckBox16 = new javax.swing.JCheckBox();
jPanel28 = new javax.swing.JPanel();
jPanel30 = new javax.swing.JPanel();
jCheckBox17 = new javax.swing.JCheckBox();
jCheckBox18 = new javax.swing.JCheckBox();
jCheckBox19 = new javax.swing.JCheckBox();
jCheckBox20 = new javax.swing.JCheckBox();
jCheckBox21 = new javax.swing.JCheckBox();
jCheckBox22 = new javax.swing.JCheckBox();
jCheckBox23 = new javax.swing.JCheckBox();
jCheckBox24 = new javax.swing.JCheckBox();
jCheckBox25 = new javax.swing.JCheckBox();
jCheckBox26 = new javax.swing.JCheckBox();
jCheckBox27 = new javax.swing.JCheckBox();
jLabel19 = new javax.swing.JLabel();
jLabel20 = new javax.swing.JLabel();
jPanel26 = new javax.swing.JPanel();
jPanel29 = new javax.swing.JPanel();
jCheckBox28 = new javax.swing.JCheckBox();
jCheckBox29 = new javax.swing.JCheckBox();
jCheckBox30 = new javax.swing.JCheckBox();
jCheckBox31 = new javax.swing.JCheckBox();
```

```
jCheckBox32 = new javax.swing.JCheckBox();
jPanel31 = new javax.swing.JPanel();
jPanel32 = new javax.swing.JPanel();
jCheckBox39 = new javax.swing.JCheckBox();
jCheckBox40 = new javax.swing.JCheckBox();
jCheckBox41 = new javax.swing.JCheckBox();
jCheckBox42 = new javax.swing.JCheckBox();
jCheckBox43 = new javax.swing.JCheckBox();
jLabel21 = new javax.swing.JLabel();
jLabel22 = new javax.swing.JLabel();
jPanel33 = new javax.swing.JPanel();
jPanel34 = new javax.swing.JPanel();
jCheckBox33 = new javax.swing.JCheckBox();
jCheckBox34 = new javax.swing.JCheckBox();
jCheckBox35 = new javax.swing.JCheckBox();
jCheckBox36 = new javax.swing.JCheckBox();
jCheckBox37 = new javax.swing.JCheckBox();
jPanel35 = new javax.swing.JPanel();
jPanel36 = new javax.swing.JPanel();
jCheckBox44 = new javax.swing.JCheckBox();
jCheckBox45 = new javax.swing.JCheckBox();
jCheckBox46 = new javax.swing.JCheckBox();
jCheckBox47 = new javax.swing.JCheckBox();
jCheckBox48 = new javax.swing.JCheckBox();
jLabel23 = new javax.swing.JLabel();
jLabel24 = new javax.swing.JLabel();
jPanel8 = new javax.swing.JPanel();
jTabbedPane5 = new javax.swing.JTabbedPane();
jPanel37 = new javax.swing.JPanel();
jLabel16 = new javax.swing.JLabel();
jComboBox4 = new javax.swing.JComboBox();
btnAdicionarProcesso11 = new javax.swing.JButton();
jPanel39 = new javax.swing.JPanel();
jScrollPane18 = new javax.swing.JScrollPane();
jTextPane1 = new javax.swing.JTextPane();
jPanel40 = new javax.swing.JPanel();
jScrollPane29 = new javax.swing.JScrollPane();
jTextPane2 = new javax.swing.JTextPane();
jPanel41 = new javax.swing.JPanel();
jScrollPane31 = new javax.swing.JScrollPane();
jTextPane3 = new javax.swing.JTextPane();
jPanel42 = new javax.swing.JPanel();
jScrollPane32 = new javax.swing.JScrollPane();
jTextPane4 = new javax.swing.JTextPane();
panelRecursos6 = new javax.swing.JPanel();
jScrollPane33 = new javax.swing.JScrollPane();
listaRecursos3 = new javax.swing.JList();
btnAdicionarRecurso3 = new javax.swing.JButton();
panelRecursos8 = new javax.swing.JPanel();
jScrollPane34 = new javax.swing.JScrollPane();
listaRecursos5 = new javax.swing.JList();
btnAdicionarRecurso5 = new javax.swing.JButton();
panelRecursos9 = new javax.swing.JPanel();
jScrollPane35 = new javax.swing.JScrollPane();
listaRecursos6 = new javax.swing.JList();
btnAdicionarRecurso6 = new javax.swing.JButton();
```

```
jPanel38 = new javax.swing.JPanel();
jTabbedPane6 = new javax.swing.JTabbedPane();
jPanel43 = new javax.swing.JPanel();
jTextPane6 = new javax.swing.JTextPane();
jPanel44 = new javax.swing.JPanel();
jScrollPane36 = new javax.swing.JScrollPane();
jTextPane5 = new javax.swing.JTextPane();
jPanel45 = new javax.swing.JPanel();
jTextPane7 = new javax.swing.JTextPane();
jPanel46 = new javax.swing.JPanel();
jTextPane8 = new javax.swing.JTextPane();
jPanel9 = new javax.swing.JPanel();
jLabel17 = new javax.swing.JLabel();
cmbTarefa1 = new javax.swing.JComboBox();
jButton3 = new javax.swing.JButton();
jPanel47 = new javax.swing.JPanel();
jScrollPane37 = new javax.swing.JScrollPane();
jTextPane9 = new javax.swing.JTextPane();
jPanel10 = new javax.swing.JPanel();
jTabbedPane7 = new javax.swing.JTabbedPane();
jPanel48 = new javax.swing.JPanel();
jLabel18 = new javax.swing.JLabel();
cmbTarefa2 = new javax.swing.JComboBox();
jButton4 = new javax.swing.JButton();
jPanel50 = new javax.swing.JPanel();
jScrollPane38 = new javax.swing.JScrollPane();
jTextPane10 = new javax.swing.JTextPane();
panelRecursos10 = new javax.swing.JPanel();
jLabel25 = new javax.swing.JLabel();
cmbTarefa3 = new javax.swing.JComboBox();
jLabel26 = new javax.swing.JLabel();
cmbTarefa4 = new javax.swing.JComboBox();
jPanel51 = new javax.swing.JPanel();
jScrollPane39 = new javax.swing.JScrollPane();
jTextPane11 = new javax.swing.JTextPane();
jPanel52 = new javax.swing.JPanel();
jScrollPane40 = new javax.swing.JScrollPane();
jTextPane12 = new javax.swing.JTextPane();
jPanel53 = new javax.swing.JPanel();
jScrollPane41 = new javax.swing.JScrollPane();
jTextPane13 = new javax.swing.JTextPane();
jPanel49 = new javax.swing.JPanel();
cmbTarefa5 = new javax.swing.JComboBox();
jLabel27 = new javax.swing.JLabel();
jButton5 = new javax.swing.JButton();
panelRecursos11 = new javax.swing.JPanel();
jLabel28 = new javax.swing.JLabel();
cmbTarefa6 = new javax.swing.JComboBox();
jLabel29 = new javax.swing.JLabel();
cmbTarefa7 = new javax.swing.JComboBox();
panelRecursos12 = new javax.swing.JPanel();
jScrollPane42 = new javax.swing.JScrollPane();
listaRecursos7 = new javax.swing.JList();
btnAdicionarRecurso7 = new javax.swing.JButton();
jMenuBar1 = new javax.swing.JMenuBar();
menuBar = new javax.swing.JMenu();
```

```
        abrir = new javax.swing.JMenuItem();
        novo = new javax.swing.JMenuItem();

        setTitle("Sistema de Documenta\u00e7\u00e3o CommonKADS");
        setBackground(new java.awt.Color(255, 255, 255));
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setFocusable(false);
        jPanel1.setLayout(new java.awt.BorderLayout());

        getContentPane().add(jPanel1, java.awt.BorderLayout.WEST);

        jTabbedPane1.setBackground(new java.awt.Color(255, 255, 255));
        jTabbedPane1.setTabLayoutPolicy(javax.swing.JTabbedPane.SCROLL_TAB_LAYOUT);
        jTabbedPane1.setTabPlacement(javax.swing.JTabbedPane.LEFT);
        jPanel2.setLayout(new java.awt.BorderLayout());

        jTabbedPane2.setTabLayoutPolicy(javax.swing.JTabbedPane.SCROLL_TAB_LAYOUT);
        jPanel3.setLayout(null);

        jPanel11.setLayout(null);

        jPanel11.setBorder(new javax.swing.border.TitledBorder("Problemas e Solu\u00e7\u00f5es"));

jScrollPane8.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEV
ER);

jScrollPane8.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
        jScrollPane8.setViewportView(edtProblemas);

        jPanel11.add(jScrollPane8);
        jScrollPane8.setBounds(10, 30, 640, 130);

        jPanel3.add(jPanel11);
        jPanel11.setBounds(20, 310, 670, 180);

        btnSalvar.setText("Salvar");
        btnSalvar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnSalvarActionPerformed(evt);
            }
        });

        jPanel3.add(btnSalvar);
        btnSalvar.setBounds(700, 20, 120, 23);

        jPanel14.setLayout(null);

        jPanel14.setBorder(new javax.swing.border.TitledBorder("Contexto Organizacional"));
        jScrollPane7.setViewportView(edtFatoresExternos);

        jPanel14.add(jScrollPane7);
        jScrollPane7.setBounds(80, 200, 580, 90);

        jScrollPane6.setViewportView(edtObjetivos);

        jPanel14.add(jScrollPane6);
```

```
        jScrollPane6.setBounds(80, 120, 580, 70);


jScrollPane5.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEV
ER);

jScrollPane5.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
        jScrollPane5.setViewportView(edtVisao);

        jPanel14.add(jScrollPane5);
        jScrollPane5.setBounds(80, 70, 580, 40);


jScrollPane4.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEV
ER);

jScrollPane4.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
        jScrollPane4.setViewportView(edtMissao);

        jPanel14.add(jScrollPane4);
        jScrollPane4.setBounds(82, 22, 580, 40);

        lblMissao3.setText("Externos");
        jPanel14.add(lblMissao3);
        lblMissao3.setBounds(10, 220, 80, 20);

        lblMissao1.setText("Fatores");
        jPanel14.add(lblMissao1);
        lblMissao1.setBounds(10, 200, 50, 20);

        lblObjetivos.setText("Objetivos");
        jPanel14.add(lblObjetivos);
        lblObjetivos.setBounds(10, 120, 70, 20);

        lblVisao.setText("Vis\u00e3o");
        jPanel14.add(lblVisao);
        lblVisao.setBounds(10, 70, 50, 20);

        lblMissao.setText("Miss\u00e3o");
        jPanel14.add(lblMissao);
        lblMissao.setBounds(10, 20, 50, 20);

        jPanel3.add(jPanel14);
        jPanel14.setBounds(20, 8, 670, 300);

        jTabbedPane2.addTab("Problemas e Oportunidades da Organiza\u00e7\u00e3o (MO-1)", jPanel3);

        jPanel4.setLayout(null);

        panelEstrutura.setLayout(null);

        panelEstrutura.setBorder(new javax.swing.border.TitledBorder("Estrutura"));
        btnVisualizaEstrutura.setActionCommand("Adicionar Processo");
        btnVisualizaEstrutura.setLabel("Visualizar");
        btnVisualizaEstrutura.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        btnVisualizaEstruturaActionPerformed(evt);
    }
});

panelEstrutura.add(btnVisualizaEstrutura);
btnVisualizaEstrutura.setBounds(280, 70, 100, 23);

btnAdicionarEstrutura.setActionCommand("Adicionar Processo");
btnAdicionarEstrutura.setLabel("Adicionar");
btnAdicionarEstrutura.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnAdicionarEstruturaActionPerformed(evt);
    }
});

panelEstrutura.add(btnAdicionarEstrutura);
btnAdicionarEstrutura.setBounds(280, 40, 100, 23);

panelEstrutura.add(txtEstrutura);
txtEstrutura.setBounds(10, 60, 240, 19);

jLabel30.setText("Arquivo que representa a estrutura a empresa:");
panelEstrutura.add(jLabel30);
jLabel30.setBounds(10, 40, 270, 14);

jPanel4.add(panelEstrutura);
panelEstrutura.setBounds(0, 0, 390, 122);

panelProcessos.setLayout(null);

panelProcessos.setBorder(new javax.swing.border.TitledBorder("Processos"));
btnAdicionarProcesso.setActionCommand("Adicionar Processo");
btnAdicionarProcesso.setLabel("Adicionar");
btnAdicionarProcesso.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnAdicionarProcessoActionPerformed(evt);
    }
});

panelProcessos.add(btnAdicionarProcesso);
btnAdicionarProcesso.setBounds(320, 20, 100, 23);

listaProcessos.setToolTipText("Lista de Processos da Organiza\u00e7\u00e3o");
listaProcessos.setVisibleRowCount(5);
jScrollPane2.setViewportView(listaProcessos);
listaProcessos.getAccessibleContext().setAccessibleName("listaProcessos");

panelProcessos.add(jScrollPane2);
jScrollPane2.setBounds(30, 20, 280, 90);

btnAdicionarProcesso1.setActionCommand("Adicionar Processo");
btnAdicionarProcesso1.setLabel("Visualizar");
panelProcessos.add(btnAdicionarProcesso1);
btnAdicionarProcesso1.setBounds(320, 50, 100, 23);

jPanel4.add(panelProcessos);
```

```
panelProcessos.setBounds(400, 0, 430, 122);

panelPessoas.setLayout(new java.awt.BorderLayout());

panelPessoas.setBorder(new javax.swing.border.TitledBorder("Pessoas  (indique as pessoas envolvidas)"));
jScrollPane1.setViewportView(edtPessoas);

panelPessoas.add(jScrollPane1, java.awt.BorderLayout.CENTER);

jPanel4.add(panelPessoas);
panelPessoas.setBounds(0, 130, 390, 122);

panelRecursos.setLayout(null);

panelRecursos.setBorder(new javax.swing.border.TitledBorder("Recursos"));
jScrollPane3.setViewportView(listaRecursos);

panelRecursos.add(jScrollPane3);
jScrollPane3.setBounds(30, 20, 280, 90);

btnAdicionarRecurso.setLabel("Adicionar");
btnAdicionarRecurso.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      btnAdicionarRecursoActionPerformed(evt);
   }
});

panelRecursos.add(btnAdicionarRecurso);
btnAdicionarRecurso.setBounds(320, 20, 100, 23);

jPanel4.add(panelRecursos);
panelRecursos.setBounds(400, 130, 430, 122);

panelPessoas1.setLayout(new java.awt.BorderLayout());

panelPessoas1.setBorder(new javax.swing.border.TitledBorder("Cultura & Poder"));
jScrollPane9.setViewportView(edtCulturaPoder);

panelPessoas1.add(jScrollPane9, java.awt.BorderLayout.CENTER);

jPanel4.add(panelPessoas1);
panelPessoas1.setBounds(0, 390, 830, 200);

panelRecursos1.setLayout(null);

panelRecursos1.setBorder(new javax.swing.border.TitledBorder("Conhecimento"));
jScrollPane10.setViewportView(listaConhecimentosMO2);

panelRecursos1.add(jScrollPane10);
jScrollPane10.setBounds(30, 20, 610, 90);

btnAdicionarConhecimento.setLabel("Adicionar");
btnAdicionarConhecimento.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      btnAdicionarConhecimentoActionPerformed(evt);
   }
```

```
});

panelRecursos1.add(btnAdicionarConhecimento);
btnAdicionarConhecimento.setBounds(650, 20, 100, 23);

jPanel4.add(panelRecursos1);
panelRecursos1.setBounds(0, 260, 830, 122);

jTabbedPane2.addTab("Aspectos Vari\u00e1veis (MO-2)", jPanel4);

jPanel5.setLayout(null);

jLabel3.setText("Processo");
jPanel5.add(jLabel3);
jLabel3.setBounds(10, 20, 70, 15);

cmbProcessosMO3.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        cmbProcessosMO3ItemStateChanged(evt);
    }
});

jPanel5.add(cmbProcessosMO3);
cmbProcessosMO3.setBounds(70, 10, 280, 22);

panelRecursos2.setLayout(null);

panelRecursos2.setBorder(new javax.swing.border.TitledBorder("Detalhes da Tarefa"));
jLabel4.setText("Tarefa");
panelRecursos2.add(jLabel4);
jLabel4.setBounds(10, 30, 70, 15);

jTextField1.setEditable(false);
jTextField1.setFont(new java.awt.Font("Microsoft Sans Serif", 1, 11));
panelRecursos2.add(jTextField1);
jTextField1.setBounds(90, 29, 260, 20);

panelRecursos2.add(txtExecutadaPor);
txtExecutadaPor.setBounds(90, 70, 260, 19);

jLabel5.setText("Executada por");
panelRecursos2.add(jLabel5);
jLabel5.setBounds(10, 70, 100, 15);

panelRecursos2.add(txtLocal);
txtLocal.setBounds(90, 100, 260, 20);

jLabel6.setText("Local");
panelRecursos2.add(jLabel6);
jLabel6.setBounds(10, 100, 70, 15);

panelRecursos3.setLayout(null);

panelRecursos3.setBorder(new javax.swing.border.TitledBorder("Conhecimentos Utilizados"));
jScrollPane11.setViewportView(listaConhecimentosMO3);
```

```
panelRecursos3.add(jScrollPane11);
jScrollPane11.setBounds(30, 20, 450, 90);

btnAdicionarConhecimentoMO3.setText("Adicionar Conhecimento");
panelRecursos3.add(btnAdicionarConhecimentoMO3);
btnAdicionarConhecimentoMO3.setBounds(490, 20, 210, 23);

panelRecursos2.add(panelRecursos3);
panelRecursos3.setBounds(10, 140, 710, 122);

chkTarefaIntensiva.setText("Tarefa Intensiva em Conhecimento");
panelRecursos2.add(chkTarefaIntensiva);
chkTarefaIntensiva.setBounds(370, 30, 270, 23);

panelRecursos2.add(spnGrauSignificancia);
spnGrauSignificancia.setBounds(520, 60, 40, 30);

jLabel7.setText("Grau de Signific\u00e2ncia");
panelRecursos2.add(jLabel7);
jLabel7.setBounds(370, 70, 140, 15);

jPanel5.add(panelRecursos2);
panelRecursos2.setBounds(10, 80, 780, 320);

btnAdicionarTarefaMO3.setText("Incluir Tarefa");
btnAdicionarTarefaMO3.setActionCommand("NovaTarefa");
btnAdicionarTarefaMO3.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      btnAdicionarTarefaMO3ActionPerformed(evt);
   }
});

jPanel5.add(btnAdicionarTarefaMO3);
btnAdicionarTarefaMO3.setBounds(360, 40, 150, 23);

jLabel31.setText("Tarefa");
jPanel5.add(jLabel31);
jLabel31.setBounds(10, 50, 180, 15);

cmbTarefasMO4.addItemListener(new java.awt.event.ItemListener() {
   public void itemStateChanged(java.awt.event.ItemEvent evt) {
      cmbTarefasMO4ItemStateChanged(evt);
   }
});

jPanel5.add(cmbTarefasMO4);
cmbTarefasMO4.setBounds(70, 40, 280, 22);

jTabbedPane2.addTab("Desdobramento de Processos (MO-3)", jPanel5);

jPanel6.setLayout(null);

jLabel8.setText("Conhecimento");
jPanel6.add(jLabel8);
jLabel8.setBounds(10, 20, 110, 15);
```

```
jPanel6.add(jComboBox2);
jComboBox2.setBounds(120, 10, 250, 22);

btnAdicionarProcesso7.setText("Incluir Conhecimento");
btnAdicionarProcesso7.setActionCommand("NovaTarefa");
jPanel6.add(btnAdicionarProcesso7);
btnAdicionarProcesso7.setBounds(390, 10, 190, 23);

panelRecursos4.setLayout(null);

panelRecursos4.setBorder(new javax.swing.border.TitledBorder("Detalhes sobre o Conhecimento Utilizado"));
panelRecursos4.add(jTextField4);
jTextField4.setBounds(90, 29, 260, 20);

jLabel9.setText("Quem possui");
panelRecursos4.add(jLabel9);
jLabel9.setBounds(10, 30, 80, 15);

panelRecursos4.add(jTextField5);
jTextField5.setBounds(90, 70, 260, 19);

jLabel10.setText("Usado em");
panelRecursos4.add(jLabel10);
jLabel10.setBounds(10, 70, 100, 15);

panelRecursos5.setLayout(null);

panelRecursos5.setBorder(new javax.swing.border.TitledBorder("Coment\u00e1rios"));
jScrollPane12.setViewportView(jEditorPane3);

panelRecursos5.add(jScrollPane12);
jScrollPane12.setBounds(20, 30, 560, 80);

panelRecursos4.add(panelRecursos5);
panelRecursos5.setBounds(10, 190, 600, 122);

jCheckBox2.setText("Usado da forma correta");
panelRecursos4.add(jCheckBox2);
jCheckBox2.setBounds(30, 100, 160, 23);

jCheckBox3.setText("Usado no lugar certo");
panelRecursos4.add(jCheckBox3);
jCheckBox3.setBounds(210, 100, 180, 23);

jCheckBox4.setText("Usando no tempo certo");
panelRecursos4.add(jCheckBox4);
jCheckBox4.setBounds(390, 100, 210, 23);

jCheckBox5.setText("Usando com a qualidade certa");
panelRecursos4.add(jCheckBox5);
jCheckBox5.setBounds(30, 140, 260, 23);

jPanel6.add(panelRecursos4);
panelRecursos4.setBounds(10, 50, 780, 360);

jTabbedPane2.addTab("Conhecimentos Utilizados (MO-4)", jPanel6);
```

```
jPanel16.setLayout(new java.awt.BorderLayout());
jPanel17.setLayout(new java.awt.BorderLayout());
jScrollPane13.setViewportView(jEditorPane4);
jPanel17.add(jScrollPane13, java.awt.BorderLayout.CENTER);
jPanel17.add(jEditorPane5, java.awt.BorderLayout.NORTH);
jScrollPane14.setViewportView(jEditorPane6);
jPanel17.add(jScrollPane14, java.awt.BorderLayout.CENTER);
jPanel17.add(jEditorPane7, java.awt.BorderLayout.SOUTH);
jTabbedPane4.addTab("Viabilidade de Neg\u00f3cios", jPanel17);
jPanel18.setLayout(new java.awt.BorderLayout());
jScrollPane15.setViewportView(jEditorPane8);
jPanel18.add(jScrollPane15, java.awt.BorderLayout.CENTER);
jPanel18.add(jEditorPane9, java.awt.BorderLayout.NORTH);
jTabbedPane4.addTab("Viabilidade T\u00e9cnica", jPanel18);
jPanel19.setLayout(new java.awt.BorderLayout());
jScrollPane16.setViewportView(jEditorPane10);
jPanel19.add(jScrollPane16, java.awt.BorderLayout.CENTER);
jPanel19.add(jEditorPane11, java.awt.BorderLayout.NORTH);
jTabbedPane4.addTab("Viabilidade de Projeto", jPanel19);
jPanel20.setLayout(new java.awt.BorderLayout());
jScrollPane17.setViewportView(jEditorPane12);
jPanel20.add(jScrollPane17, java.awt.BorderLayout.CENTER);
jPanel20.add(jEditorPane13, java.awt.BorderLayout.NORTH);
jTabbedPane4.addTab("A\u00e7\u00f5es Propostas", jPanel20);
jPanel16.add(jTabbedPane4, java.awt.BorderLayout.CENTER);
jTabbedPane2.addTab("Documento de Decis\u00e3o de Viabilidade (MO-5)", jPanel16);
jPanel2.add(jTabbedPane2, java.awt.BorderLayout.CENTER);
jTabbedPane1.addTab("ORGANIZA\u00c7\u00c3O", jPanel2);
```

```
jPanel7.setLayout(new java.awt.BorderLayout());

jPanel12.setLayout(null);

jLabel1.setText("Tarefa");
jLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
jPanel12.add(jLabel1);
jLabel1.setBounds(5, 9, 70, 14);

jPanel12.add(cmbTarefa);
cmbTarefa.setBounds(70, 0, 260, 22);

jButton1.setText("Incluir Tarefa");
jPanel12.add(jButton1);
jButton1.setBounds(340, 0, 140, 23);

panelEstrutura1.setLayout(null);

panelEstrutura1.setBorder(new javax.swing.border.TitledBorder("Organiza\u00e7\u00e3o"));
jLabel2.setText("Processo");
panelEstrutura1.add(jLabel2);
jLabel2.setBounds(14, 30, 80, 14);

panelEstrutura1.add(cmbProcesso);
cmbProcesso.setBounds(100, 30, 220, 22);

jButton2.setText("Incluir Processo");
panelEstrutura1.add(jButton2);
jButton2.setBounds(340, 30, 130, 23);

jScrollPane19.setViewportView(jEditorPane14);

panelEstrutura1.add(jScrollPane19);
jScrollPane19.setBounds(100, 60, 440, 50);

jLabel11.setText("Coment\u00e1rios");
panelEstrutura1.add(jLabel11);
jLabel11.setBounds(10, 70, 90, 14);

jPanel12.add(panelEstrutura1);
panelEstrutura1.setBounds(0, 40, 790, 122);

panelEstrutura2.setLayout(null);

panelEstrutura2.setBorder(new javax.swing.border.TitledBorder("Objetivos e Valores"));
jScrollPane20.setViewportView(jEditorPane15);

panelEstrutura2.add(jScrollPane20);
jScrollPane20.setBounds(10, 20, 360, 90);

jPanel12.add(panelEstrutura2);
panelEstrutura2.setBounds(0, 170, 430, 122);

panelEstrutura3.setLayout(null);
```

```
panelEstrutura3.setBorder(new javax.swing.border.TitledBorder("Depend\u00eancia de Fluxo"));
jScrollPane21.setViewportView(jEditorPane16);

panelEstrutura3.add(jScrollPane21);
jScrollPane21.setBounds(10, 20, 360, 90);

jPanel12.add(panelEstrutura3);
panelEstrutura3.setBounds(430, 170, 430, 122);

panelEstrutura4.setLayout(null);

panelEstrutura4.setBorder(new javax.swing.border.TitledBorder("Objetivos e Valores"));
jScrollPane22.setViewportView(jEditorPane17);

panelEstrutura4.add(jScrollPane22);
jScrollPane22.setBounds(10, 20, 360, 90);

jPanel12.add(panelEstrutura4);
panelEstrutura4.setBounds(0, 170, 430, 122);

panelEstrutura5.setLayout(null);

panelEstrutura5.setBorder(new javax.swing.border.TitledBorder("Depend\u00eancia de Fluxo"));
jScrollPane24.setViewportView(jEditorPane20);

panelEstrutura5.add(jScrollPane24);
jScrollPane24.setBounds(10, 20, 360, 90);

jPanel12.add(panelEstrutura5);
panelEstrutura5.setBounds(430, 170, 430, 122);

panelEstrutura6.setLayout(null);

panelEstrutura6.setBorder(new javax.swing.border.TitledBorder("Objetos Manipulados"));
jScrollPane23.setViewportView(jEditorPane18);

panelEstrutura6.add(jScrollPane23);
jScrollPane23.setBounds(10, 20, 360, 70);

jPanel12.add(panelEstrutura6);
panelEstrutura6.setBounds(0, 290, 430, 100);

panelEstrutura7.setLayout(null);

panelEstrutura7.setBorder(new javax.swing.border.TitledBorder("Tempo e Controle"));
jScrollPane26.setViewportView(jEditorPane22);

panelEstrutura7.add(jScrollPane26);
jScrollPane26.setBounds(10, 20, 360, 70);

jPanel12.add(panelEstrutura7);
panelEstrutura7.setBounds(430, 290, 430, 100);

panelRecursos7.setLayout(null);

panelRecursos7.setBorder(new javax.swing.border.TitledBorder("Agentes"));
```

```
jScrollPane25.setViewportView(listaRecursos4);

panelRecursos7.add(jScrollPane25);
jScrollPane25.setBounds(30, 20, 280, 80);

btnAdicionarRecurso4.setLabel("Adicionar");
panelRecursos7.add(btnAdicionarRecurso4);
btnAdicionarRecurso4.setBounds(320, 20, 100, 23);

jPanel12.add(panelRecursos7);
panelRecursos7.setBounds(0, 390, 430, 110);

panelEstrutura8.setLayout(null);

panelEstrutura8.setBorder(new javax.swing.border.TitledBorder("Conhecimento e Compet\u00eancia"));
jScrollPane27.setViewportView(jEditorPane23);

panelEstrutura8.add(jScrollPane27);
jScrollPane27.setBounds(10, 20, 360, 80);

jPanel12.add(panelEstrutura8);
panelEstrutura8.setBounds(430, 390, 430, 110);

panelEstrutura9.setLayout(null);

panelEstrutura9.setBorder(new javax.swing.border.TitledBorder("Recursos"));
jScrollPane28.setViewportView(jEditorPane19);

panelEstrutura9.add(jScrollPane28);
jScrollPane28.setBounds(10, 20, 360, 90);

jPanel12.add(panelEstrutura9);
panelEstrutura9.setBounds(0, 500, 430, 122);

panelEstrutura10.setLayout(null);

panelEstrutura10.setBorder(new javax.swing.border.TitledBorder("Qualidade e Performance"));
jScrollPane30.setViewportView(jEditorPane25);

panelEstrutura10.add(jScrollPane30);
jScrollPane30.setBounds(10, 20, 360, 90);

jPanel12.add(panelEstrutura10);
panelEstrutura10.setBounds(430, 500, 430, 122);

jTabbedPane3.addTab("An\u00e1lise da Tarefa (MT-1)", jPanel12);

jPanel13.setLayout(new java.awt.BorderLayout());

jPanel21.setLayout(null);

jPanel22.setLayout(null);

jPanel22.setBorder(new javax.swing.border.TitledBorder("Detalhes do Item de Conhecimento"));
jLabel12.setText("Conhecimento");
jPanel22.add(jLabel12);
```

```
jLabel12.setBounds(10, 30, 100, 15);

jPanel22.add(jComboBox3);
jComboBox3.setBounds(110, 20, 170, 22);

btnAdicionarProcesso9.setText("Incluir Conhecimento");
btnAdicionarProcesso9.setActionCommand("NovaTarefa");
jPanel22.add(btnAdicionarProcesso9);
btnAdicionarProcesso9.setBounds(330, 20, 180, 23);

btnAdicionarProcesso10.setText("Visualizar Todos os Conhecimentos");
btnAdicionarProcesso10.setActionCommand("NovaTarefa");
jPanel22.add(btnAdicionarProcesso10);
btnAdicionarProcesso10.setBounds(520, 20, 240, 23);

jPanel22.add(jTextField6);
jTextField6.setBounds(110, 50, 260, 20);

jLabel13.setText("Possu\u00eddo por");
jPanel22.add(jLabel13);
jLabel13.setBounds(10, 50, 80, 15);

jPanel22.add(jTextField7);
jTextField7.setBounds(110, 80, 450, 19);

jLabel14.setText("Usado em");
jPanel22.add(jLabel14);
jLabel14.setBounds(10, 80, 100, 15);

jPanel22.add(jTextField8);
jTextField8.setBounds(110, 110, 450, 19);

jLabel15.setText("Dom\u00ednio");
jPanel22.add(jLabel15);
jLabel15.setBounds(10, 110, 100, 15);

jPanel21.add(jPanel22);
jPanel22.setBounds(10, 10, 860, 140);

jPanel21.add(jPanel23);
jPanel23.setBounds(130, 220, 10, 10);

jPanel21.add(jPanel24);
jPanel24.setBounds(390, 210, 10, 10);

jPanel25.setLayout(null);

jPanel25.setBorder(new javax.swing.border.TitledBorder("Natureza do Conhecimento"));
jPanel27.setLayout(null);

jPanel27.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0)));
jCheckBox6.setText("Formal, rigoroso");
jPanel27.add(jCheckBox6);
jCheckBox6.setBounds(10, 10, 190, 23);

jCheckBox7.setText("Emp\u00edrico, quantitativo");
```

```
jPanel27.add(jCheckBox7);
jCheckBox7.setBounds(10, 30, 210, 23);

jCheckBox8.setText("Heur\u00edstico, regras pr\u00e1ticas");
jPanel27.add(jCheckBox8);
jCheckBox8.setBounds(10, 50, 210, 23);

jCheckBox9.setText("Altamente especializado");
jPanel27.add(jCheckBox9);
jCheckBox9.setBounds(10, 70, 210, 23);

jCheckBox10.setText("Baseado em experi\u00eancia");
jPanel27.add(jCheckBox10);
jCheckBox10.setBounds(10, 90, 210, 23);

jCheckBox11.setText("Baseado em a\u00e7\u00e3o");
jPanel27.add(jCheckBox11);
jCheckBox11.setBounds(10, 110, 210, 23);

jCheckBox12.setText("Incompleto");
jPanel27.add(jCheckBox12);
jCheckBox12.setBounds(10, 130, 210, 23);

jCheckBox13.setText("Incerto, pode estar incorreto");
jPanel27.add(jCheckBox13);
jCheckBox13.setBounds(10, 150, 210, 23);

jCheckBox14.setText("Muda rapidamente");
jPanel27.add(jCheckBox14);
jCheckBox14.setBounds(10, 170, 210, 23);

jCheckBox15.setText("Dif\u00edcil de verificar");
jPanel27.add(jCheckBox15);
jCheckBox15.setBounds(10, 190, 210, 23);

jCheckBox16.setText("T\u00e1cito, dif\u00edcil de transferir");
jPanel27.add(jCheckBox16);
jCheckBox16.setBounds(10, 210, 210, 23);

jPanel27.add(jPanel28);
jPanel28.setBounds(350, 40, 10, 10);

jPanel30.setLayout(null);

jPanel30.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0)));
jPanel30.add(jCheckBox17);
jCheckBox17.setBounds(10, 10, 21, 21);

jPanel30.add(jCheckBox18);
jCheckBox18.setBounds(10, 30, 20, 21);

jPanel30.add(jCheckBox19);
jCheckBox19.setBounds(10, 50, 20, 21);

jPanel30.add(jCheckBox20);
jCheckBox20.setBounds(10, 70, 20, 21);
```

```
jPanel30.add(jCheckBox21);
jCheckBox21.setBounds(10, 90, 20, 21);

jPanel30.add(jCheckBox22);
jCheckBox22.setBounds(10, 110, 20, 21);

jPanel30.add(jCheckBox23);
jCheckBox23.setBounds(10, 130, 20, 21);

jPanel30.add(jCheckBox24);
jCheckBox24.setBounds(10, 150, 20, 21);

jPanel30.add(jCheckBox25);
jCheckBox25.setBounds(10, 170, 20, 21);

jPanel30.add(jCheckBox26);
jCheckBox26.setBounds(10, 190, 20, 21);

jPanel30.add(jCheckBox27);
jCheckBox27.setBounds(10, 210, 20, 21);

jPanel27.add(jPanel30);
jPanel30.setBounds(220, 0, 90, 240);

jPanel25.add(jPanel27);
jPanel27.setBounds(10, 50, 310, 240);

jLabel19.setText("Gargalo /");
jPanel25.add(jLabel19);
jLabel19.setBounds(240, 10, 80, 20);

jLabel20.setText("ser melhorado?");
jPanel25.add(jLabel20);
jLabel20.setBounds(230, 30, 100, 14);

jPanel21.add(jPanel25);
jPanel25.setBounds(10, 160, 340, 360);

jPanel26.setLayout(null);

jPanel26.setBorder(new javax.swing.border.TitledBorder("Formas de Conhecimento"));
jPanel29.setLayout(null);

jPanel29.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0)));
jCheckBox28.setText("Mente");
jPanel29.add(jCheckBox28);
jCheckBox28.setBounds(10, 10, 190, 23);

jCheckBox29.setText("Papel");
jPanel29.add(jCheckBox29);
jCheckBox29.setBounds(10, 30, 210, 23);

jCheckBox30.setText("Eletr\u00f4nico");
jPanel29.add(jCheckBox30);
jCheckBox30.setBounds(10, 50, 210, 23);
```

```
jCheckBox31.setText("Experi\u00eancia pela pr\u00e1tica");
jPanel29.add(jCheckBox31);
jCheckBox31.setBounds(10, 70, 210, 23);

jCheckBox32.setText("Outros");
jPanel29.add(jCheckBox32);
jCheckBox32.setBounds(10, 90, 210, 23);

jPanel31.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0)));
jPanel29.add(jPanel31);
jPanel31.setBounds(350, 40, 12, 12);

jPanel32.setLayout(null);

jPanel32.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0)));
jPanel32.add(jCheckBox39);
jCheckBox39.setBounds(10, 10, 21, 21);

jPanel32.add(jCheckBox40);
jCheckBox40.setBounds(10, 30, 20, 21);

jPanel32.add(jCheckBox41);
jCheckBox41.setBounds(10, 50, 20, 21);

jPanel32.add(jCheckBox42);
jCheckBox42.setBounds(10, 70, 20, 21);

jPanel32.add(jCheckBox43);
jCheckBox43.setBounds(10, 90, 20, 21);

jPanel29.add(jPanel32);
jPanel32.setBounds(220, 0, 90, 120);

jPanel26.add(jPanel29);
jPanel29.setBounds(10, 50, 310, 120);

jLabel21.setText("Gargalo /");
jPanel26.add(jLabel21);
jLabel21.setBounds(240, 10, 80, 20);

jLabel22.setText("ser melhorado?");
jPanel26.add(jLabel22);
jLabel22.setBounds(230, 30, 100, 14);

jPanel21.add(jPanel26);
jPanel26.setBounds(380, 160, 370, 180);

jPanel33.setLayout(null);

jPanel33.setBorder(new javax.swing.border.TitledBorder("Disponibilidade do Conhecimento"));
jPanel34.setLayout(null);

jPanel34.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0)));
jCheckBox33.setText("Limita\u00e7\u00f5es no tempo");
jPanel34.add(jCheckBox33);
```

```
jCheckBox33.setBounds(10, 10, 190, 23);

jCheckBox34.setText("Limita\u00e7\u00f5es no espa\u00e7o");
jPanel34.add(jCheckBox34);
jCheckBox34.setBounds(10, 30, 210, 23);

jCheckBox35.setText("Limita\u00e7\u00f5es no acesso");
jPanel34.add(jCheckBox35);
jCheckBox35.setBounds(10, 50, 210, 23);

jCheckBox36.setText("Limita\u00e7\u00f5es na qualidade");
jPanel34.add(jCheckBox36);
jCheckBox36.setBounds(10, 70, 210, 23);

jCheckBox37.setText("Limita\u00e7\u00f5es na forma");
jPanel34.add(jCheckBox37);
jCheckBox37.setBounds(10, 90, 210, 23);

jPanel34.add(jPanel35);
jPanel35.setBounds(350, 40, 10, 10);

jPanel36.setLayout(null);

jPanel36.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0)));
jPanel36.add(jCheckBox44);
jCheckBox44.setBounds(10, 10, 21, 21);

jPanel36.add(jCheckBox45);
jCheckBox45.setBounds(10, 30, 20, 21);

jPanel36.add(jCheckBox46);
jCheckBox46.setBounds(10, 50, 20, 21);

jPanel36.add(jCheckBox47);
jCheckBox47.setBounds(10, 70, 20, 21);

jPanel36.add(jCheckBox48);
jCheckBox48.setBounds(10, 90, 20, 21);

jPanel34.add(jPanel36);
jPanel36.setBounds(220, 0, 90, 120);

jPanel33.add(jPanel34);
jPanel34.setBounds(10, 50, 310, 120);

jLabel23.setText("Gargalo /");
jPanel33.add(jLabel23);
jLabel23.setBounds(240, 10, 80, 20);

jLabel24.setText("ser melhorado?");
jPanel33.add(jLabel24);
jLabel24.setBounds(230, 30, 100, 14);

jPanel21.add(jPanel33);
jPanel33.setBounds(380, 340, 370, 180);
```

```
jPanel13.add(jPanel21, java.awt.BorderLayout.CENTER);

jTabbedPane3.addTab("Item de Conhecimento (MT-2)", jPanel13);

jPanel7.add(jTabbedPane3, java.awt.BorderLayout.CENTER);

jTabbedPane1.addTab("TAREFA", jPanel7);

jPanel8.setLayout(new java.awt.BorderLayout());

jPanel37.setLayout(null);

jLabel16.setText("Agente");
jPanel37.add(jLabel16);
jLabel16.setBounds(10, 20, 70, 15);

jPanel37.add(jComboBox4);
jComboBox4.setBounds(70, 10, 170, 22);

btnAdicionarProcesso11.setText("Incluir Agente");
btnAdicionarProcesso11.setActionCommand("NovaTarefa");
jPanel37.add(btnAdicionarProcesso11);
btnAdicionarProcesso11.setBounds(250, 10, 150, 23);

jPanel39.setLayout(null);

jPanel39.setBorder(new javax.swing.border.TitledBorder("Organiza\u00e7\u00e3o"));
jScrollPane18.setViewportView(jTextPane1);

jPanel39.add(jScrollPane18);
jScrollPane18.setBounds(10, 20, 390, 90);

jPanel37.add(jPanel39);
jPanel39.setBounds(10, 50, 410, 120);

jPanel40.setLayout(null);

jPanel40.setBorder(new javax.swing.border.TitledBorder("Organiza\u00e7\u00e3o"));
jScrollPane29.setViewportView(jTextPane2);

jPanel40.add(jScrollPane29);
jScrollPane29.setBounds(10, 20, 390, 90);

jPanel37.add(jPanel40);
jPanel40.setBounds(10, 50, 410, 120);

jPanel41.setLayout(null);

jPanel41.setBorder(new javax.swing.border.TitledBorder("Outras compet\u00eancias"));
jScrollPane31.setViewportView(jTextPane3);

jPanel41.add(jScrollPane31);
jScrollPane31.setBounds(10, 20, 390, 90);

jPanel37.add(jPanel41);
jPanel41.setBounds(10, 300, 410, 120);
```

```
jPanel42.setLayout(null);

jPanel42.setBorder(new javax.swing.border.TitledBorder("Responsabilidades e restri\u00e7\u00f5es"));
jScrollPane32.setViewportView(jTextPane4);

jPanel42.add(jScrollPane32);
jScrollPane32.setBounds(10, 20, 390, 90);

jPanel37.add(jPanel42);
jPanel42.setBounds(420, 300, 410, 120);

panelRecursos6.setLayout(null);

panelRecursos6.setBorder(new javax.swing.border.TitledBorder("Tarefas em que est\u00e1 envolvido"));
jScrollPane33.setViewportView(listaRecursos3);

panelRecursos6.add(jScrollPane33);
jScrollPane33.setBounds(30, 20, 280, 90);

btnAdicionarRecurso3.setLabel("Adicionar");
panelRecursos6.add(btnAdicionarRecurso3);
btnAdicionarRecurso3.setBounds(320, 20, 100, 23);

jPanel37.add(panelRecursos6);
panelRecursos6.setBounds(420, 50, 430, 122);

panelRecursos8.setLayout(null);

panelRecursos8.setBorder(new javax.swing.border.TitledBorder("Agentes com quem se comunica"));
jScrollPane34.setViewportView(listaRecursos5);

panelRecursos8.add(jScrollPane34);
jScrollPane34.setBounds(10, 20, 250, 90);

btnAdicionarRecurso5.setLabel("Adicionar");
panelRecursos8.add(btnAdicionarRecurso5);
btnAdicionarRecurso5.setBounds(280, 20, 100, 23);

jPanel37.add(panelRecursos8);
panelRecursos8.setBounds(10, 170, 410, 122);

panelRecursos9.setLayout(null);

panelRecursos9.setBorder(new javax.swing.border.TitledBorder("Conhecimentos que possui"));
jScrollPane35.setViewportView(listaRecursos6);

panelRecursos9.add(jScrollPane35);
jScrollPane35.setBounds(30, 20, 280, 90);

btnAdicionarRecurso6.setLabel("Adicionar");
panelRecursos9.add(btnAdicionarRecurso6);
btnAdicionarRecurso6.setBounds(320, 20, 100, 23);

jPanel37.add(panelRecursos9);
panelRecursos9.setBounds(420, 170, 430, 122);
```

```
jTabbedPane5.addTab("Agentes (MA-1)", jPanel37);

jPanel38.setLayout(new java.awt.BorderLayout());

jPanel43.setLayout(new java.awt.BorderLayout());

jPanel43.add(jTextPane6, java.awt.BorderLayout.CENTER);

jTabbedPane6.addTab("Impactos e Mudan\u00e7as na Organiza\u00e7\u00e3o", jPanel43);

jPanel44.setLayout(new java.awt.BorderLayout());

jScrollPane36.setViewportView(jTextPane5);

jPanel44.add(jScrollPane36, java.awt.BorderLayout.CENTER);

jTabbedPane6.addTab("Impactos e Mudan\u00e7as em Tarefas/Agentes Espec\u00edficos", jPanel44);

jPanel45.setLayout(new java.awt.BorderLayout());

jPanel45.add(jTextPane7, java.awt.BorderLayout.CENTER);

jTabbedPane6.addTab("Atitude e Comprometimento", jPanel45);

jPanel46.setLayout(new java.awt.BorderLayout());

jPanel46.add(jTextPane8, java.awt.BorderLayout.CENTER);

jTabbedPane6.addTab("A\u00e7\u00f5es Propostas", jPanel46);

jPanel38.add(jTabbedPane6, java.awt.BorderLayout.CENTER);

jTabbedPane5.addTab("Documento para decis\u00e3o de impactos e melhorias (OTA-1)", jPanel38);

jPanel8.add(jTabbedPane5, java.awt.BorderLayout.CENTER);

jTabbedPane1.addTab("AGENTE", jPanel8);

jPanel9.setLayout(null);

jLabel17.setText("Conhecimento");
jLabel17.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
jPanel9.add(jLabel17);
jLabel17.setBounds(20, 30, 110, 14);

jPanel9.add(cmbTarefa1);
cmbTarefa1.setBounds(140, 30, 260, 22);

jButton3.setText("Incluir Conhecimento");
jPanel9.add(jButton3);
jButton3.setBounds(410, 30, 230, 23);

jPanel47.setLayout(null);

jPanel47.setBorder(new javax.swing.border.TitledBorder("Modelo do Conhecimento"));
```

```
jScrollPane37.setViewportView(jTextPane9);

jPanel47.add(jScrollPane37);
jScrollPane37.setBounds(10, 20, 780, 490);

jPanel9.add(jPanel47);
jPanel47.setBounds(20, 60, 810, 530);

jTabbedPane1.addTab("CONHECIMENTO", jPanel9);

jPanel10.setLayout(new java.awt.BorderLayout());

jPanel48.setLayout(null);

jLabel18.setText("Transa\u00e7\u00e3o");
jLabel18.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
jPanel48.add(jLabel18);
jLabel18.setBounds(20, 30, 110, 14);

cmbTarefa2.addItem("Pedido de Avaliação de Aplicação");
jPanel48.add(cmbTarefa2);
cmbTarefa2.setBounds(100, 30, 260, 22);

jButton4.setText("Incluir Transa\u00e7\u00e3o");
jPanel48.add(jButton4);
jButton4.setBounds(370, 30, 180, 23);

jPanel50.setLayout(null);

jPanel50.setBorder(new javax.swing.border.TitledBorder("Objeto de Informa\u00e7\u00e3o"));
jScrollPane38.setViewportView(jTextPane10);

jPanel50.add(jScrollPane38);
jScrollPane38.setBounds(10, 20, 390, 90);

jPanel48.add(jPanel50);
jPanel50.setBounds(10, 60, 410, 120);

panelRecursos10.setLayout(null);

panelRecursos10.setBorder(new javax.swing.border.TitledBorder("Agentes envolvidos"));
jLabel25.setText("Envia");
jLabel25.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
panelRecursos10.add(jLabel25);
jLabel25.setBounds(20, 30, 60, 14);

cmbTarefa3.addItem("Entrada de Dados");
panelRecursos10.add(cmbTarefa3);
cmbTarefa3.setBounds(80, 30, 260, 22);

jLabel26.setText("Recebe");
jLabel26.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
panelRecursos10.add(jLabel26);
jLabel26.setBounds(20, 60, 60, 14);

cmbTarefa4.addItem("Sistema de Conhecimento");
```

```
        panelRecursos10.add(cmbTarefa4);
        cmbTarefa4.setBounds(80, 60, 260, 22);

        jPanel48.add(panelRecursos10);
        panelRecursos10.setBounds(420, 60, 410, 122);

        jPanel51.setLayout(null);

        jPanel51.setBorder(new javax.swing.border.TitledBorder("Plano de Comunica\u00e7\u00e3o"));
        jScrollPane39.setViewportView(jTextPane11);

        jPanel51.add(jScrollPane39);
        jScrollPane39.setBounds(10, 20, 390, 90);

        jPanel48.add(jPanel51);
        jPanel51.setBounds(10, 180, 410, 120);

        jPanel52.setLayout(null);

        jPanel52.setBorder(new javax.swing.border.TitledBorder("Restri\u00e7\u00f5es"));
        jScrollPane40.setViewportView(jTextPane12);

        jPanel52.add(jScrollPane40);
        jScrollPane40.setBounds(10, 20, 390, 90);

        jPanel48.add(jPanel52);
        jPanel52.setBounds(420, 180, 410, 120);

        jPanel53.setLayout(null);

        jPanel53.setBorder(new javax.swing.border.TitledBorder("Especifica\u00e7\u00e3o das
Informa\u00e7\u00f5es transferidas"));
        jScrollPane41.setViewportView(jTextPane13);

        jPanel53.add(jScrollPane41);
        jScrollPane41.setBounds(10, 20, 390, 90);

        jPanel48.add(jPanel53);
        jPanel53.setBounds(10, 310, 410, 120);

        jTabbedPane7.addTab("Descri\u00e7\u00e3o da Transa\u00e7\u00e3o (MC-1)", jPanel48);

        jPanel49.setLayout(null);

        jPanel49.add(cmbTarefa5);
        cmbTarefa5.setBounds(100, 30, 260, 22);

        jLabel27.setText("Transa\u00e7\u00e3o");
        jLabel27.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
        jPanel49.add(jLabel27);
        jLabel27.setBounds(20, 30, 110, 14);

        jButton5.setText("Incluir Transa\u00e7\u00e3o");
        jPanel49.add(jButton5);
        jButton5.setBounds(370, 30, 180, 23);
```

```
panelRecursos11.setLayout(null);

panelRecursos11.setBorder(new javax.swing.border.TitledBorder("Agentes envolvidos"));
jLabel28.setText("Envia");
jLabel28.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
panelRecursos11.add(jLabel28);
jLabel28.setBounds(20, 30, 60, 14);

panelRecursos11.add(cmbTarefa6);
cmbTarefa6.setBounds(80, 30, 260, 22);

jLabel29.setText("Recebe");
jLabel29.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
panelRecursos11.add(jLabel29);
jLabel29.setBounds(20, 60, 60, 14);

panelRecursos11.add(cmbTarefa7);
cmbTarefa7.setBounds(80, 60, 260, 22);

jPanel49.add(panelRecursos11);
panelRecursos11.setBounds(0, 60, 410, 122);

panelRecursos12.setLayout(null);

panelRecursos12.setBorder(new javax.swing.border.TitledBorder("Itens de Informa\u00e7\u00e3o"));
jScrollPane42.setViewportView(listaRecursos7);

panelRecursos12.add(jScrollPane42);
jScrollPane42.setBounds(10, 20, 250, 90);

btnAdicionarRecurso7.setLabel("Adicionar");
panelRecursos12.add(btnAdicionarRecurso7);
btnAdicionarRecurso7.setBounds(280, 20, 100, 23);

jPanel49.add(panelRecursos12);
panelRecursos12.setBounds(0, 190, 410, 122);

jTabbedPane7.addTab("Especifica\u00e7\u00e3o das informa\u00e7\u00f5es transferidas (MC-2)", jPanel49);

jPanel10.add(jTabbedPane7, java.awt.BorderLayout.CENTER);

jTabbedPane1.addTab("COMUNICA\u00c7\u00c3O", jPanel10);

getContentPane().add(jTabbedPane1, java.awt.BorderLayout.CENTER);

menuBar.setText("Principal");
abrir.setText("Abrir Projeto");
abrir.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      abrirActionPerformed(evt);
   }
});

menuBar.add(abrir);

novo.setText("Novo Projeto");
```

```
        menuBar.add(novo);

        jMenuBar1.add(menuBar);

        setJMenuBar(jMenuBar1);

        pack();
    }

    private void cmbTarefasMO4ItemStateChanged(java.awt.event.ItemEvent evt) {
        // TODO add your handling code here:
        Tarefa tarefa = retornaTarefa(((String)cmbTarefasMO4.getSelectedItem()));
        if  (!(tarefa == null)) {
            txtExecutadaPor.setText(tarefa.getExecutadaPor());
            txtLocal.setText(tarefa.getLocalizacao());
            chkTarefaIntensiva.setSelected(tarefa.isIntensiva());
            spnGrauSignificancia.setValue(tarefa.getGrauSignificancia());
        }
    }

    private void btnAdicionarTarefaMO3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int idProcesso = retornaIdProcesso(((String)cmbProcessosMO3.getSelectedItem()));
        String msg = JOptionPane.showInputDialog(this,"Informe o nome da Tarefa","Cadastro de
Tarefas",JOptionPane.PLAIN_MESSAGE);
        ListModel model;
        if (msg.trim().equals("")){
            JOptionPane.showMessageDialog(this,"O nome da Tarefa não foi
informado.","Atenção",JOptionPane.WARNING_MESSAGE);

        }else{
            try{
                Tarefa tar = new Tarefa(msg);
                tar.setIdProcesso(idProcesso);
                fachada.salveTarefa(tar);

            }catch(ExcecaoBancoIndisponivel e){
                JOptionPane.showMessageDialog(this,e.getMessage(),"Atenção",JOptionPane.WARNING_MESSAGE);
            }
            carregaComboTarefa(idProcesso);
            cmbTarefasMO4.setSelectedItem(msg.trim());
        }

    }

    private void cmbProcessosMO3ItemStateChanged(java.awt.event.ItemEvent evt) {
        // TODO add your handling code here:
        //String msg = JOptionPane.showInputDialog(this,"Informe o nome do Conhecimento","Cadastro de
Recursos",JOptionPane.PLAIN_MESSAGE);
        String item = new String(((String)cmbProcessosMO3.getSelectedItem()));
        carregaComboTarefa(retornaIdProcesso(item));


    }

    private void btnAdicionarConhecimentoActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        // TODO add your handling code here:
        String msg = JOptionPane.showInputDialog(this,"Informe o nome do Conhecimento","Cadastro de
Conhecimentos",JOptionPane.PLAIN_MESSAGE);
        ListModel model;
        if (msg.trim().equals("")){
            JOptionPane.showMessageDialog(this,"O nome do Conhecimento não foi
informado","Atenção",JOptionPane.WARNING_MESSAGE);

        }else{
            try{
                Conhecimento con = new Conhecimento(msg);

                con.setIdOrganizacao(organizacao.getIdOrganizacao());

                fachada.salveConhecimento(con);
            }catch(ExcecaoBancoIndisponivel e){
                JOptionPane.showMessageDialog(this,e.getMessage(),"Atenção",JOptionPane.WARNING_MESSAGE);
            }
            model = listaConhecimentosMO2.getModel();
            ((DefaultListModel)model).add(((DefaultListModel)model).size(),msg);
        }


    }

    private void btnAdicionarRecursoActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String msg = JOptionPane.showInputDialog(this,"Informe o nome do Recurso","Cadastro de
Recursos",JOptionPane.PLAIN_MESSAGE);
        ListModel model;
        if (msg.trim().equals("")){
            JOptionPane.showMessageDialog(this,"O nome do Recurso não foi
informado","Atenção",JOptionPane.WARNING_MESSAGE);

        }else{
            try{
                Recurso rec = new Recurso(msg);
                rec.setidOrganizacao(organizacao.getIdOrganizacao());

                fachada.salveRecurso(rec);
            }catch(ExcecaoBancoIndisponivel e){
                JOptionPane.showMessageDialog(this,e.getMessage(),"Atenção",JOptionPane.WARNING_MESSAGE);
            }
            model = listaRecursos.getModel();
            ((DefaultListModel)model).add(((DefaultListModel)model).size(),msg);
        }

    }

    private void abrirActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void btnVisualizaEstruturaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
```

```
        ImageView formVisao = new ImageView(new ImageIcon(txtEstrutura.getText()),"Estrutura da Organização");
        formVisao.setExtendedState(this.MAXIMIZED_BOTH);
        formVisao.setVisible(true);


    }

    private void btnAdicionarEstruturaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        FileDialog getFile = new FileDialog(this,"Selecione arquivo",FileDialog.LOAD);
        getFile.setVisible(true);
        txtEstrutura.setText(getFile.getDirectory()+getFile.getFile());

    }

    private void btnSalvarActionPerformed(java.awt.event.ActionEvent evt) {
        // Salvando Informações da Organizacao
        organizacao.setMissao(edtMissao.getText());
        organizacao.setVisao(edtVisao.getText());
        organizacao.setObjetivos(edtObjetivos.getText());
        organizacao.setFatoresExternos(edtFatoresExternos.getText());
        organizacao.setOportunidades(edtProblemas.getText());
        organizacao.setEstrutura(txtEstrutura.getText());
        organizacao.setPessoas(edtPessoas.getText());
        organizacao.setCulturaPoder(edtCulturaPoder.getText());


        try {
            fachada.salveOrganizacao(organizacao);
        }catch(ExcecaoBancoIndisponivel e){
            JOptionPane.showMessageDialog(this,e.getMessage(),"Atenção",JOptionPane.WARNING_MESSAGE);
        }
    }

    private void btnAdicionarProcessoActionPerformed(java.awt.event.ActionEvent evt) {
        String msg = JOptionPane.showInputDialog(this,"Informe o nome do Processo","Cadastro de
Processos",JOptionPane.PLAIN_MESSAGE);
        ListModel model;
        if (msg.trim().equals("")){
            JOptionPane.showMessageDialog(this,"O nome do Processo não foi
informado","Atenção",JOptionPane.WARNING_MESSAGE);

        }else{
            try{
                Processo proc = new Processo(msg);
                proc.setidOrganizacao(organizacao.getIdOrganizacao());

                fachada.salveProcesso(proc);
            }catch(ExcecaoBancoIndisponivel e){
                JOptionPane.showMessageDialog(this,e.getMessage(),"Atenção",JOptionPane.WARNING_MESSAGE);
            }
            model = listaProcessos.getModel();
            ((DefaultListModel)model).add(((DefaultListModel)model).size(),msg);
        }
    }
```

```
public void carregaDadosProjeto(){

   try {
      int idOrganizacao;
      this.idProjetoAberto = fachada.getIdProjetoAtual();
      // Carregando Organizacao
      organizacao = fachada.retorneOrganizacao(idProjetoAberto);
      edtMissao.setText(organizacao.getMissao());
      edtVisao.setText(organizacao.getVisao());
      edtObjetivos.setText(organizacao.getObjetivos());
      edtFatoresExternos.setText(organizacao.getFatoresExternos());
      edtProblemas.setText(organizacao.getOportunidades().trim());
      txtEstrutura.setText(organizacao.getEstrutura().trim());
      edtPessoas.setText(organizacao.getPessoas());
      edtCulturaPoder.setText(organizacao.getCulturaPoder());

      idOrganizacao = organizacao.getIdOrganizacao();
      // Carregando informações do Processo
      processos = fachada.retorneProcessos(idOrganizacao);
      carregaComboProcesso(processos);


      // Carregando informações dos recursos
      recursos = fachada.retorneRecursos(idOrganizacao);

      // Carregando informações dos conhecimentos
      conhecimentos = fachada.retorneConhecimentos(idOrganizacao);

   }catch (ExcecaoBancoIndisponivel e){
      System.out.println(e.getMessage());
   }
   listaProcessos.setModel(criaListModelProcesso(processos));
   listaRecursos.setModel(criaListModelRecurso(recursos));
   listaConhecimentosMO2.setModel(criaListModelConhecimento(conhecimentos));



}


public DefaultListModel criaListModelProcesso(List processo){
   DefaultListModel model = new DefaultListModel();
   for (int i = 0;i < processos.size();i++){
      model.add(i,((Processo)processos.get(i)).getNome());
   }
   return model;
}

public DefaultListModel criaListModelRecurso(List recursos){
   DefaultListModel model = new DefaultListModel();
   for (int i = 0;i < recursos.size();i++){
      model.add(i,((Recurso)recursos.get(i)).getDescricao());
   }
   return model;
}
```

```
public DefaultListModel criaListModelConhecimento(List conhecimentos){
   DefaultListModel model = new DefaultListModel();
   for (int i = 0;i < conhecimentos.size();i++){
      model.add(i,((Conhecimento)conhecimentos.get(i)).getDescricao());
   }
   return model;
}

public void carregaComboProcesso(List processos){
   for (int i = 0;i < processos.size();i++){
      cmbProcessosMO3.addItem(new String(((Processo)processos.get(i)).getNome()));

   }

}

public void carregaComboTarefa(int idProcesso){
   try {
      tarefas = fachada.retorneTarefas(idProcesso);
   }catch(ExcecaoBancoIndisponivel e){
      JOptionPane.showMessageDialog(this,e.getMessage(),"Atenção",JOptionPane.WARNING_MESSAGE);
   }


   // Limpando o combo antes de carregar os dados
   cmbTarefasMO4.removeAllItems();
   for (int i = 0;i < tarefas.size();i++){
      cmbTarefasMO4.addItem(new String(((Tarefa)tarefas.get(i)).getNome().trim()));

   }

}



public int retornaIdProcesso(String nome){

   for (int i = 0;i < processos.size();i++){
      if (nome.equals(((Processo)processos.get(i)).getNome())) {
         return ((Processo)processos.get(i)).getidProcesso();
      }
   }
   return -1;
}

public Tarefa retornaTarefa(String nome){

   nome = nome.trim();

   for (int i = 0;i < tarefas.size();i++){
      if (nome.equals(((Tarefa)tarefas.get(i)).getNome())) {
         return ((Tarefa)tarefas.get(i));
      }
   }
   return null;
}
```

```
/**
 * @param args the command line arguments
 */
/**  public static void main(String args[]) {
 * java.awt.EventQueue.invokeLater(new Runnable() {
 * public void run() {
 * new Principal().setVisible(true);
 * }
 * }
 * }**/
// Variables declaration - do not modify
private javax.swing.JMenuItem abrir;
private javax.swing.JButton btnAdicionarConhecimento;
private javax.swing.JButton btnAdicionarConhecimentoMO3;
private javax.swing.JButton btnAdicionarEstrutura;
private javax.swing.JButton btnAdicionarProcesso;
private javax.swing.JButton btnAdicionarProcesso1;
private javax.swing.JButton btnAdicionarProcesso10;
private javax.swing.JButton btnAdicionarProcesso11;
private javax.swing.JButton btnAdicionarProcesso7;
private javax.swing.JButton btnAdicionarProcesso9;
private javax.swing.JButton btnAdicionarRecurso;
private javax.swing.JButton btnAdicionarRecurso3;
private javax.swing.JButton btnAdicionarRecurso4;
private javax.swing.JButton btnAdicionarRecurso5;
private javax.swing.JButton btnAdicionarRecurso6;
private javax.swing.JButton btnAdicionarRecurso7;
private javax.swing.JButton btnAdicionarTarefaMO3;
private javax.swing.JButton btnSalvar;
private javax.swing.JButton btnVisualizaEstrutura;
private javax.swing.JCheckBox chkTarefaIntensiva;
private javax.swing.JComboBox cmbProcesso;
private javax.swing.JComboBox cmbProcessosMO3;
private javax.swing.JComboBox cmbTarefa;
private javax.swing.JComboBox cmbTarefa1;
private javax.swing.JComboBox cmbTarefa2;
private javax.swing.JComboBox cmbTarefa3;
private javax.swing.JComboBox cmbTarefa4;
private javax.swing.JComboBox cmbTarefa5;
private javax.swing.JComboBox cmbTarefa6;
private javax.swing.JComboBox cmbTarefa7;
private javax.swing.JComboBox cmbTarefasMO4;
private javax.swing.JEditorPane edtCulturaPoder;
private javax.swing.JEditorPane edtFatoresExternos;
private javax.swing.JEditorPane edtMissao;
private javax.swing.JTextPane edtObjetivos;
private javax.swing.JEditorPane edtPessoas;
private javax.swing.JEditorPane edtProblemas;
private javax.swing.JEditorPane edtVisao;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JCheckBox jCheckBox10;
private javax.swing.JCheckBox jCheckBox11;
```

```
private javax.swing.JCheckBox jCheckBox12;
private javax.swing.JCheckBox jCheckBox13;
private javax.swing.JCheckBox jCheckBox14;
private javax.swing.JCheckBox jCheckBox15;
private javax.swing.JCheckBox jCheckBox16;
private javax.swing.JCheckBox jCheckBox17;
private javax.swing.JCheckBox jCheckBox18;
private javax.swing.JCheckBox jCheckBox19;
private javax.swing.JCheckBox jCheckBox2;
private javax.swing.JCheckBox jCheckBox20;
private javax.swing.JCheckBox jCheckBox21;
private javax.swing.JCheckBox jCheckBox22;
private javax.swing.JCheckBox jCheckBox23;
private javax.swing.JCheckBox jCheckBox24;
private javax.swing.JCheckBox jCheckBox25;
private javax.swing.JCheckBox jCheckBox26;
private javax.swing.JCheckBox jCheckBox27;
private javax.swing.JCheckBox jCheckBox28;
private javax.swing.JCheckBox jCheckBox29;
private javax.swing.JCheckBox jCheckBox3;
private javax.swing.JCheckBox jCheckBox30;
private javax.swing.JCheckBox jCheckBox31;
private javax.swing.JCheckBox jCheckBox32;
private javax.swing.JCheckBox jCheckBox33;
private javax.swing.JCheckBox jCheckBox34;
private javax.swing.JCheckBox jCheckBox35;
private javax.swing.JCheckBox jCheckBox36;
private javax.swing.JCheckBox jCheckBox37;
private javax.swing.JCheckBox jCheckBox39;
private javax.swing.JCheckBox jCheckBox4;
private javax.swing.JCheckBox jCheckBox40;
private javax.swing.JCheckBox jCheckBox41;
private javax.swing.JCheckBox jCheckBox42;
private javax.swing.JCheckBox jCheckBox43;
private javax.swing.JCheckBox jCheckBox44;
private javax.swing.JCheckBox jCheckBox45;
private javax.swing.JCheckBox jCheckBox46;
private javax.swing.JCheckBox jCheckBox47;
private javax.swing.JCheckBox jCheckBox48;
private javax.swing.JCheckBox jCheckBox5;
private javax.swing.JCheckBox jCheckBox6;
private javax.swing.JCheckBox jCheckBox7;
private javax.swing.JCheckBox jCheckBox8;
private javax.swing.JCheckBox jCheckBox9;
private javax.swing.JComboBox jComboBox2;
private javax.swing.JComboBox jComboBox4;
private javax.swing.JEditorPane jEditorPane10;
private javax.swing.JEditorPane jEditorPane11;
private javax.swing.JEditorPane jEditorPane12;
private javax.swing.JEditorPane jEditorPane13;
private javax.swing.JEditorPane jEditorPane14;
private javax.swing.JEditorPane jEditorPane15;
private javax.swing.JEditorPane jEditorPane16;
private javax.swing.JEditorPane jEditorPane17;
private javax.swing.JEditorPane jEditorPane18;
private javax.swing.JEditorPane jEditorPane19;
```

```
private javax.swing.JEditorPane jEditorPane20;
private javax.swing.JEditorPane jEditorPane22;
private javax.swing.JEditorPane jEditorPane23;
private javax.swing.JEditorPane jEditorPane25;
private javax.swing.JEditorPane jEditorPane3;
private javax.swing.JEditorPane jEditorPane4;
private javax.swing.JEditorPane jEditorPane5;
private javax.swing.JEditorPane jEditorPane6;
private javax.swing.JEditorPane jEditorPane7;
private javax.swing.JEditorPane jEditorPane8;
private javax.swing.JEditorPane jEditorPane9;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel18;
private javax.swing.JLabel jLabel19;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel20;
private javax.swing.JLabel jLabel21;
private javax.swing.JLabel jLabel22;
private javax.swing.JLabel jLabel23;
private javax.swing.JLabel jLabel24;
private javax.swing.JLabel jLabel25;
private javax.swing.JLabel jLabel26;
private javax.swing.JLabel jLabel27;
private javax.swing.JLabel jLabel28;
private javax.swing.JLabel jLabel29;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel30;
private javax.swing.JLabel jLabel31;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel10;
private javax.swing.JPanel jPanel11;
private javax.swing.JPanel jPanel12;
private javax.swing.JPanel jPanel13;
private javax.swing.JPanel jPanel14;
private javax.swing.JPanel jPanel16;
private javax.swing.JPanel jPanel17;
private javax.swing.JPanel jPanel18;
private javax.swing.JPanel jPanel19;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel20;
private javax.swing.JPanel jPanel21;
```

```
private javax.swing.JPanel jPanel22;
private javax.swing.JPanel jPanel23;
private javax.swing.JPanel jPanel24;
private javax.swing.JPanel jPanel25;
private javax.swing.JPanel jPanel26;
private javax.swing.JPanel jPanel27;
private javax.swing.JPanel jPanel28;
private javax.swing.JPanel jPanel29;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel30;
private javax.swing.JPanel jPanel31;
private javax.swing.JPanel jPanel32;
private javax.swing.JPanel jPanel33;
private javax.swing.JPanel jPanel34;
private javax.swing.JPanel jPanel35;
private javax.swing.JPanel jPanel36;
private javax.swing.JPanel jPanel37;
private javax.swing.JPanel jPanel38;
private javax.swing.JPanel jPanel39;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel40;
private javax.swing.JPanel jPanel41;
private javax.swing.JPanel jPanel42;
private javax.swing.JPanel jPanel43;
private javax.swing.JPanel jPanel44;
private javax.swing.JPanel jPanel45;
private javax.swing.JPanel jPanel46;
private javax.swing.JPanel jPanel47;
private javax.swing.JPanel jPanel48;
private javax.swing.JPanel jPanel49;
private javax.swing.JPanel jPanel5;
private javax.swing.JPanel jPanel50;
private javax.swing.JPanel jPanel51;
private javax.swing.JPanel jPanel52;
private javax.swing.JPanel jPanel53;
private javax.swing.JPanel jPanel6;
private javax.swing.JPanel jPanel7;
private javax.swing.JPanel jPanel8;
private javax.swing.JPanel jPanel9;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane10;
private javax.swing.JScrollPane jScrollPane11;
private javax.swing.JScrollPane jScrollPane12;
private javax.swing.JScrollPane jScrollPane13;
private javax.swing.JScrollPane jScrollPane14;
private javax.swing.JScrollPane jScrollPane15;
private javax.swing.JScrollPane jScrollPane16;
private javax.swing.JScrollPane jScrollPane17;
private javax.swing.JScrollPane jScrollPane18;
private javax.swing.JScrollPane jScrollPane19;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane20;
private javax.swing.JScrollPane jScrollPane21;
private javax.swing.JScrollPane jScrollPane22;
private javax.swing.JScrollPane jScrollPane23;
private javax.swing.JScrollPane jScrollPane24;
```

```
private javax.swing.JScrollPane jScrollPane25;
private javax.swing.JScrollPane jScrollPane26;
private javax.swing.JScrollPane jScrollPane27;
private javax.swing.JScrollPane jScrollPane28;
private javax.swing.JScrollPane jScrollPane29;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JScrollPane jScrollPane30;
private javax.swing.JScrollPane jScrollPane31;
private javax.swing.JScrollPane jScrollPane32;
private javax.swing.JScrollPane jScrollPane33;
private javax.swing.JScrollPane jScrollPane34;
private javax.swing.JScrollPane jScrollPane35;
private javax.swing.JScrollPane jScrollPane36;
private javax.swing.JScrollPane jScrollPane37;
private javax.swing.JScrollPane jScrollPane38;
private javax.swing.JScrollPane jScrollPane39;
private javax.swing.JScrollPane jScrollPane4;
private javax.swing.JScrollPane jScrollPane40;
private javax.swing.JScrollPane jScrollPane41;
private javax.swing.JScrollPane jScrollPane42;
private javax.swing.JScrollPane jScrollPane5;
private javax.swing.JScrollPane jScrollPane6;
private javax.swing.JScrollPane jScrollPane7;
private javax.swing.JScrollPane jScrollPane8;
private javax.swing.JScrollPane jScrollPane9;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTabbedPane jTabbedPane2;
private javax.swing.JTabbedPane jTabbedPane3;
private javax.swing.JTabbedPane jTabbedPane4;
private javax.swing.JTabbedPane jTabbedPane5;
private javax.swing.JTabbedPane jTabbedPane6;
private javax.swing.JTabbedPane jTabbedPane7;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField4;
private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
private javax.swing.JTextField jTextField7;
private javax.swing.JTextField jTextField8;
private javax.swing.JTextPane jTextPane1;
private javax.swing.JTextPane jTextPane10;
private javax.swing.JTextPane jTextPane11;
private javax.swing.JTextPane jTextPane12;
private javax.swing.JTextPane jTextPane13;
private javax.swing.JTextPane jTextPane2;
private javax.swing.JTextPane jTextPane3;
private javax.swing.JTextPane jTextPane4;
private javax.swing.JTextPane jTextPane5;
private javax.swing.JTextPane jTextPane6;
private javax.swing.JTextPane jTextPane7;
private javax.swing.JTextPane jTextPane8;
private javax.swing.JTextPane jTextPane9;
private javax.swing.JLabel lblMissao;
private javax.swing.JLabel lblMissao1;
private javax.swing.JLabel lblMissao3;
private javax.swing.JLabel lblObjetivos;
private javax.swing.JLabel lblVisao;
```

```
private javax.swing.JList listaConhecimentosMO2;
private javax.swing.JList listaConhecimentosMO3;
public javax.swing.JList listaProcessos;
private javax.swing.JList listaRecursos;
private javax.swing.JList listaRecursos3;
private javax.swing.JList listaRecursos4;
private javax.swing.JList listaRecursos5;
private javax.swing.JList listaRecursos6;
private javax.swing.JList listaRecursos7;
private javax.swing.JMenu menuBar;
private javax.swing.JMenuItem novo;
private javax.swing.JPanel panelEstrutura;
private javax.swing.JPanel panelEstrutura1;
private javax.swing.JPanel panelEstrutura10;
private javax.swing.JPanel panelEstrutura2;
private javax.swing.JPanel panelEstrutura3;
private javax.swing.JPanel panelEstrutura4;
private javax.swing.JPanel panelEstrutura5;
private javax.swing.JPanel panelEstrutura6;
private javax.swing.JPanel panelEstrutura7;
private javax.swing.JPanel panelEstrutura8;
private javax.swing.JPanel panelEstrutura9;
private javax.swing.JPanel panelPessoas;
private javax.swing.JPanel panelPessoas1;
private javax.swing.JPanel panelProcessos;
private javax.swing.JPanel panelRecursos;
private javax.swing.JPanel panelRecursos1;
private javax.swing.JPanel panelRecursos10;
private javax.swing.JPanel panelRecursos11;
private javax.swing.JPanel panelRecursos12;
private javax.swing.JPanel panelRecursos2;
private javax.swing.JPanel panelRecursos3;
private javax.swing.JPanel panelRecursos4;
private javax.swing.JPanel panelRecursos5;
private javax.swing.JPanel panelRecursos6;
private javax.swing.JPanel panelRecursos7;
private javax.swing.JPanel panelRecursos8;
private javax.swing.JPanel panelRecursos9;
private javax.swing.JSpinner spnGrauSignificancia;
private javax.swing.JTextField txtEstrutura;
private javax.swing.JTextField txtExecutadaPor;
private javax.swing.JTextField txtLocal;
// End of variables declaration

private int idProjetoAberto;
private List processos;
private List recursos;
private List conhecimentos;
private List tarefas;
private Organizacao organizacao;

public static void main(String args[]) {
   java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
         new Principal().setVisible(true);
      }
```

```
        });
    }

}
```

- Camada Lógica

```
package commonkads.logica.fachada;

/**
 *
 * @author Cleosvaldo Jr.
 */
public class FabricaDeFachadaLogica {

    private FabricaDeFachadaLogica() {
    }

    /**
     * Retorna uma classe que implementa a fachada.
     */
    public static FachadaLogica crie() {
        return new MinhaFachadaLogica();
    }
}
```

```
package commonkads.logica.fachada;


import commonkads.logica.modelo.Agente;
import commonkads.logica.modelo.Conhecimento;
import commonkads.logica.modelo.Organizacao;
import commonkads.logica.modelo.Problema;
import commonkads.logica.modelo.Processo;
import commonkads.logica.modelo.Projeto;
import commonkads.logica.modelo.Recurso;
import commonkads.logica.modelo.Solucao;
import commonkads.logica.modelo.Tarefa;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.List;

/**
 * Define os serviços oferecidos pela camada lógica.
 * @author Cleosvaldo Jr.
 */
public interface FachadaLogica {

    /**
     * Cria um novo processo. Retorna false se não puder criar.
     */
    public boolean salveProcesso(Processo processo) throws ExcecaoBancoIndisponivel;
```

```
    public boolean salveAgente(Agente agente) throws ExcecaoBancoIndisponivel;
    public boolean salveConhecimento(Conhecimento conhecimento) throws ExcecaoBancoIndisponivel;
    public boolean salveOrganizacao(Organizacao organizacao) throws ExcecaoBancoIndisponivel;
    public boolean salveProblema(Problema problema) throws ExcecaoBancoIndisponivel;
    public boolean salveProjeto(Projeto projeto) throws ExcecaoBancoIndisponivel;
    public boolean salveRecurso(Recurso recurso) throws ExcecaoBancoIndisponivel;
    public boolean salveSolucao(Solucao solucao) throws ExcecaoBancoIndisponivel;
    public boolean salveTarefa(Tarefa tarefa) throws ExcecaoBancoIndisponivel;


    public List retorneProcessos(int idOrganizacao) throws ExcecaoBancoIndisponivel;
    public List retorneRecursos(int idOrganizacao) throws ExcecaoBancoIndisponivel;
    public Organizacao retorneOrganizacao(int idProjeto) throws ExcecaoBancoIndisponivel;
    public List retorneProjetos() throws ExcecaoBancoIndisponivel;
    public List retorneConhecimentos(int idOrganizacao) throws ExcecaoBancoIndisponivel;
    public List retorneTarefas(int idProcesso) throws ExcecaoBancoIndisponivel;

    public void setIdProjetoAtual(int idProjeto);
    public int getIdProjetoAtual();
    public void setIdOrganizacaoAtual(int idOrganizacao);
    public int getIdOrganizacaoAtual();

}



package commonkads.logica.fachada;

import commonkads.logica.modelo.Agente;
import commonkads.logica.modelo.Conhecimento;
import commonkads.logica.modelo.Organizacao;
import commonkads.logica.modelo.Problema;
import commonkads.logica.modelo.Processo;
import commonkads.logica.modelo.Projeto;
import commonkads.logica.modelo.Recurso;
import commonkads.logica.modelo.Solucao;
import commonkads.logica.modelo.Tarefa;
import commonkads.logica.regras.ModeloGerente;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import commonkads.persistencia.fachada.FabricaDeFachadaPersistencia;
import commonkads.persistencia.fachada.FachadaPersistencia;
import java.util.List;



public class MinhaFachadaLogica implements FachadaLogica {

    ModeloGerente modeloGerente;

    FachadaPersistencia fachadaPersistencia;

    public MinhaFachadaLogica() {
        fachadaPersistencia = FabricaDeFachadaPersistencia.crie();
        modeloGerente = new ModeloGerente(fachadaPersistencia);

    }
```

```
public boolean salveProcesso(Processo processo) throws ExcecaoBancoIndisponivel{
   return modeloGerente.salveProcesso(processo);
}

public boolean salveAgente(Agente agente) throws ExcecaoBancoIndisponivel{
   return modeloGerente.salveAgente(agente);
}
public boolean salveConhecimento(Conhecimento conhecimento) throws ExcecaoBancoIndisponivel{
   return modeloGerente.salveConhecimento(conhecimento);
}
public boolean salveOrganizacao(Organizacao organizacao) throws ExcecaoBancoIndisponivel{
   return modeloGerente.salveOrganizacao(organizacao);
}
public boolean salveProblema(Problema problema) throws ExcecaoBancoIndisponivel{
   return modeloGerente.salveProblema(problema);
}
public boolean salveProjeto(Projeto projeto) throws ExcecaoBancoIndisponivel{
   return modeloGerente.salveProjeto(projeto);
}

public boolean salveRecurso(Recurso recurso) throws ExcecaoBancoIndisponivel{
   return modeloGerente.salveRecurso(recurso);
}

public boolean salveSolucao(Solucao solucao) throws ExcecaoBancoIndisponivel{
   return modeloGerente.salveSolucao(solucao);
}

public boolean salveTarefa(Tarefa tarefa) throws ExcecaoBancoIndisponivel{
   return modeloGerente.salveTarefa(tarefa);
}

public List retorneProcessos(int idOrganizacao) throws ExcecaoBancoIndisponivel{
   return modeloGerente.retorneProcessos(idOrganizacao);
}

public List retorneConhecimentos(int idOrganizacao) throws ExcecaoBancoIndisponivel{
   return modeloGerente.retorneConhecimentos(idOrganizacao);
}

public List retorneRecursos(int idOrganizacao) throws ExcecaoBancoIndisponivel{
   return modeloGerente.retorneRecursos(idOrganizacao);
}

public Organizacao retorneOrganizacao(int idProjeto) throws ExcecaoBancoIndisponivel{
   return modeloGerente.retorneOrganizacao(idProjeto);
}

public List retorneProjetos() throws ExcecaoBancoIndisponivel{
   return modeloGerente.retorneProjetos();
}

public List retorneTarefas(int idProcesso) throws ExcecaoBancoIndisponivel{
   return modeloGerente.retorneTarefas(idProcesso);
}
```

```
    public void setIdProjetoAtual(int idProjeto){
        modeloGerente.setIdProjetoAtual(idProjeto);
    }
    public int getIdProjetoAtual(){
        return modeloGerente.getIdProjetoAtual();
    }

    public void setIdOrganizacaoAtual(int idOrganizacao){
        modeloGerente.setIdOrganizacaoAtual(idOrganizacao);
    }

    public int getIdOrganizacaoAtual(){
        return modeloGerente.getIdOrganizacaoAtual();
    }

}


package commonkads.logica.modelo;


public class Agente {

    public Conhecimento m_Conhecimento;
    public Tarefa m_Tarefa;
    private int idAgente;
    private String nome;
    private boolean humano;
    private String organizacao;
    private String competencias;
    private String responsabilidades;
    private String restricoes;


    public Agente(){

    }
    public int  getidAgente(){
        return idAgente;
    }

    public void setidAgente(int  idAgente){
        this.idAgente = idAgente;
    }

    public String getNome(){
        return nome;
    }

    public void setNome(String nome){
        this.nome = nome;
    }

    public boolean getHumano(){
        return humano;
    }
```

```
    public void setHumano(boolean humano){
        this.humano = humano;
    }

    public String getOrganizacao(){
        return organizacao;
    }

    public void setOrganizacao(String organizacao){
        this.organizacao = organizacao;
    }

    public String getCompetencias(){
        return competencias;
    }

    public void setCompetencias(String competencias){
        this.competencias = competencias;
    }

    public String getResponsabilidades(){
        return responsabilidades;
    }

    public void setResponsabilidades(String responsabilidades){
        this.responsabilidades = responsabilidades;
    }

    public String getRestricoes(){
        return restricoes;
    }

    public void setRestricoes(String restricoes){
        this.restricoes = restricoes;
    }


}



package commonkads.logica.modelo;


/**
 * @version 1.0
 * @created 09-out-2005 17:43:59
 */
public class AgenteTarefa {

        public AgenteTarefa(){

        }

        public void finalize() throws Throwable {
```

```
            }
}

package commonkads.logica.modelo;

public class Conhecimento {

    public Tarefa m_Tarefa;
    private int idAgente;
    private int  idConhecimento;
    private String descricao;

    /**
     * Holds value of property idOrganizacao.
     */
    private int idOrganizacao;


    public Conhecimento(){

    }

    public Conhecimento(String descricao){
        this.setDescricao(descricao);
    }

    public int getIdAgente(){
        return idAgente;
    }

    public void setIdAgente(int idagente){
        this.idAgente = idAgente;
    }

    public int  getIdConhecimento(){
        return idConhecimento;
    }

    public void setIdConhecimento(int  idConhecimento){
        this.idConhecimento = idConhecimento;
    }

    public String getDescricao(){
        return descricao;
    }

    public void setDescricao(String descricao){
        this.descricao = descricao;
    }

    /**
     * Getter for property idOrganizacao.
     * @return Value of property idOrganizacao.
```

```
     */
    public int getIdOrganizacao() {

        return this.idOrganizacao;
    }

    /**
     * Setter for property idOrganizacao.
     * @param idOrganizacao New value of property idOrganizacao.
     */
    public void setIdOrganizacao(int idOrganizacao) {

        this.idOrganizacao = idOrganizacao;
    }



}


package commonkads.logica.modelo;

/**
 * @version 1.0
 * @created 09-out-2005 17:43:59
 */
public class Oportunidade {

        public Oportunidade(){

        }

        public void finalize() throws Throwable {

        }

}


package commonkads.logica.modelo;



public class Organizacao {

    private int idProjeto;
    private int idOrganizacao;
    private String nome;
    private String estrutura;
    private String oportunidades;
    private String missao;
    private String visao;
    private String objetivos;
    private String fatoresExternos;
    private String estrategias;
    private String valores;
```

```
private String arquivoEstrutura;
private String pessoas;
private String conhecimento;
private String culturaPoder;



public Organizacao(){

}

public Organizacao(String nome){
   this.setNome(nome);
}

public int getIdProjeto(){
   return idProjeto;
}

public void setIdProjeto(int idProjeto){
   this.idProjeto = idProjeto;
}

public int getIdOrganizacao(){
   return idOrganizacao;
}

public void setIdOrganizacao(int idOrganizacao){
   this.idOrganizacao = idOrganizacao;
}

public String getNome(){
   return nome;
}

public void setNome(String nome){
   this.nome = nome;
}

public String getEstrutura(){
   return estrutura;
}

public void setEstrutura(String estrutura){
   this.estrutura = estrutura;
}

public String getOportunidades(){
   return oportunidades;
}

public void setOportunidades(String oportunidades){
   this.oportunidades = oportunidades;
}

public String getMissao(){
```

```
    return missao;
}

public void setMissao(String missao){
    this.missao = missao;
}

public String getVisao(){
    return visao;
}

public void setVisao(String visao){
    this.visao = visao;
}

public String getObjetivos(){
    return objetivos;
}

public void setObjetivos(String objetivos){
    this.objetivos = objetivos;
}

public String getFatoresExternos(){
    return fatoresExternos;
}

public void setFatoresExternos(String fatoresExternos){
    this.fatoresExternos = fatoresExternos;
}

public String getEstrategias(){
    return estrategias;
}

public void setEstrategias(String estrategias){
    this.estrategias = estrategias;
}

public String getValores(){
    return valores;
}

public void setValores(String valores){
    this.valores = valores;
}

public String getArquivoEstrutura(){
    return arquivoEstrutura;
}

public void setArquivoEstrutura(String arquivoEstrutura){
    this.arquivoEstrutura = arquivoEstrutura;
}

public String getPessoas(){
```

```
        return pessoas;
    }

    public void setPessoas(String pessoas){
        this.pessoas = pessoas;
    }

    public String getConhecimento(){
        return conhecimento;
    }

    public void setConhecimento(String conhecimento){
        this.conhecimento = conhecimento;
    }

    public String getCulturaPoder(){
        return culturaPoder;
    }

    public void setCulturaPoder(String culturaPoder){
        this.culturaPoder = culturaPoder;
    }

}


package commonkads.logica.modelo;
public class Pessoa {

            public Pessoa(){

            }

            public void finalize() throws Throwable {

            }

}


package commonkads.logica.modelo;


public class Problema {

    public Solucao m_Solucao;


    private int idOrganizacao;
    private int idProblema;
    private String descricao;

    public Problema(){

    }
```

```
    public int getidOrganizacao(){
        return idOrganizacao;
    }

    public void setidOrganizacao(int idOrganizacao){
        this.idOrganizacao = idOrganizacao;
    }

    public int getidProblema(){
        return idProblema;
    }

    public void setidProblema(int idproblema){
        this.idProblema = idProblema;
    }

    public String getDescricao(){
        return descricao;
    }

    public void setDescricao(String descricao){
        this.descricao = descricao;
    }


}

package commonkads.logica.modelo;

import java.util.HashSet;
import java.util.Set;


public class Processo {

    private Set tarefas = new HashSet();
    public Organizacao m_Organizacao;

    private int idProcesso;
    private String nome;
    private int idOrganizacao;

    public Processo(){

    }

    public Processo(String nome){
        this.setNome(nome);
    }

    public int getidProcesso(){
        return idProcesso;
    }

    public void setidProcesso(int idProcesso){
        this.idProcesso = idProcesso;
```

```
    }

    public String getNome(){
        return nome;
    }

    public void setNome(String nome){
        this.nome = nome;
    }

    public int getidOrganizacao(){
        return idOrganizacao;
    }

    public void setidOrganizacao(int idOrganizacao){
        this.idOrganizacao = idOrganizacao;
    }

    public void setTarefas(Set tarefas){
        this.tarefas = tarefas;
    }

    public Set getTarefas(){
        return tarefas;
    }

    public void addTarefa(Tarefa tarefa){
        this.tarefas.add(tarefa);
    }
}


/*
 * Projeto.java
 *
 */

package commonkads.logica.modelo;

/**
 *
 * @author Cleosvaldo Jr.
 */
public class Projeto {

    /**
     * Holds value of property idProjeto.
     */
    private int idProjeto;

    /**
     * Holds value of property nome.
     */
    private String nome;

    /** Creates a new instance of Projeto */
```

```
    public Projeto() {
    }

    /**
     * Getter for property idProjeto.
     * @return Value of property idProjeto.
     */
    public int getIdProjeto() {

        return this.idProjeto;
    }

    /**
     * Setter for property idProjeto.
     * @param idProjeto New value of property idProjeto.
     */
    public void setIdProjeto(int idProjeto) {

        this.idProjeto = idProjeto;
    }

    /**
     * Getter for property nome.
     * @return Value of property nome.
     */
    public String getNome() {

        return this.nome;
    }

    /**
     * Setter for property nome.
     * @param nome New value of property nome.
     */
    public void setNome(String nome) {

        this.nome = nome;
    }


}


package commonkads.logica.modelo;



public class Recurso {

    private int  idRecursos;
    private int idOrganizacao;
    private String descricao;

    public Recurso(){
```

```
    }

    public Recurso(String descricao){
        setDescricao(descricao);
    }

    public int  getidRecursos(){
        return idRecursos;
    }

    public void setidRecursos(int  idRecursos){
        this.idRecursos = idRecursos;
    }

    public int getidOrganizacao(){
        return idOrganizacao;
    }

    public void setidOrganizacao(int idOrganizacao){
        this.idOrganizacao = idOrganizacao;
    }

    public String getDescricao(){
        return descricao;
    }

    public void setDescricao(String descricao){
        this.descricao = descricao;
    }


}


package commonkads.logica.modelo;


public class Solucao {

    private int idProblema;
    private int idSolucao;
    private String descricao;

    public Solucao(){

    }

    public int getidProblema(){
        return idProblema;
    }

    public void setidProblema(int idProblema){
        this.idProblema = idProblema;
    }

    public int getidSolucao(){
```

```
        return idSolucao;
    }

    public void setidSolucao(int idsolucao){
        this.idSolucao = idSolucao;
    }

    public String getDescricao(){
        return descricao;
    }

    public void setDescricao(String descricao){
        this.descricao = descricao;
    }

}


package commonkads.logica.modelo;



public class Tarefa {

    public Conhecimento m_Conhecimento;
    public Agente m_Agente;
    private int idProcesso;
    private int idTarefa;
    private String nome;
    private String objetivos;
    private String fluxo;
    private String objetos;
    private String tempoControle;
    private String recursos;
    private String qualidade;
    private String executadaPor;
    private String localizacao;
    private boolean intensiva;
    private int grauSignificancia;


    public Tarefa(){

    }

    public Tarefa(String nome){
        this.setNome(nome);
    }


    public int getIdProcesso(){
        return idProcesso;
    }

    public void setIdProcesso(int idProcesso){
```

```
    this.idProcesso = idProcesso;
}

public int  getIdTarefa(){
    return idTarefa;
}

public void setIdTarefa(int  idTarefa){
    this.idTarefa = idTarefa;
}

public String getNome(){
    return nome;
}

public void setNome(String nome){
    this.nome = nome;
}

public String getObjetivos(){
    return objetivos;
}

public void setObjetivos(String objetivos){
    this.objetivos = objetivos;
}

public String getFluxo(){
    return fluxo;
}

public void setFluxo(String fluxo){
    this.fluxo = fluxo;
}

public String getObjetos(){
    return objetos;
}

public void setObjetos(String objetos){
    this.objetos = objetos;
}

public String getTempoControle(){
    return tempoControle;
}

public void setTempoControle(String tempoControle){
    this.tempoControle = tempoControle;
}

public String getRecursos(){
    return recursos;
}

public void setRecursos(String recursos){
```

```
    this.recursos = recursos;
}

public String getQualidade(){
    return qualidade;
}

public void setQualidade(String qualidade){
    this.qualidade = qualidade;
}

/**
 * Getter for property executadaPor.
 * @return Value of property executadaPor.
 */
public String getExecutadaPor() {

    return this.executadaPor;
}

/**
 * Setter for property executadaPor.
 * @param executadaPor New value of property executadaPor.
 */
public void setExecutadaPor(String executadaPor) {

    this.executadaPor = executadaPor;
}

/**
 * Getter for property localizacao.
 * @return Value of property localizacao.
 */
public String getLocalizacao() {

    return this.localizacao;
}

/**
 * Setter for property localizacao.
 * @param localizacao New value of property localizacao.
 */
public void setLocalizacao(String localizacao) {

    this.localizacao = localizacao;
}

/**
 * Getter for property intensiva.
 * @return Value of property intensiva.
 */
public boolean isIntensiva() {

    return this.intensiva;
}
```

```
    /**
     * Setter for property intensiva.
     * @param intensiva New value of property intensiva.
     */
    public void setIntensiva(boolean intensiva) {

        this.intensiva = intensiva;
    }

    /**
     * Getter for property grauSignificancia.
     * @return Value of property grauSignificancia.
     */
    public int getGrauSignificancia() {

        return this.grauSignificancia;
    }

    /**
     * Setter for property grauSignificancia.
     * @param grauSignificancia New value of property grauSignificancia.
     */
    public void setGrauSignificancia(int grauSignificancia) {

        this.grauSignificancia = grauSignificancia;
    }
}
```

```
package commonkads.logica.regras;

import commonkads.persistencia.fachada.FachadaPersistencia;

public abstract class ModeloAbstrato {

    FachadaPersistencia fachada;

    /** Creates a new instance of GerenciadorAbstrato */
    public ModeloAbstrato(FachadaPersistencia fachada) {
        this.fachada = fachada;
    }

}
```

```
package commonkads.logica.regras;


import commonkads.logica.modelo.Agente;
import commonkads.logica.modelo.Conhecimento;
import commonkads.logica.modelo.Organizacao;
import commonkads.logica.modelo.Problema;
import commonkads.logica.modelo.Processo;
import commonkads.logica.modelo.Projeto;
import commonkads.logica.modelo.Recurso;
```

```
import commonkads.logica.modelo.Solucao;
import commonkads.logica.modelo.Tarefa;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import commonkads.persistencia.fachada.FachadaPersistencia;
import java.util.List;



public class ModeloGerente extends ModeloAbstrato {

    private int idProjetoAtual;
    private int idOrganizacaoAtual;


    /** Creates a new instance of GerenciadorDeClientes */
    public ModeloGerente(FachadaPersistencia fachada) {
        super(fachada);
    }

    public boolean salveProcesso(Processo processo)throws ExcecaoBancoIndisponivel {
        fachada.salveProcesso(processo);
        return true;
    }


    public boolean salveAgente(Agente agente) throws ExcecaoBancoIndisponivel {
        fachada.salveAgente(agente);
        return true;
    }

    public boolean salveOrganizacao(Organizacao organizacao) throws ExcecaoBancoIndisponivel {
        fachada.salveOrganizacao(organizacao);
        return true;
    }

    public boolean salveConhecimento(Conhecimento conhecimento) throws ExcecaoBancoIndisponivel {
        fachada.salveConhecimento(conhecimento);
        return true;
    }

    public boolean salveProjeto(Projeto projeto) throws ExcecaoBancoIndisponivel {
        fachada.salveProjeto(projeto);
        return true;
    }

    public boolean salveRecurso(Recurso recurso) throws ExcecaoBancoIndisponivel {
        fachada.salveRecurso(recurso);
        return true;
    }


    public boolean salveSolucao(Solucao solucao) throws ExcecaoBancoIndisponivel {
        fachada.salveSolucao(solucao);
        return true;
    }
```

```
public boolean salveTarefa(Tarefa tarefa) throws ExcecaoBancoIndisponivel {
   fachada.salveTarefa(tarefa);
   return true;
}

public boolean salveProblema(Problema problema) throws ExcecaoBancoIndisponivel {
   fachada.salveProblema(problema);
   return true;
}

public Organizacao crieOrganizacao(String nome){
   return new Organizacao(nome);
}

public List retorneProcessos(int idOrganizacao) throws ExcecaoBancoIndisponivel {
   return fachada.retorneProcessos(idOrganizacao);
}

public List retorneRecursos(int idOrganizacao) throws ExcecaoBancoIndisponivel {
   return fachada.retorneRecursos(idOrganizacao);
}

public Organizacao retorneOrganizacao(int idProjeto) throws ExcecaoBancoIndisponivel {
   List lstOrganizacao = fachada.retorneOrganizacao(idProjeto);

   this.setIdOrganizacaoAtual(((Organizacao)lstOrganizacao.get(0)).getIdOrganizacao());

   return ((Organizacao)lstOrganizacao.get(0));

}

public List retorneProjetos() throws ExcecaoBancoIndisponivel {
   return fachada.retorneProjetos();
}

public List retorneConhecimentos(int idOrganizacao) throws ExcecaoBancoIndisponivel {
   return fachada.retorneConhecimentos(idOrganizacao);
}

public List retorneTarefas(int idProcesso) throws ExcecaoBancoIndisponivel {
   return fachada.retorneTarefas(idProcesso);
}

/**
 * Getter for property idProjetoAtual.
 * @return Value of property idProjetoAtual.
 */
public int getIdProjetoAtual() {

   return this.idProjetoAtual;
}

/**
 * Setter for property idProjetoAtual.
 * @param idProjetoAtual New value of property idProjetoAtual.
 */
```

```
    public void setIdProjetoAtual(int idProjetoAtual) {

        this.idProjetoAtual = idProjetoAtual;
    }

    /**
     * Getter for property idOrganizacaoAtual.
     * @return Value of property idOrganizacaoAtual.
     */
    public int getIdOrganizacaoAtual() {

        return this.idOrganizacaoAtual;
    }

    /**
     * Setter for property idOrganizacaoAtual.
     * @param idOrganizacaoAtual New value of property idOrganizacaoAtual.
     */
    public void setIdOrganizacaoAtual(int idOrganizacaoAtual) {

        this.idOrganizacaoAtual = idOrganizacaoAtual;
    }

}
```

- Persistência

```
/*
 * AgenteDAO.java
 *
 */

package commonkads.persistencia.dao;

import commonkads.logica.modelo.Agente;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

/**
 *
 * @author Cleosvaldo Jr.
 */
public class AgenteDAO {
    SessionFactory sessionFactory;
    /** Creates a new instance of AgenteDAO */
    public AgenteDAO(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }
```

```
    public void salve(Agente novoAgente) throws ExcecaoBancoIndisponivel {
        try {
            Session sessao = sessionFactory.openSession();
            Transaction tx = sessao.beginTransaction();

            sessao.save(novoAgente);
            tx.commit();
            sessao.close();
        } catch (HibernateException e) {
            throw new ExcecaoBancoIndisponivel(e);
        }
    }


}


package commonkads.persistencia.dao;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

/**
 *
 * @author Cleosvaldo
 */
public class BancoDAO{
    SessionFactory sessionFactory;

    ProcessoDAO processoDAO;
    ConhecimentoDAO conhecimentoDAO;
    OrganizacaoDAO organizacaoDAO;
    ProblemaDAO problemaDAO;
    ProjetoDAO projetoDAO;
    RecursoDAO recursoDAO;
    TarefaDAO tarefaDAO;
    AgenteDAO agenteDAO;
    SolucaoDAO solucaoDAO;


    public BancoDAO() {
        sessionFactory = definaSessionFactory();

        processoDAO     = new ProcessoDAO(sessionFactory);
        projetoDAO      = new ProjetoDAO(sessionFactory);
        organizacaoDAO  = new OrganizacaoDAO(sessionFactory);
        problemaDAO     = new ProblemaDAO(sessionFactory);
        conhecimentoDAO = new ConhecimentoDAO(sessionFactory);
        recursoDAO      = new RecursoDAO(sessionFactory);
        tarefaDAO       = new TarefaDAO(sessionFactory);
        agenteDAO       = new AgenteDAO(sessionFactory);
        solucaoDAO      = new SolucaoDAO(sessionFactory);
    }

    public ProcessoDAO getProcessoDAO() {
```

```
        return processoDAO;
    }

    public AgenteDAO getAgenteDAO(){
        return agenteDAO;
    }

    public ConhecimentoDAO getConhecimentoDAO(){
        return conhecimentoDAO;
    }

    public OrganizacaoDAO getOrganizacaoDAO(){
        return organizacaoDAO;
    }

    public ProblemaDAO getProblemaDAO(){
        return problemaDAO;
    }

    public ProjetoDAO getProjetoDAO(){
        return projetoDAO;
    }

    public RecursoDAO getRecursoDAO(){
        return recursoDAO;
    }

    public SolucaoDAO getSolucaoDAO(){
        return solucaoDAO;
    }

    public TarefaDAO getTarefaDAO(){
        return tarefaDAO;
    }


    private SessionFactory definaSessionFactory() {
        // A configuração é feita lendo o arquivo hibernate.cfg.xml que deve
        // estar no classpath
        SessionFactory sf = new Configuration().configure().buildSessionFactory();

        return sf;
    }


}


/*
 * Conhecimento.java
 *
 */

package commonkads.persistencia.dao;

import commonkads.logica.modelo.Conhecimento;
```

```
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.List;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

/**
 *
 * @author Cleosvaldo Jr.
 */
public class ConhecimentoDAO {
    SessionFactory sessionFactory;
    /** Creates a new instance of Conhecimento */
    public ConhecimentoDAO(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }


    public void salve(Conhecimento novoConhecimento) throws ExcecaoBancoIndisponivel {
        try {
            Session sessao = sessionFactory.openSession();
            Transaction tx = sessao.beginTransaction();

            sessao.save(novoConhecimento);
            tx.commit();
            sessao.close();
        } catch (HibernateException e) {
            throw new ExcecaoBancoIndisponivel(e);
        }
    }

       public List retorneConhecimentos(int idOrganizacao) throws ExcecaoBancoIndisponivel {
        String id = new Integer(idOrganizacao).toString();
        Session sessao = sessionFactory.openSession();
        List result = sessao.createQuery("from Conhecimento where idOrganizacao = "+id+ " order by descricao
asc").list();

        return result;

    }


}


package commonkads.persistencia.dao;

import commonkads.logica.modelo.Organizacao;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.List;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
```

```
public class OrganizacaoDAO {
    SessionFactory sessionFactory;
    /** Creates a new instance of OrganizacaoDAO */
    public OrganizacaoDAO(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    public void salve(Organizacao novaOrganizacao) throws ExcecaoBancoIndisponivel {
        try {
            Session sessao = sessionFactory.openSession();
            Transaction tx = sessao.beginTransaction();

            sessao.update(novaOrganizacao);
            tx.commit();
            sessao.close();
        } catch (HibernateException e) {
            throw new ExcecaoBancoIndisponivel(e);
        }
    }

    public List retorneOrganizacao(int idProjeto) throws ExcecaoBancoIndisponivel {
        String id = new Integer(idProjeto).toString();
        Session sessao = sessionFactory.openSession();
        List result = sessao.createQuery("from Organizacao where idProjeto = "+id+ " order by nome asc").list();

        return result;

    }

}


/*
 * ProblemaDAO.java
 *
 */

package commonkads.persistencia.dao;

import commonkads.logica.modelo.Problema;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

/**
 *
 * @author Cleosvaldo Jr.
 */
public class ProblemaDAO {
    SessionFactory sessionFactory;

    /** Creates a new instance of ProblemaDAO */
    public ProblemaDAO(SessionFactory sessionFactory) {
```

```
        this.sessionFactory = sessionFactory;
    }

    public void salve(Problema novoProblema) throws ExcecaoBancoIndisponivel {
        try {
            Session sessao = sessionFactory.openSession();
            Transaction tx = sessao.beginTransaction();

            sessao.save(novoProblema);
            tx.commit();
            sessao.close();
        } catch (HibernateException e) {
            throw new ExcecaoBancoIndisponivel(e);
        }
    }

}


package commonkads.persistencia.dao;


import commonkads.logica.modelo.Processo;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.Iterator;
import java.util.List;
import org.hibernate.Criteria;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import org.hibernate.Transaction;
import org.hibernate.criterion.Restrictions;


public class ProcessoDAO {

    SessionFactory sessionFactory;


    public ProcessoDAO(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }
    public Processo encontrePorNome(String nome) throws ExcecaoBancoIndisponivel {
        Processo processo = null;
        try {
            Session sessao = sessionFactory.openSession();
            Criteria criterio = sessao.createCriteria(Processo.class);
            Iterator<Processo> it;

            criterio.add(Restrictions.like("nome",nome));
            it = criterio.list().iterator();
            if (it.hasNext())
                processo = it.next();
            sessao.close();
        } catch (HibernateException e) {
```

```
            throw new ExcecaoBancoIndisponivel(e);
        }
        return processo;
    }

    public List retorneProcessos(int idOrganizacao) throws ExcecaoBancoIndisponivel {
        String id = new Integer(idOrganizacao).toString();
        Session sessao = sessionFactory.openSession();
        List result = sessao.createQuery("from Processo where idOrganizacao = "+id+ " order by nome asc").list();

        return result;

    }
    public void salve(Processo novoProcesso) throws ExcecaoBancoIndisponivel {
        try {
            Session sessao = sessionFactory.openSession();
            Transaction tx = sessao.beginTransaction();

            sessao.save(novoProcesso);
            tx.commit();
            sessao.close();
        } catch (HibernateException e) {
            throw new ExcecaoBancoIndisponivel(e);
        }
    }

    public void atualize(Processo processo) throws ExcecaoBancoIndisponivel {
        try {
            Session sessao = sessionFactory.openSession();
            Transaction tx = sessao.beginTransaction();

            sessao.update(processo);
            tx.commit();
            sessao.close();
        } catch (HibernateException e) {
            throw new ExcecaoBancoIndisponivel(e);
        }

    }

}


/*
 * ProjetoDAO.java
 *
 * Created on 13 de Outubro de 2005, 22:19
 */

package commonkads.persistencia.dao;

import commonkads.logica.modelo.Projeto;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.List;
import org.hibernate.HibernateException;
import org.hibernate.Session;
```

```
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

/**
 *
 * @author JR
 */
public class ProjetoDAO {
   SessionFactory sessionFactory;
   /** Creates a new instance of ProjetoDAO */
   public ProjetoDAO(SessionFactory sessionFactory) {
      this.sessionFactory = sessionFactory;
   }

   public void salve(Projeto novoProjeto) throws ExcecaoBancoIndisponivel {
      try {
         Session sessao = sessionFactory.openSession();
         Transaction tx = sessao.beginTransaction();

         sessao.save(novoProjeto);
         tx.commit();
         sessao.close();
      } catch (HibernateException e) {
         throw new ExcecaoBancoIndisponivel(e);
      }
   }

      public List retorneProjetos() throws ExcecaoBancoIndisponivel {

      Session sessao = sessionFactory.openSession();
      List result = sessao.createQuery("from Projeto order by nome asc").list();

      return result;
   }
}


/*
 * RecursoDAO.java
 *
 */

package commonkads.persistencia.dao;

import commonkads.logica.modelo.Recurso;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.List;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

/**
 *
 * @author Cleosvaldo Jr.
 */
```

```
public class RecursoDAO {
   SessionFactory sessionFactory;
   /** Creates a new instance of RecursoDAO */
   public RecursoDAO(SessionFactory sessionFactory) {
      this.sessionFactory = sessionFactory;
   }



   public void salve(Recurso novaRecurso) throws ExcecaoBancoIndisponivel {

      try {
          Session sessao = sessionFactory.openSession();
          Transaction tx = sessao.beginTransaction();

        try {
          sessao.saveOrUpdate(novaRecurso);
          tx.commit();
         }catch (HibernateException e) {
          throw new ExcecaoBancoIndisponivel(e);
      }
          sessao.close();
      } catch (HibernateException e) {
          throw new ExcecaoBancoIndisponivel(e);
      }

   }

      public List retorneRecursos(int idOrganizacao) throws ExcecaoBancoIndisponivel {
      String id = new Integer(idOrganizacao).toString();
      Session sessao = sessionFactory.openSession();
      List result = sessao.createQuery("from Recurso where idOrganizacao = "+id+ " ").list();

      return result;

   }


}


/*
 * SolucaoDAO.java
 *
 */

package commonkads.persistencia.dao;

import commonkads.logica.modelo.Solucao;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
```

```
public class SolucaoDAO {
   SessionFactory sessionFactory;
   /** Creates a new instance of SolucaoDAO */
   public SolucaoDAO(SessionFactory sessionFactory) {
      this.sessionFactory = sessionFactory;
   }

   public void salve(Solucao novaSolucao) throws ExcecaoBancoIndisponivel {
      try {
         Session sessao = sessionFactory.openSession();
         Transaction tx = sessao.beginTransaction();

         sessao.save(novaSolucao);
         tx.commit();
         sessao.close();
      } catch (HibernateException e) {
         throw new ExcecaoBancoIndisponivel(e);
      }
   }


}


/*
 * TarefaDAO.java
 */

package commonkads.persistencia.dao;

import commonkads.logica.modelo.Tarefa;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.List;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

public class TarefaDAO {

   SessionFactory sessionFactory;


   /** Creates a new instance of TarefaDAO */
   public TarefaDAO(SessionFactory sessionFactory) {
      this.sessionFactory = sessionFactory;
   }

   public void salve(Tarefa novaTarefa) throws ExcecaoBancoIndisponivel {
      try {
         Session sessao = sessionFactory.openSession();
         Transaction tx = sessao.beginTransaction();

         sessao.save(novaTarefa);
         tx.commit();
         sessao.close();
```

```
        } catch (HibernateException e) {
            throw new ExcecaoBancoIndisponivel(e);
        }
    }

    public List retorneTarefas(int idProcesso) throws ExcecaoBancoIndisponivel {

        String id = new Integer(idProcesso).toString();

        Session sessao = sessionFactory.openSession();
        List result = sessao.createQuery("from Tarefa where idProcesso= "+id+ " order by nome asc").list();

        return result;

    }



}


package commonkads.persistencia.excecoes;

public class ExcecaoBancoIndisponivel extends Exception {

    /** Creates a new instance of ExcecaoBancoIndisponivel */
    public ExcecaoBancoIndisponivel(Throwable e) {
        super(e);
    }

}

package commonkads.persistencia.fachada;



public class FabricaDeFachadaPersistencia {

    /**
     * Impede a instanciacao.
     */
    private FabricaDeFachadaPersistencia() {
    }

    public static FachadaPersistencia crie() {
        return new MinhaFachadaPersistencia();
    }
}


/*
 * FachadaPersistencia.java
 *
 * Created on May 27, 2005, 7:25 PM
 */
```

```
package commonkads.persistencia.fachada;



import commonkads.logica.modelo.Agente;
import commonkads.logica.modelo.Conhecimento;
import commonkads.logica.modelo.Organizacao;
import commonkads.logica.modelo.Problema;
import commonkads.logica.modelo.Processo;
import commonkads.logica.modelo.Projeto;
import commonkads.logica.modelo.Recurso;
import commonkads.logica.modelo.Solucao;
import commonkads.logica.modelo.Tarefa;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.List;

/**
 * Serviços oferecidos pela camada de persistência
 * @author Cleosvaldo Jr
 */
public interface FachadaPersistencia {


    public void salveProcesso(Processo processo) throws ExcecaoBancoIndisponivel;
    public void salveAgente(Agente agente) throws ExcecaoBancoIndisponivel;
    public void salveConhecimento(Conhecimento conhecimento) throws ExcecaoBancoIndisponivel;
    public void salveOrganizacao(Organizacao organizacao) throws ExcecaoBancoIndisponivel;
    public void salveProblema(Problema problema) throws ExcecaoBancoIndisponivel;
    public void salveProjeto(Projeto projeto) throws ExcecaoBancoIndisponivel;
    public void salveRecurso(Recurso recurso) throws ExcecaoBancoIndisponivel;
    public void salveSolucao(Solucao solucao) throws ExcecaoBancoIndisponivel;
    public void salveTarefa(Tarefa tarefa) throws ExcecaoBancoIndisponivel;


    public List retorneProcessos(int idOrganizacao) throws ExcecaoBancoIndisponivel;
    public List retorneRecursos(int idOrganizacao) throws ExcecaoBancoIndisponivel;
    public List retorneOrganizacao(int idProjeto) throws ExcecaoBancoIndisponivel;
    public List retorneProjetos() throws ExcecaoBancoIndisponivel;
    public List retorneConhecimentos(int idOrganizacao) throws ExcecaoBancoIndisponivel;
    public List retorneTarefas(int idProcesso) throws ExcecaoBancoIndisponivel;


}


/*
 * MinhaFachadaPersistencia.java
 *
 */

package commonkads.persistencia.fachada;


import commonkads.logica.modelo.Agente;
import commonkads.logica.modelo.Conhecimento;
import commonkads.logica.modelo.Organizacao;
```

```
import commonkads.logica.modelo.Problema;
import commonkads.logica.modelo.Processo;
import commonkads.logica.modelo.Projeto;
import commonkads.logica.modelo.Recurso;
import commonkads.logica.modelo.Solucao;
import commonkads.logica.modelo.Tarefa;
import commonkads.persistencia.dao.BancoDAO;
import commonkads.persistencia.excecoes.ExcecaoBancoIndisponivel;
import java.util.List;

/**
 *
 * @author Cleosvaldo Jr.
 */
public class MinhaFachadaPersistencia implements FachadaPersistencia {

    BancoDAO banco;

    /** Creates a new instance of MinhaFachadaPersistencia */
    public MinhaFachadaPersistencia() {
        banco = new BancoDAO();
    }

    public void salveProcesso(Processo processo)throws ExcecaoBancoIndisponivel{
        banco.getProcessoDAO().salve(processo);
    }



    public void salveAgente(Agente agente) throws ExcecaoBancoIndisponivel{
        banco.getAgenteDAO().salve(agente);
    }
    public void salveConhecimento(Conhecimento conhecimento) throws ExcecaoBancoIndisponivel{
        banco.getConhecimentoDAO().salve(conhecimento);
    }
    public void salveOrganizacao(Organizacao organizacao) throws ExcecaoBancoIndisponivel{
        banco.getOrganizacaoDAO().salve(organizacao);
    }
    public void salveProblema(Problema problema) throws ExcecaoBancoIndisponivel{
        banco.getProblemaDAO().salve(problema);
    }
    public void salveProjeto(Projeto projeto) throws ExcecaoBancoIndisponivel{
        banco.getProjetoDAO().salve(projeto);
    }
    public void salveRecurso(Recurso recurso) throws ExcecaoBancoIndisponivel{
        banco.getRecursoDAO().salve(recurso);
    }
    public void salveSolucao(Solucao solucao) throws ExcecaoBancoIndisponivel{
        banco.getSolucaoDAO().salve(solucao);
    }

    public void salveTarefa(Tarefa tarefa) throws ExcecaoBancoIndisponivel{
        banco.getTarefaDAO().salve(tarefa);
    }

    public List retorneProcessos(int idOrganizacao) throws ExcecaoBancoIndisponivel{
```

```
        return banco.getProcessoDAO().retorneProcessos(idOrganizacao);
    }

    public List retorneRecursos(int idOrganizacao) throws ExcecaoBancoIndisponivel{
        return banco.getRecursoDAO().retorneRecursos(idOrganizacao);
    }

    public List retorneOrganizacao(int idProjeto) throws ExcecaoBancoIndisponivel{
        return banco.getOrganizacaoDAO().retorneOrganizacao(idProjeto);
    }
    public List retorneProjetos() throws ExcecaoBancoIndisponivel{
        return banco.getProjetoDAO().retorneProjetos();
    }

    public List retorneConhecimentos(int idOrganizacao) throws ExcecaoBancoIndisponivel{
        return banco.getConhecimentoDAO().retorneConhecimentos(idOrganizacao);
    }

    public List retorneTarefas(int idProcesso) throws ExcecaoBancoIndisponivel {
        return banco.getTarefaDAO().retorneTarefas(idProcesso);
    }
}
```

- Arquivos Hibernate

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<!--
    Document   : hibernate.cfg.xml
 -->

<hibernate-configuration>

    <session-factory>
        <!-- Parâmetros de configuração JDBC -->
        <property name="connection.driver_class">sun.jdbc.odbc.JdbcOdbcDriver</property>
        <property name="connection.url">jdbc:odbc:banco</property>
        <property name="connection.username">sa</property>
        <property name="connection.password">sa</property>

        <!-- Definição de pool de conexões -->
        <!-- Usar pool de conexões do Hibernate - só para pequenos testes!
        <property name="connection.pool_size">1</property>
        -->
        <!-- Pool de conexões JDBC usando c3p0 -->
        <property name="c3p0.min_size">5</property>
        <property name="c3p0.max_size">20</property>
        <property name="c3p0.timeout">1800</property>
        <property name="c3p0.max_statements">50</property>
```

```
        <!-- Definição de qual banco de dados será usado -->
        <property name="dialect">org.hibernate.dialect.SQLServerDialect</property>

        <!-- Mostra SQL gerados (bom para debug) -->
        <property name="show_sql">false</property>

        <!-- Geração e criação automática de esquema relacional no banco.
        <property name="hbm2ddl.auto">create</property>
        -->

        <!-- Relação de classes que serão mapeadas para o banco -->
     <mapping resource="commonkads/logica/modelo/processo.hbm.xml"/>
     <mapping resource="commonkads/logica/modelo/tarefa.hbm.xml"/>
     <mapping resource="commonkads/logica/modelo/conhecimento.hbm.xml"/>
     <mapping resource="commonkads/logica/modelo/organizacao.hbm.xml"/>
     <mapping resource="commonkads/logica/modelo/recurso.hbm.xml"/>
     <mapping resource="commonkads/logica/modelo/problema.hbm.xml"/>
     <mapping resource="commonkads/logica/modelo/solucao.hbm.xml"/>
    <mapping resource="commonkads/logica/modelo/projeto.hbm.xml"/>
    </session-factory>
</hibernate-configuration>



<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

    <class name="Agente" table="Agentes">
        <id name="idAgente" column="idAgente">
            <generator class="native"/>
        </id>

                        <property
                            name="Nome"
                            column="Nome"/>
                        <property
                            name="Humano"
                            column="Humano"/>
                        <property
                            name="Organizacao"
                            column="Organizacao"/>
                        <property
                            name="Competencias"
                            column="Competencias"/>
                        <property
                            name="Responsabilidades"
                            column="Responsabilidades"/>
                        <property
                            name="Restricoes"
                            column="Restricoes"/>
```

```
   </class>

</hibernate-mapping>


<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

   <class name="AgenteTarefa" table="AgenteTarefa">
      <id name="idProcesso" column="idProcesso">
         <generator class="native"/>
      </id>

            <property
                  name="idAgente"
                  column="idAgente"/>
                        <property
                  name="idTarefa"
                  column="idTarefa"/>

   </class>

</hibernate-mapping>



<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

   <class name="Conhecimento" table="Conhecimentos">

      <id name="idConhecimento" column="idConhecimento">
         <generator class="native"/>
      </id>

      <property
         name="idAgente"
         column="idAgente"/>
      <property
         name="Descricao"
         column="Descricao"/>

      <property
         name="idOrganizacao"
         column="idOrganizacao"/>
```

```
   </class>

</hibernate-mapping>



<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

   <class name="Organizacao" table="Organizacoes">
      <id name="idOrganizacao" column="idOrganizacao">
         <generator class="native"/>
      </id>

                  <property
                     name="Nome"
                     column="Nome"/>
                  <property
                     name="Estrutura"
                     column="Estrutura"/>
                  <property
                     name="Oportunidades"
                     column="Oportunidades"/>
                  <property
                     name="Missao"
                     column="Missao"/>
                  <property
                     name="Visao"
                     column="Visao"/>
                  <property
                     name="Objetivos"
                     column="Objetivos"/>
                  <property
                     name="FatoresExternos"
                     column="FatoresExternos"/>
                  <property
                     name="Estrategias"
                     column="Estrategias"/>
                  <property
                     name="Valores"
                     column="Valores"/>
                  <property
                     name="ArquivoEstrutura"
                     column="ArquivoEstrutura"/>
                  <property
                     name="Pessoas"
                     column="Pessoas"/>
                  <property
                     name="Conhecimento"
                     column="Conhecimento"/>
```

```
                    <property
                       name="CulturaPoder"
                       column="CulturaPoder"/>

   </class>

</hibernate-mapping>


<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

   <class name="Problema" table="Problemas">
      <id name="idProblema" column="idProblema">
         <generator class="native"/>
      </id>

                    <property
                       name="idOrganizacao"
                       column="idOrganizacao"/>

                    <property
                       name="Descricao"
                       column="Descricao"/>

   </class>

</hibernate-mapping>


<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

   <class name="Processo" table="Processos">
      <id name="idProcesso" column="idProcesso">
         <generator class="native"/>
      </id>

      <property
         name="nome"
         column="nome"/>

      <property
         name="idOrganizacao"
         column="idOrganizacao"/>
```

```
        <set
            name="tarefas"
            inverse="true"
            cascade="save-update">
            <key column="idProcesso"/>
            <one-to-many class="Tarefa"/>
        </set>


    </class>

</hibernate-mapping>


<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

    <class name="Projeto" table="Projetos">
        <id name="idProjeto" column="idProjeto">
            <generator class="native"/>
        </id>

        <property
            name="nome"
            column="nome"/>

    </class>

</hibernate-mapping>



<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

    <class name="Recurso" table="Recursos">
        <id name="idRecursos" column="idRecursos">
            <generator class="native"/>
        </id>

                    <property
                        name="idOrganizacao"
                        column="idOrganizacao"/>
                    <property
                        name="Descricao"
                        column="Descricao"/>
```

```
   </class>

</hibernate-mapping>


<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

   <class name="Solucao" table="Solucoes">
      <id name="idSolucao" column="idSolucao">
         <generator class="native"/>
      </id>

                  <property
                     name="idProblema"
                     column="idProblema"/>
                  <property
                     name="Descricao"
                     column="Descricao"/>

   </class>

</hibernate-mapping>


<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

   <class name="Tarefa" table="Tarefas">
      <id name="idTarefa" column="idTarefa">
         <generator class="native"/>
      </id>

                  <property
                     name="idProcesso"
                     column="idProcesso"/>
                  <property
                     name="Nome"
                     column="Nome"/>
                  <property
                     name="Objetivos"
                     column="Objetivos"/>
                  <property
                     name="Fluxo"
                     column="Fluxo"/>
```

```
                    <property
                       name="Objetos"
                       column="Objetos"/>
                    <property
                       name="TempoControle"
                       column="TempoControle"/>
                    <property
                       name="Recursos"
                       column="Recursos"/>
                    <property
                       name="Qualidade"
                       column="Qualidade"/>

   </class>

</hibernate-mapping>


<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">


<hibernate-mapping package="commonkads.logica.modelo">

   <class name="TarefaConhecimento" table="TarefaConhecimento">
      <id name="idProcesso" column="idProcesso">
         <generator class="native"/>
      </id>

            <property
                       name="idTarefa"
                       column="idTarefa"/>
                              <property
                       name="idConhecimento"
                       column="idConhecimento"/>

   </class>

</hibernate-mapping>
```

– Script do Banco de Dados

```
IF EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'commonKADS')
          DROP DATABASE [commonKADS]
GO

CREATE DATABASE [commonKADS]  ON (NAME = N'commonKADS_Data', FILENAME = N'C:\Arquivos de
programas\Microsoft SQL Server\MSSQL\data\commonKADS_Data.MDF' , SIZE = 2, FILEGROWTH = 10%)
LOG ON (NAME = N'commonKADS_Log', FILENAME = N'C:\Arquivos de programas\Microsoft SQL
Server\MSSQL\data\commonKADS_Log.LDF' , SIZE = 1, FILEGROWTH = 10%)
 COLLATE SQL_Latin1_General_CP1_CI_AI
GO
```

```
exec sp_dboption N'commonKADS', N'autoclose', N'true'
GO

exec sp_dboption N'commonKADS', N'bulkcopy', N'false'
GO

exec sp_dboption N'commonKADS', N'trunc. log', N'true'
GO

exec sp_dboption N'commonKADS', N'torn page detection', N'true'
GO

exec sp_dboption N'commonKADS', N'read only', N'false'
GO

exec sp_dboption N'commonKADS', N'dbo use', N'false'
GO

exec sp_dboption N'commonKADS', N'single', N'false'
GO

exec sp_dboption N'commonKADS', N'autoshrink', N'true'
GO

exec sp_dboption N'commonKADS', N'ANSI null default', N'false'
GO

exec sp_dboption N'commonKADS', N'recursive triggers', N'false'
GO

exec sp_dboption N'commonKADS', N'ANSI nulls', N'false'
GO

exec sp_dboption N'commonKADS', N'concat null yields null', N'false'
GO

exec sp_dboption N'commonKADS', N'cursor close on commit', N'false'
GO

exec sp_dboption N'commonKADS', N'default to local cursor', N'false'
GO

exec sp_dboption N'commonKADS', N'quoted identifier', N'false'
GO

exec sp_dboption N'commonKADS', N'ANSI warnings', N'false'
GO

exec sp_dboption N'commonKADS', N'auto create statistics', N'true'
GO

exec sp_dboption N'commonKADS', N'auto update statistics', N'true'
GO

use [commonKADS]
GO
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_AgenteTarefa_Agentes]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[AgenteTarefa] DROP CONSTRAINT FK_AgenteTarefa_Agentes
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Conhecimentos_Agentes]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Conhecimentos] DROP CONSTRAINT FK_Conhecimentos_Agentes
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_ItensInformacao_Agentes]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[ItensInformacao] DROP CONSTRAINT FK_ItensInformacao_Agentes
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_ItensInformacao_Agentes1]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[ItensInformacao] DROP CONSTRAINT FK_ItensInformacao_Agentes1
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Transacao_Agentes]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Transacao] DROP CONSTRAINT FK_Transacao_Agentes
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Transacao_Agentes1]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Transacao] DROP CONSTRAINT FK_Transacao_Agentes1
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_TarefaConhecimento_Conhecimentos]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[TarefaConhecimento] DROP CONSTRAINT FK_TarefaConhecimento_Conhecimentos
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Problemas_Organizacoes]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Problemas] DROP CONSTRAINT FK_Problemas_Organizacoes
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Processos_Organizacoes]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Processos] DROP CONSTRAINT FK_Processos_Organizacoes
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Recursos_Organizacoes]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Recursos] DROP CONSTRAINT FK_Recursos_Organizacoes
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Solucoes_Problemas]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Solucoes] DROP CONSTRAINT FK_Solucoes_Problemas
GO
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Tarefas_Processos]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Tarefas] DROP CONSTRAINT FK_Tarefas_Processos
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Organizacoes_Projetos]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Organizacoes] DROP CONSTRAINT FK_Organizacoes_Projetos
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_AgenteTarefa_Tarefas]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[AgenteTarefa] DROP CONSTRAINT FK_AgenteTarefa_Tarefas
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_TarefaConhecimento_Tarefas]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[TarefaConhecimento] DROP CONSTRAINT FK_TarefaConhecimento_Tarefas
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_ItensInformacao_Transacao]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[ItensInformacao] DROP CONSTRAINT FK_ItensInformacao_Transacao
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[AgenteTarefa]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[AgenteTarefa]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Agentes]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Agentes]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Conhecimentos]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Conhecimentos]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[ItensInformacao]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[ItensInformacao]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Organizacoes]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Organizacoes]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Problemas]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Problemas]
GO
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Processos]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Processos]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Projetos]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Projetos]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Recursos]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Recursos]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Solucoes]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Solucoes]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[TarefaConhecimento]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[TarefaConhecimento]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Tarefas]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Tarefas]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Transacao]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Transacao]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[mensagens]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[mensagens]
GO

CREATE TABLE [dbo].[AgenteTarefa] (
        [idAgente] [int] NOT NULL ,
        [idTarefa] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Agentes] (
        [idAgente] [int] IDENTITY (1, 1) NOT NULL ,
        [Nome] [char] (40) COLLATE SQL_Latin1_General_CP1_CI_AI NOT NULL ,
        [Humano] [bit] NULL ,
        [Organizacao] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
        [Competencias] [varchar] (300) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
        [Responsabilidades] [varchar] (300) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
        [Restricoes] [varchar] (300) COLLATE SQL_Latin1_General_CP1_CI_AI NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[Conhecimentos] (
	[idAgente] [int] NULL ,
	[idConhecimento] [int] IDENTITY (1, 1) NOT NULL ,
	[Descricao] [char] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[idOrganizacao] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[ItensInformacao] (
	[idItem] [int] NOT NULL ,
	[idTransacao] [int] NOT NULL ,
	[idAgenteEnvia] [int] NOT NULL ,
	[idAgenteRecebe] [int] NOT NULL ,
	[itensInformacao] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[mensagem] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[controle] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Organizacoes] (
	[IdProjeto] [int] NOT NULL ,
	[IdOrganizacao] [int] IDENTITY (1, 1) NOT NULL ,
	[Nome] [varchar] (60) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Estrutura] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Oportunidades] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Missao] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Visao] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Objetivos] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[FatoresExternos] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Estrategias] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Valores] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[ArquivoEstrutura] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Pessoas] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Conhecimento] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[CulturaPoder] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Problemas] (
	[idOrganizacao] [int] NULL ,
	[idProblema] [int] NOT NULL ,
	[Descricao] [text] COLLATE SQL_Latin1_General_CP1_CI_AI NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

CREATE TABLE [dbo].[Processos] (
	[idProcesso] [int] IDENTITY (1, 1) NOT NULL ,
	[Nome] [char] (40) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[idOrganizacao] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Projetos] (
	[idProjeto] [int] IDENTITY (1, 1) NOT NULL ,
	[nome] [char] (40) COLLATE SQL_Latin1_General_CP1_CI_AI NULL
```

```
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Recursos] (
	[idRecursos] [int] IDENTITY (1, 1) NOT NULL ,
	[idOrganizacao] [int] NULL ,
	[Descricao] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Solucoes] (
	[idProblema] [int] NULL ,
	[idSolucao] [int] NOT NULL ,
	[Descricao] [text] COLLATE SQL_Latin1_General_CP1_CI_AI NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

CREATE TABLE [dbo].[TarefaConhecimento] (
	[idTarefa] [int] NOT NULL ,
	[idConhecimento] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Tarefas] (
	[idProcesso] [int] NOT NULL ,
	[idTarefa] [int] IDENTITY (1, 1) NOT NULL ,
	[Nome] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AI NOT NULL ,
	[Objetivos] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Fluxo] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Objetos] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[TempoControle] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Recursos] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Qualidade] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[ExecutadaPor] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Localizacao] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Intensiva] [bit] NULL ,
	[GrauSignificancia] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Transacao] (
	[idTransacao] [int] NOT NULL ,
	[idAgenteEnvia] [int] NOT NULL ,
	[idAgenteRecebe] [int] NOT NULL ,
	[objeto] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[planoComunicacao] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[restricoes] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AI NULL ,
	[Especificacao] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AI NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[mensagens] (
	[message_id] [int] IDENTITY (1, 1) NOT NULL ,
	[message_text] [char] (60) COLLATE SQL_Latin1_General_CP1_CI_AI NULL
) ON [PRIMARY]
GO
```